UNIVERSITE PARIS 8 - VINCENNES→SAINT-DENIS

U.F.R. 6

THESE
pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE PARIS 8
Spécialité Informatique

Jean-François LUCAS

Modéliser le fonctionnement cognitif d'agents sociaux et reproduire des phénomènes linguistiques

Simulations d'agents sociaux situés en survie dans un monde digital,
pour modéliser le fonctionnement cognitif d'agents coopératifs
et reproduire des phénomènes linguistiques.

Thèse dirigée par Gilles BERNARD

Soutenue le lundi 13 décembre 2010

Jury :

Herman AKDAG, Rapporteur
Tristan CAZENAVE,
Jacques FERBER, Rapporteur
Patrick GREUSSAY,
Herbert STOYAN, Rapporteur
Harald WERTZ,

UNIVERSITE PARIS 8 - VINCENNES → SAINT-DENIS

U.F.R. 6

THESE
pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE PARIS 8
Spécialité Informatique

Jean-François LUCAS

Modeling a cognitive functioning of social agents and reproducing some linguistic phenomena

Simulations of social situated agents, surviving in a digital world,
to model the functioning of cognitive cooperating agents,
and to reproduce some linguistic phenomena.

Thesis directed by Gilles BERNARD

Sustained Monday the 13th, December 2010

Jury :

Herman AKDAG, Rapporteur
Tristan CAZENAVE,
Jacques FERBER, Rapporteur
Patrick GREUSSAY,
Herbert STOYAN, Rapporteur
Harald WERTZ,

Résumé

Notre progression de simulations d'agents sociaux situés dans un monde digital modélise le fonctionnement cognitif d'agents coopératifs et reproduit des phénomènes linguistiques.

Nous construisons pas à pas une progression d'agents situés en survie sur un support. Nous partons de la cellule végétative vue comme un séquenceur sur un chemin, puis enrichissons le support : nous plaçons des obstacles sur une grille. Pour ce faire, nous utilisons le modèle causal et déterministe de l'automate fini (AF), que nous simplifions en un agent réactif empirique. Ainsi, en formalisant la construction de l'agent situé (AS), nous décrivons un passage d'un domaine déterminisme à celui de l'empirisme. Puis nous introduisons la mémorisation, pour construire un agent intentionnel (AI) et continuons par quelques agents croyances désirs intentions (CDI). Finalement nous évoquons un peu les agents rationnels cognitifs (ARC).

Notre but : exhiber dans notre taxinomie des exemples de simulations significatives. Nous les construisons en guidant une émergence selon le trajet AF→AS→AI→Agent CDI→ARC. Dans cette progression, les agents évoluent en héritant de leurs parents. Ainsi ils se classent naturellement dans un arbre phylogénétique.

Cette plate-forme de recherche est aussi proposée pour l'enseignement. Le mécanisme interne des agents est écrit en C ; il est épuré, simple et accessible : il peut servir à enseigner l'informatique et l'IA au moyen d'exemples concrets. Notre outil s'inscrit dans le long terme : il est ouvert et peut être repris et prolongé par ceux qui voudront le faire évoluer.

Summary

We provide a progression of simulations. Each one is modelling social agent(s) situated in a digital world. Some of them are modelling the functioning of cognitive cooperatives agents, and others are reproducing some linguistic phenomena.

We are building, step by step, a progression of simulations of situated agents (SA), surviving on support. We start from vegetative cell, seen like a sequencer on a path. Then, we enrich the support, we put obstacles on the grid. To do so, we use the causal and deterministic model of the finite state automaton (FSA), which we simplify into an empirical reactive agent. So, by formalizing the situated agent construction, we describe a transformation from a deterministic field to an empirical one. Then we introduce memorizing, to build an intentional agent (IA). We go on and construct some belief desire intention (BDI) agents. Finally, we scarcely evocate some rational cognitive agents (RCA).

Our goal is to exhibit, within our taxonomy, some examples of significant simulations. We are building them, guiding an emergence, along the path FSA→SA→IA→BDI Agent→RCA. Along this progression, agents are evolving, inheriting from their parents. That's why we are naturally able to classify them in a phylogenetic tree.

This research workbench is also provided to teaching. The agent's internal mechanism is written in C ; it is expurgated, simple and accessible : it can be used to teach computer science and artificial intelligence by means of concrete examples. Our tool is designed for future : it's open and can be re-used by those who will to make it evolve.

Ch. 1 Résumé synthétique de cette thèse

Sujet de thèse (Inscrit le 1 11 2004 - Discipline : M.I.A.S.S.) :

Interprète pour modéliser le fonctionnement cognitif d'agents coopératifs et reproduire des phénomènes linguistiques

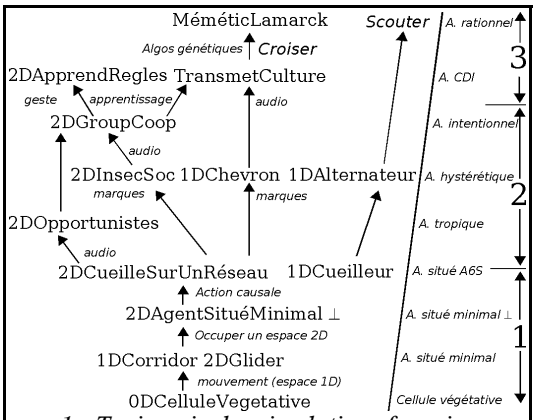
Sujet de thèse 2010 :

Simulations d'agents sociaux situés, en survie, dans un monde digital

Plate-forme informatique constituée d'une progression de simulations d'agents sociaux situés en survie (*A6S*) dans un monde digital, pour modéliser le fonctionnement cognitif d'agents coopératifs et reproduire des phénomènes linguistiques.

1) Cadre mathématico-informatique :

Sur un plan informatique, nous travaillons avec des agents situés en survie dans un monde digital. Par ailleurs, ce travail est étayé par une partie théorique qui s'inscrit dans un cadre mathématique.



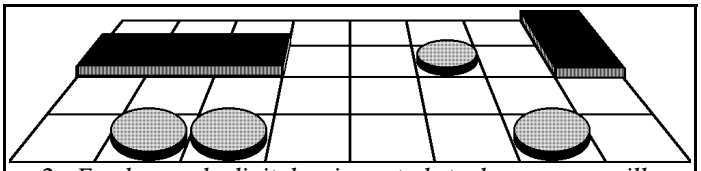
1 - Taxinomie des simulations fournies

Cadre mathématique des systèmes formels.

Nous utilisons les prédicats de la logique pour représenter le monde au moyen d'une liste de relations décrivant son mobilier posé sur un support digital. Ensuite les systèmes formels nous servent à simuler des agents qui le transforment au moyen de leurs actions.

Le cadre informatique de la modélisation/simulation.

Nous fournissons à l'utilisateur une plate-forme informatique constituée d'une progression de simulations d'agents évoluant dans ce cadre.


2 - Ex. de monde digital : pions et obstacles sur une grille

Le cadre des *agents situés, en survie,* sur un support Simple (*A6S*).

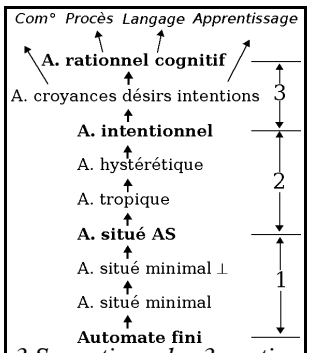
Un *A6S* est constitué de six facettes :

- Agents	: ils agissent, leurs actions, sous-tendues par un but direct ou indirect, transforment l'état du monde.
- Somatiques	: ils possèdent un soma, i.e. un corps, qui gère de façon autonome ses états internes (faim soif chaud...)
- Situés	: les agents sont placés sur/dans un monde digital (grille ou graphe).
- Survie	: ceux qui survivent se reproduisent, ils subissent une pression de sélection.
- Sur un Support	: ils sont placés sur un support de lieux digitaux. La portée de cette relation est limitée par effet d'horizon.
- Simple	: ils entretiennent une relation simple avec leur support : elle se limite à la case sur laquelle ils sont situés.

2) Problématique : livrer à l'utilisateur un outil : une plate-forme

Notre travail se situe dans le domaine de l'intelligence artificielle distribuée (IAD), et plus précisément dans celui des agents situés (AS), agents intentionnels (AI) et agents rationnels cognitifs (ARC). Au départ, il est basé sur l'automate fini (AF) et notre problématique est de formaliser et illustrer les passages $AF \rightarrow AS$, puis $AS \rightarrow AI$, enfin $AI \rightarrow ARC$.

Pour ce faire, d'abord nous formalisons le passage $AF \rightarrow AS$ au moyen des mathématiques, puis nous balisons le chemin $AS \rightarrow AI$ au moyen d'agents situés sub-symboliques dont nous décrivons la structure informatique. Finalement, dans la transition $AI \rightarrow ARC$ nous présentons des applications de phénomènes cognitifs et langagiers.



3 Synoptique des 3 parties

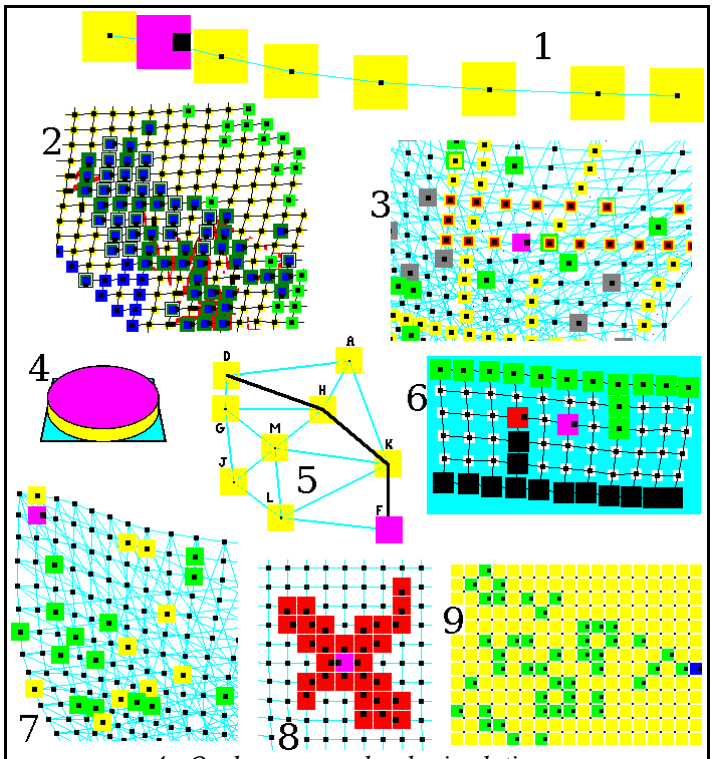
2.A) Fournir un outil pratique

Nous fournissons une plate-forme assez pratique à utiliser pour plusieurs raisons :

Nous fournissons une boîte à outils.

Cette plate-forme est constituée d'une progression de programmes exemples basés sur les agents réactifs. Elle s'étage, depuis la simple cellule végétative (4), jusqu'à des systèmes cognitifs (2, 6).

1 – *1DCorridor* 2 – *Mémétic*
3 – *2DInsecSoc* 4 – *0DCellule végétative*
5 – *Scouter* 6 – *2DApprendRègles*
7 – *2DCueilleSurNet* 8 – *2DOpportunistes*
9 - *Croiser*



4 - Quelques exemples de simulations appartenant à la progression fournie

Nous livrons un outil pratique.

Assez simple à utiliser, il implémente, au moyen de *gcc* et *SDL*, le concept des *A6S*. Le code des programmes correspondants est simple et léger, à dimension humaine : il tient dans seulement quelques pages de listing.

Un domaine prolifique et gratifiant.

Les utilisateurs reçoivent un outil qu'ils peuvent ensuite prolonger. Certes, ils doivent mettre les mains à la pâte et faire preuve d'imagination afin de programmer des modèles, mais la technique des *A6S*, fournit des simulations prolifiques et gratifiantes car elles peuvent modéliser différentes situations de la vie.

Un outil cohérent et harmonieux.

L'outil fourni aux développeurs est cohérent et harmonieux simplement parce qu'il est construit par émergence guidée à partir de la même structure de base : la cellule végétative. Il est spécialement dédié à cette tâche. En le réalisant, nous avons intégré dans le système uniquement les éléments requis.

Dans le cadre des *A6S* = *AS* + *Vie artificielle* + support digital.

Au départ, cet outil s'inscrit dans un cadre large et général : celui des agents. Mais ensuite s'ajoutent deux spécifications qui restreignent un peu son ouverture : l'agent est situé dans un monde digital, où il doit survivre.

Mais en fait, le domaine initial est si vaste, qu'après ces restrictions, il demeure large et général, au point qu'on peut encore y effectuer un travail adaptable et fédérateur : l'utilisateur peut y simuler beaucoup de modélisations différentes.

2.B) Notre cible : l'informatique distribuée et les sciences humaines et sociales

Nous travaillons à décrire les prémisses de la cognition et de la communication chez des agents sociaux. Pour ce faire, nous restreignons notre modélisation à des agents réactifs, et seuls les plus complexes sont enrichis de croyances et de méta-regard. Ce choix de la simplicité s'explique par les deux créneaux que nous visons : nous souhaitons continuer d'intéresser les chercheurs de l'intelligence artificielle distribuée, tout en réalisant des simulations à taille humaine destinées aux praticiens des sciences sociales.

Un outil simple pour traiter la cognition.

Un de nos buts est de toucher, d'aborder la cognition (le traitement rationnel cognitif de l' information). Certes cela constitue une tâche difficile, mais au moyen d'un peu de mathématiques et d'informatique, nous fournissons quelques simulations de traitements cognitifs, certes ponctuels et simplifiés, mais non triviaux.

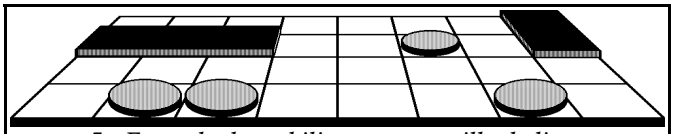
Fournir un outil qui modélise des phénomènes langagiers.

Les plus élaborés de nos agents sont sociaux : il communiquent, transmettent et échangent leur savoir. Nous fournissons ce cadre prolifique à l'utilisateur pour lui permettre de recréer des phénomènes langagiers. Alors, quand il réussit à faire fonctionner une telle simulation, il peut ensuite facilement en disséquer les mécanismes de ses modèles pour en analyser minutieusement les rouages.

3) Modéliser les agents classés dans cette taxinomie

Dans la partie 1 : AF→AS, nous simplifions à outrance.

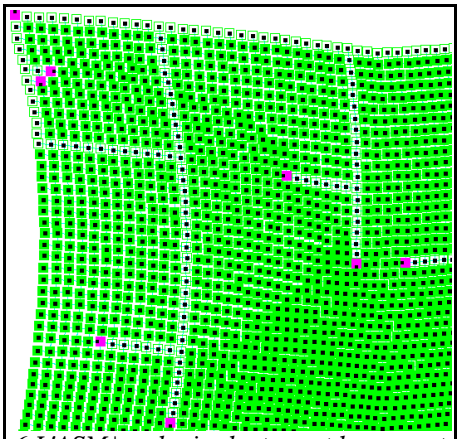
Le monde simulé est constitué d'un support, qui peut être un graphe ou une grille de lieux ou de cases. Sur chacun est posé une pièce de mobilier (agent nourriture obstacle). Lieu ou pièces de mobilier, chacun a la forme d'un pion ou d'une case.



5 - Exemple de mobilier sur une grille de lieux

Cette organisation induit des relations simples entre le support et son mobilier : nous mettons un meuble par lieu et réciproquement. De plus, ici, nous faisons le choix d'un effet d'horizon dans les relations : nous focalisons sur les relations de voisinage et délaissons les interactions à distance.

Les procès, les transformations, effectuées par les agents correspondent à des variations de ces relations. Elles dépendent de deux aspects : du comportement des agents au vu de leur environnement, et des réponses du monde aux actions des agents.



6 L'ASM⊥ colonise lentement le support

En utilisant ces représentations simplifiées, nous obtenons des résultats sur deux plans : - D'abord, à l'issue de cette transformation *AF→AS*, nous exhibons un agent situé minimal orthogonal (ASM⊥) d'une grande simplicité, cependant il est capable de se reproduire et de coloniser un support 2D. - Ensuite, en focalisant sur la transformation *AF→AS*

nous disséquons et analysons le passage d'un traitement déterministe et complet vers un traitement non déterministe. Sur le support, cet empirisme est dû à deux manifestations de l'effets d'horizon. Il limite aux cases voisines, d'abord les relations de voisinage entre les lieux, et ensuite la perception visuelle de l'agent. Par ailleurs, au sein de l'agent, nous trouvons encore un troisième empirisme, qui est dû à la façon dont cet être déclenche ses actions : pour connecter ses perceptions aux actions, il utilise une stratégie incomplète, basée sur un parcours en largeur d'abord.

Dans la partie 2 : *AS→AI,* nous construisons progressivement l'AI.

Dans la seconde partie de notre travail, en partant de l'ASM⊥, un AS élémentaire, nous construisons progressivement l'AI. À touches pointillistes, nous ajoutons les capteurs, actionneurs et règles selon une stratégie en largeur d'abord. Pour ce faire nous prospectons avec éclectisme dans toutes les directions (espace 0D, 1D et 2D ; temps (mémorisation) ; vue ; ouïe).

La partie 3 : AI→Agent CDI→agent rationnel cognitif (ARC).

Dans la troisième partie de notre travail, depuis l'AI nous explorons un peu en direction de l'Agent CDI, puis de l'agent rationnel cognitif. Nous mettons en oeuvre la vie artificielle qui permet une reproduction sélective, et le parallélisme qui fait apparaître le groupe car il induit la communication. Nous illustrons cette recherche en présentant des procès rationnels cognitifs, et des phénomènes de communication par gestes, cris et langage.

4) État de l'art :

4.A) Situer ce travail selon un point de vue mathématique

Ce travail emprunte aux mathématiques à la fois la représentation des connaissances et la logique descriptive classique.

Nous simulons un monde constitué d'un mobilier placé sur un support de lieux. Nous représentons les connaissances à son propos au moyen des relations, prédicats et fonctions...

Mais pour raisonner, pour transformer la représentation que nous entretenons du monde, nous délaissons le modus-ponens, car il correspond seulement à l'affinement de notre connaissance de des propriétés de ce monde. En effet, pour nous, le fait d'approfondir notre savoir, i.e. les subtilités du lien *sens-référence* constitue une préoccupation secondaire. Car en fait, afin d'évacuer ces problèmes de représentation, nous choisissons de placer les agents dans un monde simplifié et digital.

Finalement, notre but ne consiste pas à représenter un monde statique, mais nous portons plutôt notre intérêt sur sa représentation dynamique, i.e. sur la transformation de son état par les actions des agents.

Ce travail emprunte aux systèmes formels des mathématiques.

Nous effectuons un traitement séquentiel du monde au moyen des actions, i.e. nous représentons son évolution, au fil du temps, sous l'action des agents. Pour cela, nous empruntons la règle de réécriture aux systèmes formels des mathématiques et nous l'utilisons abondamment. Ce choix s'effectue au détriment du modus ponens de la logique descriptive statique, que nous délaissons.

Ce travail apporte et emprunte aux systèmes multi-agents (SMA).

La partie théorique de ce travail apporte aux SMA en montrant comment construire un agent situé, à partir de l'*automate à états finis*.

Par opposition aux projets industriels qui généralement sont bien plus complexes, ce travail apporte une théorie, une pratique et une pédagogie pour des applications simples des SMA. En effet, au départ, il est basé sur les agents réactifs, puis ensuite, seulement sur les *agents Croyances* munis de méta-regard.

4.B) Situer ce travail selon le point de vue de l'intelligence artificielle (IA)

Une nouvelle synthèse, vu les découvertes de ces dernières années.

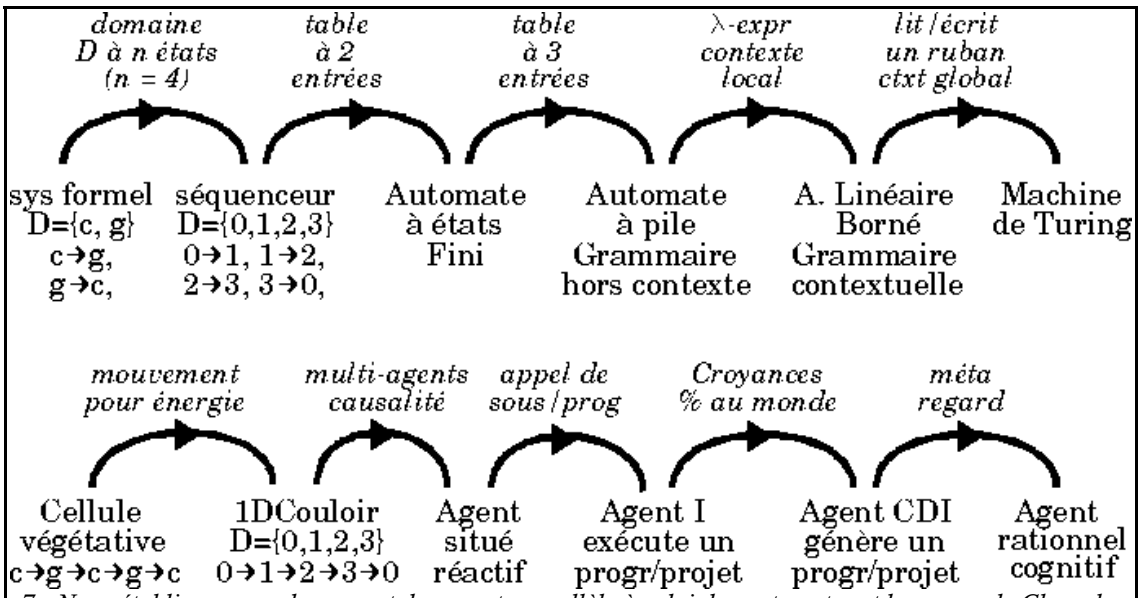
Aux début de l'IA, il suffisait d'avoir une idée et on se laissait porter par la vague. Maintenant, que plusieurs générations de chercheurs ont enrichi le domaine, ça devient difficile d'améliorer des thèmes qui ont été perfectionnés pendant 30, 40 ou 50 années de recherche.

Le domaine des agents situés est plus récent, et déjà beaucoup traité. Certes, on peut encore l'améliorer ponctuellement, en prolongeant certaines de ses feuilles, ou en précisant certains de ses domaines. Mais l'autre direction/solution peut consister à trouver un nouveau positionnement, en prenant du recul afin d'établir des passerelles entre disciplines. Ainsi ce travail, à la lumière des découvertes de ces dernières années, propose une nouvelle synthèse informatico-mathématique : elle établit une passerelle entre, d'un côté, les agents puis l'IAD ; et de l'autre, le connexionisme, i.e. les réseaux de neurones et la RC sub-symbolique qu'il induit.

Situer ce travail relativement à la classification de Chomsky.

Le travail de Noam Chomsky tire sa cohérence et son intérêt de la classification des grammaires des langages (naturels ou de programmation) et des différents types d'automates et machines correspondants.

Dans notre travail, nous classifions les *A6S* selon la façon dont ils traitent l'information pour programmer des actions cohérentes et amples. Et, ce faisant, nous retombons bien sur les résultats de Chomsky.



7 - Nous établissons un classement des agents parallèle à celui des automates et langages de Chomsky

Situer ce travail relativement aux multi-agents, à l'IAD.

Dans son livre *les systèmes multi-agents*, Jacques Ferber effectue une classification plutôt exhaustive des différents agents, mais, par la force des choses, on se perd un peu dans cette immensité. Bien sûr il y utilise un formalisme général, et des conventions de notations adaptées (le formalisme Bricks), mais le plan général de son livre disparaît sous l'abondance de la matière. Par ailleurs, je n'ai pas trouvé dans son livre de justification théorique du passage de l'AF à l'agent situé. Cette démarche n'a pas été effectué dans le monde des agents réactifs. Et, dans le monde de l'industrie, qui travaille plutôt avec des agents plus complexes, elle apparaît moins nécessaire.

Dans notre travail, qui se veut cohérent plutôt qu'exhaustif, nous avançons deux buts : faire apparaître des phénomènes langagiers et modéliser une cognition simple. Pour cela, nous nous plaçons délibérément dans le monde restreint des agents situés, pour lesquels nous proposons une démarche formalisée et cohérente, qui demeure cependant assez générale.

5) Méthodes : construire un modèle de traitement cognitif au moyen d'*A6S*

5.A) Problèmes de représentation des connaissances induits par notre approche

a) S'interdire la représentation des connaissances(RC) symbolique de Von Neumann

Notre but est aussi de modéliser certains aspects de la cognition humaine. À ce titre, dans nos simulations, nous choisissons de délaisser la RC symbolique formelle classique[1], car elle se situe vraiment trop loin du traitement effectué par le cerveau humain. En effet, chez l'homme, on n'observe pas la moindre fonctionnalité qui ressemblerait à une RAM, un tableau, une pile ou un registre.

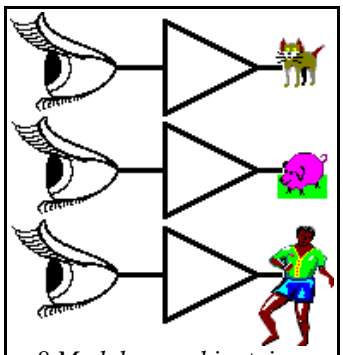
Donc nous nous interdisons ce type de symboles formels. Mais en procédant ainsi, nous perdons tout ce qui en découle. Nous éliminons du même coup la mobilité des instances de symboles. En effet, si les symboles n'existent plus, on ne peut pas les instancier, et leurs instances ne peuvent pas être mobiles. Si par exemple, nous nous interdisons d'utiliser le symbole 123, alors l'ordinateur ne peut pas l'instancier pour le dupliquer et lui faire traverser le bus de données afin de le ranger dans la RAM.

Pour la même raison, nous devons délaisser le traitement des données dans les unités arithmétiques et logiques (UAL). Ainsi, de proche en proche, tout le fonctionnement de l'ordinateur de Von Neumann nous devient interdit.

Mais alors que faire ? Nous devons bien remplacer cette RC par une autre !

b) Utiliser la RC de la logique combinatoire classique

Pour résoudre ce problème de mobilité des symboles, nous choisissons le traitement câblé et sub-symbolique de la logique combinatoire classique.



8 Modules combinatoires d'analyse d'images

Représentation.

En effet, si par exemple, un module d'analyse de scène perçoit, devant l'agent, la présence d'un chat, alors il fournit au niveau de sa sortie, un digit qui porte <u>implicitement</u> le symbole *chat*.

Transport.

Ensuite, pour transporter plus loin ce sub-symbole implicite, il suffit d'utiliser le digit qui porte ce résultat, pour lui connecter un fil conducteur : la tension électrique chemine le long de cette connexion et devient disponible ailleurs. Ainsi, en logique combinatoire câblée, le transport d'un sub-symbole s'effectue simplement au moyen d'un connecteur électrique.

Traitement.

De plus, si le capteur d'entrée est numérique ou analogique, si par exemple il mesure la quantité de nourriture, alors l'amplitude de la perception peut ensuite subir un traitement arithmétique : elle peut être additionnée, soustraite, ou multipliée par une constante.

Mémorisation.

Finalement, nous pouvons proposer des schémas de mémoire RAM et ROM pour cette RC. D'abord, dans une de nos simulations (*2DApprendRègles*) nous montrons un exemple où le réseau de neurones de l'agent apprend par renforcement de coefficients. Ceci constitue un apprentissage vers une ROM.

Nous disposons aussi de schéma pour réaliser une mémoire en connectant dynamiquement deux feuilles d'arbre, ce qui permet d'implémenter une RAM dans le cadre d'une représentation sub-symbolique.

1 ... et le traitement de ces connaissances ainsi représentées.

c) La propagation de marqueurs :

De plus nous avons en projet l'étude d'une RC séquentielle modulée, basée sur la propagation de marqueurs et le procès correspondant.

Représentation.

Voici globalement comment procéder : dans la représentation sub-symbolique câblée précédente, l'amplitude du signal électrique fourni est constante. Maintenant, il s'agit donc de lui ajouter un codage périodique d'amplitude : chaque information est codée par un signal modulé en tension.

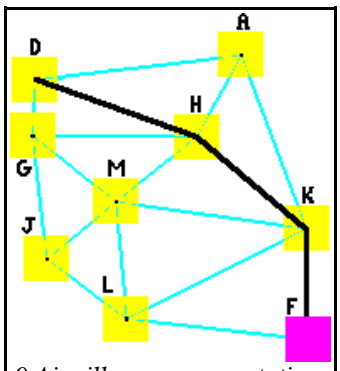
Transport.

Puisque l'information est codée en amplitude et porté par un fil conducteur, son transport continue de s'effectuer simplement au moyen d'un connecteur électrique.

Traitement.

On peut imaginer au moins deux sortes de traitements : identification et mémorisation/traduction. Un traitement d'identification formelle de l'information peut se réaliser au moyen d'un aiguillage par commutation. Cette démarche est dégrossie dans le programme exemple *Scouter*.

Ainsi cette information codée formellement acquiert la dimension d'un symbole.



9 Aiguiller par commutation, depuis D, *jusqu'à* F

Mémorisation/traduction.

Mémorisation : au terme de l'identification du symbole, le système mémorise le parcours en renforçant les coefficients des synapses (en augmentant leur sensibilité au phonème du signal). Il renforce aussi le poids du lieu d'arrivé.

Traduction : au terme de l'identification du symbole, le système déclenche la modulation d'un autre symbole.

5.B) Méthode : une recherche ouverte, une démarche éclectique

a) Une démarche opportuniste et éclectique

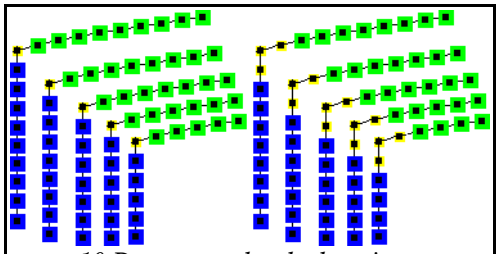
Notre démarche n'est pas directement téléologique.

Elle n'est pas directement orientée vers un cahier des charges. Nous ne nous sommes pas dit : comment arriver à ce but. Elle est plutôt opportuniste et éclectique.

Imaginer des exemples de programmes à insérer dans notre progression.

Donc, au départ, nous avons construit un système, assez contraint car il est formalisé en maths ; et avons dit : que faire de simple dans ce cadre des *A6S* ? Et nous avons recherché/imaginé des exemples simples de programmes spectaculaires qui pourraient correctement s'insérer dans notre progression de structures, à la suite de l'ASM⊥.

b) À la recherche d'un thème de programme


10 Deux exemples de domaines compliqués, tirés de Chevron

À l'intérieur du domaine des *A6S,* le thème est laissé libre.

Dans cette recherche, seul l'outil de programmation est fixé : il s'agit de travailler dans le cadre des *A6S*. À part cette contrainte, le programmeur dispose d'une grande liberté : le sujet qu'il doit traiter est souple, modifiable et aménageable car le thème de l'exemple est laissé libre. Il peut encore modifier d'autres variables d'ajustement, entre autres le domaine où sont situés les agents. Par exemple, il peut en tailler un à sa guise.

Ainsi, dans notre progression de simulations, se trouve le programme *1DChevron* qui propose un domaine compliqué, et nous permet d'obtenir un agent simple.

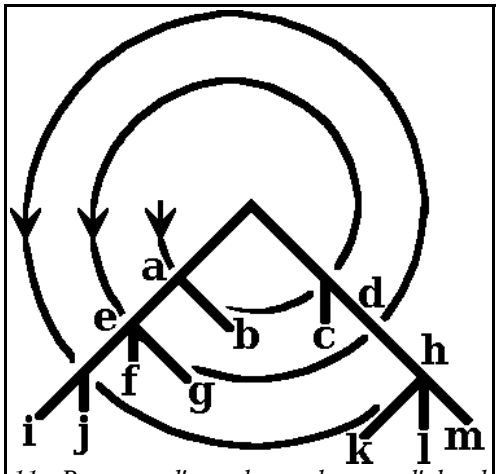
Trouver un thème qui s'insère bien dans la progression de structures d'agents *A6S*.

Notre but est de fournir à l'utilisateur une progression de simulations d'*A6S*. Nous la construisons graduellement à partir de l'ASM⊥: quand nous avons trouvé un nouvel exemple de simulation, nous l'ajoutons, nous le classons au sein de notre taxinomie. Donc, pour effectuer ce travail, nous recherchons des exemples d'*A6S*. Chacun doit s'insérer dans cette progression, à condition qu'elle demeure cohérente, i.e. sans large fossé : la nouvelle simulation doit se positionner ni trop loin, ni trop près de chacune des autres.

5.C) Notre démarche : ratisser large pour sélectionner

Au départ, nous disposons d'une taxinomie de simulations, et nous cherchons à l'enrichir, i.e. à y ajouter une nouvelle simulation, construite en améliorant un de ses modèles viables.

Voici comment nous procédons pour construire la structure et le câblage neuronal de chaque modèle inséré dans notre progression de simulations. Il doit être polyvalent afin de ne pas présenter de faiblesse, i.e. d'éviter la présence de défaut à sa cuirasse. À partir de la structure de cet agent, nous travaillons à l'améliorer, i.e. nous aidons artificiellement l'émergence en guidant nous-même son amélioration.



11 - Parcours d'un arbre en largeur d'abord : on relève les étapes a, b, c, d ... k, l, m.

Pour ce faire, nous avons le choix parmi différentes fonctionnalités distribuées selon deux directions. Horizontalement nous pouvons choisir parmi les diverses directions correspondant en général à des grandeurs physiques. Verticalement nous choisissons au sein d'une architecture en couches : au départ, une fonctionnalité élémentaire ; puis, la même améliorée ; ensuite une couche constituée par un ensemble de ces fonctions, puis plusieurs couches...).

5.D) Démarche éclectique : sans vergogne, nous empruntons tout azimut

Alors, parmi toutes ces briques élémentaires disponibles, pour créer une nouvelle simulation, nous sélectionnons, à touches pointillistes, quelques fonctionnalités que nous ajoutons à la structure déjà opérationnelle de cet agent. Ce choix s'effectue selon une stratégie en largeur d'abord (figure 10): nous travaillons en prospectant horizontalement. Ceci consiste à améliorer d'un cran vertical toutes les fonctionnalités existantes, et, quand c'est possible, à rajouter horizontalement une nouvelle directions.

C'est ainsi que nous construisons progressivement notre taxinomie au moyen de ces nouveaux prototypes d'agents : nous les plaçons en position de feuille, au sein de notre arborescence. Et, après un cycle, au passage suivant, quand la feuille s'est développée verticalement, alors à son tour, elle peut porter, soit une nouvelle feuille si elle est devenue branche, soit plusieurs si elle est devenue fourche.

a) Une concentration verticale pour une organisation en pelure d'oignons

Nous reconstruisons la structure physique des agents en couches concentriques, i.e. verticalement, de l'atome initial vers l'assemblage complexe. À la base nous commençons avec une fonctionnalité élémentaire, puis nous reprenons la même en l'améliorant, et ensuite en la groupant : nous utilisons une couche constituée par un ensemble de ces fonctions, puis plusieurs couches...

Au début de notre progression, nous commençons par les systèmes formels, nous mettons les règles dans des tables à 1, 2 ou 3 entrées. Ainsi nous fabriquons des séquenceurs, puis des

automates à états finis. Ensuite elles nous servent, pas à pas, à construire des automates à registres, à pile...

b) Des emprunts horizontaux dans toutes les directions

Sans vergogne, nous continuons d'emprunter aussi horizontalement.

D'abord, dans la partie 1 de notre taxinomie, nous partons en direction de diverses grandeurs physiques :

Dans une démarche pratique, pragmatique, nous plongeons l'automate dans un monde digital et débouchons sur l'agent situé. Ainsi, nous construisons ce support où évoluent les agents situés, en partant d'un espace 0D, i.e. sans dimension. Puis nous ajoutons d'abord une dimension pour obtenir un trajet 1D. Ensuite nous utilisons une seconde dimension pour disposer d'une grille ou d'un réseau.

Puis, dans la partie 2 de notre taxinomie, nous empruntons plutôt des techniques informatiques :

- Vie artificielle (VA): En introduisant la VA, nous obtenons des agents en situation de pénurie sur un support où ils cherchent de la nourriture pour survivre. Cette quête donne un sens à leur vie, et par ce biais nous évaluons l'efficacité de leur fonctionnement et atteignons une sémantique pratique.
- Les agents sociaux de l'intelligence artificielle distribuée (IAD) : À l'instar de l'IAD, nous introduisons les multi-agents. Ils induisent encore un effet holistique, celui du groupe. Ainsi nous obtenons des agents qui se mettent à communiquer, et deviennent sociaux.
- Une représentation de la connaissance adaptée : Avec opportunisme, nous choisissons une représentation de la connaissance adaptée : le monde où évoluent les agents, est digital afin d'y simplifier les interactions ; mais le traitement de l'information *perception décision action* effectué par les agents réactifs peut être sub-symbolique : il peut être implémenté au moyen de la logique combinatoire, mais aussi des réseaux de neurones.

5.E) Aspect holistique du fonctionnement *soma+psyché* d'un agent

L'assemblage de la structure de l'agent est complexe : afin de modéliser un agent capable d'effectuer un traitement efficace nous devons au moins assembler dans sa structure tous les éléments nécessaires à son fonctionnement i.e. le soma et la psyché.

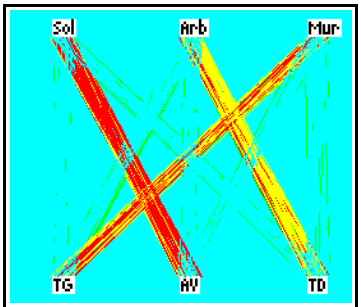
Le fonctionnement de l'agent présente une efficacité holistique. Si un des maillons de la chaîne manque, l'assemblage global est défaillant. Ainsi, avant que nous obtenions un fonctionnement global, trois fonctionnalités doivent être mises au point : le corps, son assemblage et sa programmation. Sa structure corporelle, i.e. ses composants physiques doivent être complets ; leur interconnexion doit être correctement effectuée ; et le comportement de l'agent, i.e. la programmation de ses action doit être juste.

Cette triple contrainte explique notre façon de procéder. Notre travail de reconstruction est basé sur une structure très simple, dont nous guidons pas à pas l'émergence. Nous partons de la cellule végétative, dont nous maîtrisons le fonctionnement, et à chaque étape nous améliorons l'agent de base de la simulation, et le classons dans notre taxinomie.

5.F) Concept utilisés : nécessité d'un éclectisme

a) Le problème de l'abondance de matière

Nous construisons l'arborescence de notre taxinomie d'abord en largeur, puis verticalement : nous en empruntons à gauche et à droite toutes les avancées qui peuvent induire une amélioration. Finalement, cette prospection verticale et horizontale recueille une matière si abondante que nous devons choisir avec parcimonie les composants de notre assemblage.



12 - L'apprentissage des règles de comportement commence par des connexions directes

b) Construire en parcourant un arbre selon une stratégie en largeur d'abord

Dans notre travail de modélisation de l'agent, nous utilisons plusieurs fois la stratégie de construction selon un parcours en largeur d'abord. Elle est particulièrement visible en deux endroits : la construction de sa structure et la programmation de ses déplacements.

Amélioration des constituants physiques de l'agent.

De façon générale, cette stratégie apparaît nettement dans la construction de son architecture, tout au long de la progression (cellule végétative, séquenceur, être causal...). Son utilisation s'explique par la structure holistique de l'agent. À partir du moment où ce système se met à fonctionner, il présente une structure de chaîne à plusieurs maillons[2].

Pour continuer de la construire, i.e. pour améliorer son fonctionnement, nous choisissons de rectifier chacun d'eux, sans privilégier une direction plutôt qu'une autre. Pour nous, une bonne manière d'effectuer une telle modification consiste à utiliser la stratégie de reconstruction en largeur d'abord que nous avons décrite précédemment.

Programmation de la gouvernance : élaboration des règles de conduite *décisions.*

De façon particulière, la stratégie de parcours en largeur d'abord est encore plus visible dans l'exemple *2DApprendRegle*. En effet, elle y sert à construire, les règles de comportement de l'étage *perception décision action* à la manière des univers de Herbrand.

2 C'est l'effet du holisme dont nous parlions tout à l'heure.

6) Résultats : une plate-forme de simulation, fournie avec des exemples

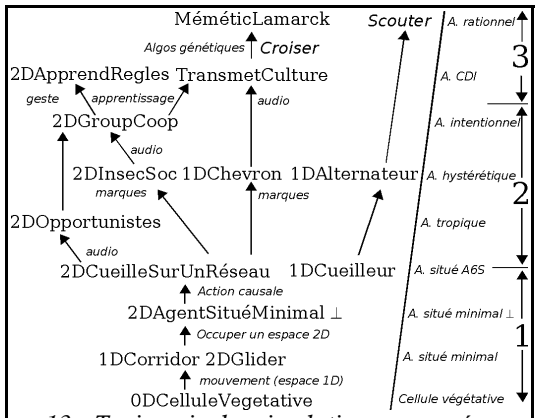
6.A) Comme résultats, nous fournissons des exemples de simulations

a) Partie 1 *AF→AS*, nous créons une progression de simulations vers le traitement empirique de l'AS réactif

Afin de préciser le cadre épistémologique, rappelons l'histoire de l'informatique :

D'un côté, dans le domaine de la perfection, nous avons certains traitements formels de l'IA symbolique (*arithmétique des entiers, déduction au moyen du syllogisme, algèbre de Boole, calcul des propositions...*) qui fournissent des résultats complets, parfaits et optimaux. Par exemple, précisons que selon l'étymologie, un *théorème*, c'est la *chose de Dieu*.

Ensuite, dans l'histoire des sciences, cette perfection est déjà entachée par l'*optimisation*, une discipline qui délaisse la recherche d'une solution parfaite, et se contente d'un extremum. Et parfois, pire encore, elle se satisfait même d'un extremum local.



13 - Taxinomie des simulations proposées

Puis, avec la vie artificielle (VA) viennent l'*évolution* et l'*émergence* qui continuent à dégrader la qualité de la prestation fournie à l'utilisateur. Par exemple, en simulant des agents situés sur un support, nous continuons à altérer la qualité du traitement de l'information et introduisons un procès empirique. Quand nous aidons ces êtres à évoluer, le résultat est encore moindre : d'abord ils partent de très bas car ils n'émergent que du chaos, et ensuite, ils se contentent de s'adapter, de s'améliorer.

Ainsi, dans ce cadre du traitement de l'émergence, nous disposons de simulations d'êtres empiriques. Nous pourrions penser qu'ils ne planent pas haut, cependant ils présentent de l'intérêt. En effet, voici qu'elles animent des êtres qui paraissent bien imparfaits. Ils possèdent seulement le désir de monter, de s'améliorer, de faire mieux que les autres, et pourtant, à l'issue de ce traitement, nous obtenons un résultat remarquable, ils s'avèrent adaptés pour modéliser des phénomènes nouveaux auxquels nous prêtons attention. Par exemple, ils modélisent plusieurs phénomènes de sciences humaines (organisation et communication au sein du groupe), et pour cela nous les jugeons dignes d'intérêt.

Finalement nous concluons que le domaine de l'émergence, même s'il est imparfait, ne doit pas être négligé, sous prétexte qu'il est empirique.

b) Dans la partie 2 *AS→AI*, nous simulons avec traitement réactif de l'information

Dans la partie 2 *AS→AI*, nous partons de l'AS, qui est un agent réactif au fonctionnement béhaviouriste : il boucle en permanence sur le traitement *perception décision action*.

Ensuite, nous lui ajoutons une petite couche mémoire, afin de simuler des traitements élémentaires de l'information, et ainsi construire une progression de simulations en direction des agents intentionnels. Ces traitements sont légers et donc facile à implémenter avec ces agents simples, il suffit de prolonger leur architecture en ajoutant un ou quelques drapeaux.

c) Partie 3, nous simulons un traitement de l'information et une communication :

Nous fournissons des simulations de traitement rationnel cognitif de l'information.

Au bout de l'arbre, dans la partie 3 *AI→A.CDI→ARC*, nous explorons le traitement rationnel cognitif de l'information. Dans ces simulations, des agents situés génèrent différents comportements plus élaborés : certains anticipent, l'un d'eux planifie.

Nous ne sommes plus en présence d'agents réactifs qui agissent instantanément, mais d'agents secondaires qui effectuent un traitement interne séquentiel de l'information, afin de planifier une séquence d'actions, et d'agir ensuite.

Pour ce faire, ils doivent disposer de mécanismes internes de simulation, et connaître l'état du monde, les actions qu'ils peuvent effectuer, et la réponse du monde à ces actions.

En fait, ces traitements sont lourds et donc difficiles à simuler car ils requièrent des agents complexes. Dans cette partie 3, nous proposons juste quelques simulations partielles qui dégrossissent seulement le problème.

Nous fournissons des simulations de *communication au sein du groupe*.

Cette communication prend plusieurs formes : geste, mot, phrase. Selon sa complexité, elle permet aux agents d'échanger des états internes, des informations de lieux, jusqu'à des règles de conduites. Dans ce dernier cas, nous touchons à des attitudes d'enseignement et d'apprentissage.

Ensuite nous nous posons la question du sens, nous évaluons la sémantique de ces organisations. Finalement, nous concluons qu'une communication de bonne qualité augmente l'efficacité du groupe, ce qui nous permet de conclure que la communication au sein du groupe possède un sens.

6.B) Comme résultats, nous fournissons des outils pédagogiques

Nous fournissons des outils qui modélisent des phénomènes de sciences humaines et sociales

Dans le cadre de la vie artificielle, nous construisons des simulations avec les agents situés, des êtres imparfaits, et notre but est seulement de les aider à émerger, à s'améliorer. Dans ce cadre, nous somme loin de l'idéal de perfection absolue des mathématiques formelles, mais, paradoxalement, nous ne regrettons pas cette situation. Au contraire, cette mouvance, qui mêle *émergence* et *empirisme*, ce domaine qui mélange *évolution* et *non-déterminisme*, s'avèrent intéressants car ils modélisent ce que nous cherchions : des phénomènes de sciences humaines et sociales.

Nous fournissons des modélisations simples et ouvertes de ces phénomènes humain et sociaux

Certes l'empirisme de la VA constitue un domaine non-déterministe, jusque là mal appréhendé. Mais maintenant, nous en possédons une version présentant trois aspects intéressants : simple, modélisée, et ouverte. D'abord elle simple car est construite progressivement depuis l'ASM$\perp$. Ensuite elle est modélisée car construite à partir des mathématiques, i.e. de l'AF. Enfin elle est ouverte, car nous pouvons soulever le capot pour regarder le moteur. Ainsi nous accédons à la connaissance de ses mécanismes, qui sont mis à nu et accessibles. À nous d'en explorer les rouages !

La transition $AF \rightarrow AS$ fournit une illustration du raisonnement inductif

De plus, la transition $AF \rightarrow AS$ requiert un monde muni d'un support très régulier. C'est à ce prix que les effecteurs et les règles d'action peuvent se simplifier. Ainsi par exemple, pour l' agent situé, nous obtenons cette règle classique de comportement : *quand la nature de la case située devant moi est du sol, alors je dois avancer*. En fait cette règle est pleine de déictiques (ici, devant, moi, je, en avant) et sa formalisation mathématique induit une étape de généralisation au moyen de variables. Sur un plan pédagogique, ceci permet d'illustrer l'opération, la démarche d'induction en mathématiques.

6.C) Comme résultats, nous modélisons des phénomènes de communication

a) Nous modélisons des transmissions d'information à propos du monde

Nous commençons par présenter ou seulement évoquer des communications non langagières. Nous regardons tout d'abord celles qui se réduisent à des cris : ces phénomènes de communication, sont parmi les plus simples.

Ils codent plutôt des états internes aux agents : *il y a danger quelque part*, *passer en état d'alerte*, *je suis en état de détresse / quiétude / bien-être*.

Ensuite, à ces cris, s'ajoutent implicitement des informations tropiques qui codent plutôt des déictiques : *nourriture ici*, *moi je suis ici*, *au secours à moi*.

b) Nous modélisons des phénomènes langagiers

Puis nous continuons en présentant une communication non élémentaire, utilisée au sein du groupe social. Elle est basée sur un langage (articulée ou gestuel). Elle sert à transmette une information plus complexe, i.e. des faits à propos du monde ou des règles pour y vivre en situation.

Nous proposons deux exemples relatifs à des règles de déplacement : la simulation d'une transmission langagière de règles d'action, et l'enseignement de ce même apprentissage, au moyen de l'exemple du geste corporel.

Par ailleurs, nous évoquons seulement la transmission de règles décrivant la réponse du monde à l'action d'un effecteur.

c) Nous modélisons une autre sémantique : celle qui évalue une fonction de transfert

Selon l'approche classique des mathématiques, la sémantique peut être vue comme une fonction de transfert : elle correspond à la table de vérité d'une fonction logique booléenne.

Au contraire, selon notre approche, nous évoquons l'introduction d'un autre point de vue, où la sémantique est plutôt considérée comme l'évaluation, le test de cette même fonction, en la mettant en situation dans le monde réel. Elle exprime que la structure testée possède une efficacité dans la vie quotidienne. Ainsi elle se trouve plus conforme à l'intuition de l'homme de la rue, et elle rejoint la notion de *comportement sensé*, i.e. qui *possède un sens* selon la psychologie naïve.

d) Nous modélisons les déictiques de la pragmatique linguistique

Au sein de plusieurs de nos simulations, nous mettons en scène des agents en situation de survie sur un support où est réparti du mobilier. Souvent nous utilisons de simples agents réactifs qui fonctionnent en itérant la boucle *perception décision action*.

Quand nous analysons les connexions qui relient leur perception à leurs actionneurs, nous pouvons les verbaliser en pseudo-langage : *maintenant, quand dans la case ci-devant, il n'y a rien (c'est du sol), alors j'avance*. *Maintenant, quand dans la case devant moi se tient un obstacle, alors je tourne à droite*. Alors nous remarquons déjà la présence de quelques-unes des déictiques les plus communes : *ici et maintenant, moi et lui, ci-devant*.

7) Évaluation des résultats :

Ce travail consiste en une progression d'une quinzaine de simulations, où les premières sont très simples, les dernières un peu plus complexes ; cependant il répond bien à notre problématique : il modélise une large gamme de phénomènes cognitifs et langagiers. Nous croyons en lui pour plusieurs raisons :

- Il est rattaché à la démarche académique : dans la partie 1, chaque simulation est formalisé mathématiquement sur la base de l'automate fini, dans la partie 2 chaque simulation est centré sur l'agent situé, un outil basique de l'informatique distribuée.

- Il est applicable à beaucoup de disciplines car il embrasse un large secteur. D'un côté, il touche les sciences humaines et sociales ; de l'autre, les mathématiques et l'informatique.

- Il est simple pour trois raisons : D'abord, à l'instar des langages objets, nous décentralisons le code et les données. Ensuite, nous le construisons en guidant l'émergence de ses agents, ce qui induit des simplifications énergiques dans leurs structures. Enfin, nous plaçons la barre très bas : le but de chaque agent modélisé est ni d'être parfait, ni d'être optimal, il est seulement de survivre.

- Il est facile à mettre en oeuvre car son code, très léger, est écrit en C, et nous avons fait un effort de rédaction pour le vulgariser.

8) Conclusion : un travail de recherche fondamentale

a) Un travail qui s'inscrit dans le long terme

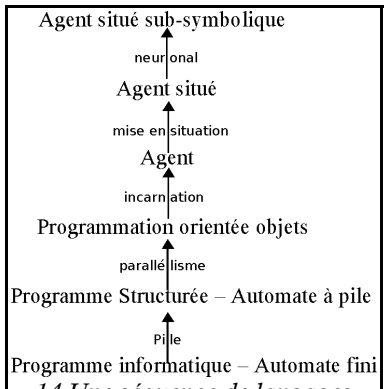
Ce travail de recherche fondamentale s'inscrit difficilement dans la gestion à court terme des entreprises. Son but n'est pas de participer à un projet industriel afin de répondre au cahier des charges d'une société. Il n'est pas de résoudre un problème ponctuel qui se pose occasionnellement dans un service informatique. Il est donc plutôt tourné vers des perspectives à long terme. Il se positionne un peu comme les chercheurs du CNRS qui pouvaient autrefois s'investir à longue échéance, sans devoir rendre des comptes dans l'immédiat.

b) Dégager les structures et mécanismes simples dans les langages de programmation à agents

Ce travail de recherche fondamentale s'attache à dégager les structures et mécanismes simples de l'informatique.

Selon le point de vue de l'histoire nous analysons la séquence des langages de programmation : *machine, assembleur, macroscopique, structuré...* Et nous focalisons seulement sur les quatre étapes: *structuré, à objets, à agents, à agents situés*.

Nous regardons plus particulièrement, la dernière étape *agent→agents situés*, qui met en situation des agents dans un monde probatoire concret. Nous analysons cette projection des agents abstraits vers une application concrète comme une instanciation et une matérialisation.



14 Une séquence de langages de programmation

Finalement, trois raisons simplifient le traitement qu'ils effectuent sur l'information : l'effet holistique du groupe, la simplicité de relation au support digital dû à l'effet d'horizon, la mise en situation de survie (dans le cadre de la VA, nous leurs fournissons de l'énergie et nous leur demandons juste de survivre).

Ainsi nous en tirons deux avantages. D'un côté nous simulons des phénomènes sociaux intéressants. De l'autre, puisque leur représentation et traitement de la connaissance deviennent très simples, alors leur structure interne se simplifie aussi et par là même devient plus visible.

9) Perspectives

Communication.

Nous souhaitons communiquer, publier, afin de faire connaître notre travail, et d'y intéresser les chercheurs. Ceci consiste à améliorer la qualité des explications techniques, ajouter quelques simulations plus spectaculaires, et présenter des exposés pour le faire connaître.

Technique informatique.

Sur un plan informatique, nous allons prolonger notre travail, le porter vers les objets et développer de nouveaux exemples.

D'abord, il est important pour nous de le réécrire en langage objet[3] afin de mieux faire apparaître sa structure basée sur l'héritage.

Ensuite, nous devons écrire encore d'autres exemples attractifs qui viendront se placer en position de feuilles, au bout des branches de notre taxinomie. Par exemple, nous pensons simuler des rapports socio-économiques de troc, ou continuer en direction d'un simulateur mémétique.

3 Pour ce faire nous pensons utiliser C++, Java, ou même ajouter une couche en C.

a) Mots clés

agent intentionnel,
agents rationnels,
agent réactif,
agent situé,
agents sociaux,
automate fini,
causalité,
cognition,
coopération,
croyances
déterminisme,
désirs,
émergence,
empirisme,
enseignement,
évolution,
grille,
héritage,
informatique,
intentions,
intelligence artificielle,
jeu digital,
langage objets,
linguistique,
obstacles,
phylogenèse,
plate-forme,
progression,
programmation,
rationalité,
séquenceur,
simulation,
système multi-agents,
taxinomie,
vie artificielle.

b) Key words

artificial intelligence,
artificial life,
belief,
causality,
cognition,
computer science,
cooperation,
desire,
determinism,
digital game,
emergence,
empiricism,
evolution,
finite automaton,
grid,
intentional agent,
inheriting,
intentions,
linguistic,
multi-agents system,
object language,
obstacles,
phylogenesis,
progression,
programming,
rational agent,
rationality,
reactive agent,
sequencer,
simulation,
situated agent,
social agent,
taxonomy,
teaching,
work-bench.

c) Cette thèse à été préparée au :

LIASD - Laboratoire d'Informatique Avancée de Saint-Denis
Département Informatique de l'Université Paris 8
U.F.R. 6
Ecole Doctorale Sciences Sociales
Université Paris 8 Vincennes→Saint-Denis
2 rue de la Liberté
93526 SAINT-DENIS cedex

Tél : 01 49 40 67 89

http://www.univ-paris8.fr

Ch. 2 Remerciements

Au cours de ce travail de recherche, au sein du laboratoire, j'ai reçu de la part des membres des avis, conseils, commentaires, remarques, aides, qui mis bout à bout m'ont aidé à effectuer cette thèse. Je les remercie donc.

De façon plus globale je tiens aussi à remercier ceux qui m'ont aidé d'une façon plus générale. J'ai choisi de le faire sur une forme qui se veut différente de la méthode classique. Finalement elle se rapproche de celle de la dédicace. Bon, de toutes façons, qu'importe la forme, pourvu que le coeur y soit.

1) Jury

Herman Akdag - Je le remercie d'avoir accepté de participer à mon jury, de sa disponibilité et de ses précisions sur la déduction dans les systèmes mathématiques basés sur une axiomatique.

Gille Bernard m'a guidé tout au long de ces 1+6 années, avec rondeur et délicatesse ; il m'a transmis certaines clés qui, à touches pointillistes, ont orienté mon travail. Je retiens particulièrement ses remarques sur les états internes des êtres, et sur l'action vue comme un procès finaliste.

Tristan Cazenave, aimable, si discret qu'on pourrait faire l'erreur de ne pas comprendre l'intérêt de son travail. Tristan, j'ai déjà eu l'occasion de te le dire : le fait que tu acceptes d'être à mon jury a été un déclencheur et m'a permis de rentrer dans une dynamique gagnante.

Jacques Ferber. Je l'ai connu à la fin des années 70 en lisant Micro-Systèmes. Je suis honoré qu'il ait accepté de participer à mon jury. Son livre sur les SMA est immense, quand je le feuilletais, il me filait le cafard, je me sentais comme un enfant face à la mer.

Patrick Greussay : après le premier exposé à propos de mon travail, qui n'était que projets et suppositions, il m'a dit : vas-y. C'est ce qui reste de l'esprit de Paris8-Vincennes. Je me demande si ailleurs on m'aurait laissé tant de liberté dans ma recherche.

Harald Wertz - Il m'a beaucoup aidé, par ses questions, ses remarques. Harald, merci d'avoir pris le temps de lire, de corriger et de commenter certaines de mes pages, même si parfois je t'énervais par mes raccourcis trop rapides.

2) La faculté Paris 8 – Vincennes → St-Denis

Brahim Achili, toujours aimable et souriant : pour lui tout semble facile. Et puis un jour j'ai bien compris qu'il devait s'accrocher lui aussi.

Arab Ali Cherif, m'a constamment encouragé, avec un esprit positif. Dans les moments de doute, ça me rassurait de voir qu'il comptait sur moi.

Hania Aoudia - Très discrète, mais active et qui s'accroche.

Archil, un petit jeune positif et brillant, qui monte, qui monte.

Annaïs Atencia, l'impresionisme et ses tableaux pointillistes : un exemple de constance.

Françoise Balma - Toujours aimable et accueillante ; discrète, sérieuse, exigeante, elle inspire mon respect.

Rim Chabane, nous avons fait toute la fin du parcours en nous soutenant mutuellement.

Jean-Jacques Bourdin m'a bien guidé quand j'étais TPeïste et que j'encadrais les étudiants de son cours de C. Nous avons fait du bon boulot.

Farès Belhadj : Farès tu es perfectionniste, c'est agréable de traiter avec toi, mais STP, maintenant prends soin de ta santé.

Vincent Boyer : toujours aimable et positif. Bravo pour ton séminaire où j'ai écouté des conférenciers bien intéressants.

Jim Feat : un professionnel exigeant, mais ce ne sont pas les étudiants qui vont s'en plaindre.

Brigitte Groleas. J'ai bien aimé apprendre le Java avec elle, puis l'enseigner au Greta de la Défense.

Hind : partie depuis quelques année déjà.

Jean Méhat : En DEA, j'ai bien aimé son cours sur la théorie des jeux. J'aime bien son travail sur ARI. Jean : un passionné à l'état pur !

Marie-So - Merci pour son accueil. Elle m'a bien soutenu quand j'avais des problèmes familiaux. Je me souviens des discussions au RU, qui m'ont bien remonté le moral.

Jean-Jacques Mariage : il m'a bien aidé pour ma première publication.

Karine Mordal : brillante, mais en DEA tu as eu des notes meilleures que les miennes : ne recommence jamais ça !

Nicolas et Youcef - Je leur dois le concept de "génération de fonctions par emboîtement descendant de sous-parties".

Anna Papa, qui m'a accueilli si chaleureusement dans le groupe CSAR.

Jacqueline Signorini, qui m'a aimablement mis en garde : c'est très ambitieux tout ça. En effet, il est important de savoir dire aimablement des critiques positives.

Philippe Pinon - Il m'a recruté pour faire les TP alors que je connaissais très peu Unix. Ce travail m'a permis de progresser avec Linux, et de mettre du beurre dans les épinards.

Raoul Kuczyk, administrateur du bocal - Nous sommes plusieurs à penser que l'excellence des étudiants de notre UFR6 en informatique tient aussi au bocal, à ce fait que 24h/24, 7j/7 et 365j/an, ils puissent disposer de ce libre service, si difficile à maintenir. Bravo !

Lynda Seddiki - Positive, elle a toujours un commentaire pour faire avancer le schmilblick.

Dom Sobczyk, il cache son inquiétude derrière un air débonnaire, et un coeur gros comme ça.

Henoc Soude - Il a toujours la pêche, c'est communicatif.

Isis Truc, qui m'a aimablement mis en garde : c'est un peu philosophique tout ça. J'en ai tenu compte et j'ai cherché à formaliser mon travail avec plus de rigueur. Et j'ai eu cette chance de voir comment relier l'automate fini à l'agent situé.

Moïse Valvasori m'a beaucoup aidé dans mon travail de recherche, par exemple, il m'a montré au moins trois choses importantes : les Mind Maps, le travail de Braitenberg, les L-Formes.

Yan Zhi - J'aime bien son sujet de thèse. Bon courage à lui.

Jacques Pitrat (à Paris 6) - Je suis fasciné par son travail magnifique : son chercheur en IA. Je le remercie pour ces longues discussions où il m'a beaucoup enseigné.

3) Société Francophone de Mémétique

Bertand Biss, pour lui tout semble facile, surtout cuisiner les gâteaux.

J'ai bien aimé les contributions de Jean-Pierre Crespin, un libre penseur, un homme libre.

Pascal Jouxtel, il m'a bien épaté quand il m'a expliqué le travail qu'il avait fait sur les verbes d'action du français.

Charles Mougel, il faut que tu continues tes études pour faire avancer les simulateurs mémétiques. Accroche-toi !

Bernard le Suisse, un peu gourou, mais profondément humaniste.

Éric de Rochefort, positif et optimiste, comme sa belle moustache !

4) Enseignement Publique Informatique (EPI)

Jacques Béziat, qui a aimé mon interprète Logo. Je le remercie de m'avoir fait connaître l'EPI.

Le plaisir de travailler avec Alain Kapur, si délicat, si aimable.

Jean-Bernard Viaud, à l'humour corrosif, qui a bien voulu diffuser mon travail. Tous les livres sur l'informatique qu'il m'a repassés me sont bien utiles.

5) L'association Philotechnique

Ric-Martial Albertini : compagnon de galère, félicitation pour ta réussite, voici ta persévérance et ton talent récompensés.

Jean-Marc Capellero, les randonnées dans le parc des Buttes-Chaumont : nous refaisions le monde en parlant Linux et économie.

Robert Chassinat qui m'a insufflé un peu de son énergique constance.

Pierre Etchégaray m'a aidé à comprendre la philosophie des langages objets.

Claude Margue m'a fait confiance au départ et m'a donné ma chance.

Phil Maverick : un exemple de débrouillardise. Par ailleurs, il m'a enseigné comment assembler une unité centrale.

Jean-Michel Meulien m'a clairement résumé les nombreux aspects de la programmation objets.

Arlette Ovaere, l'histoire d'une auditrice devenu animatrice. Elle m'a bien secondé dans mon travail.

Dominique Philippot a montré beaucoup d'intérêt pour les plate-formes pédagogiques afin d' enseigner le langage C et la logique.

Hélène Quenel : Musique, chant, théatre, philosophie et scouter : très polyvalente cette personne.

Bernard Sabrou, qui m'a montré la route et fait partager son intérêt pour la linguistique (Sapir).

Béatrice Thiébaut, quand j'intervenais au siège, et que j'avais des problèmes familiaux, elle m'a bien soutenu.

François Thiébaut - De plus en plus présent, il prend de l'assurance et fait son chemin.

Bruno Tredez m'a fait suivre l'évolution des ordinateurs, et m'a aidé à passer du développement sous DOS à celui sous Windows.

Claire Tonello : en fait je pense que, artiste et chercheur, sont deux démarches qui sont finalement proches car ils partagent le même but : la quête du graal.

6) L'informatique à Paris pour Les Jeunes

Alain Asseum : un exemple d'esprit positif.

Jean-Luc Ben Gui Gui - L'accent rocailleux et chantant qui nous envoie des bouffées d'air du Sud-Ouest. Le souvenir d'une épopée ensemble, le *passeport pour l'informatique*. Quand nous avions tant d'énergie que nous ne doutions de rien.

Cecil Ben Simon, toujours bienveillant.

Avec Philippe E, Laure P, Antoine S, Ghislaine M, OSC, Aziz, DAR, Mzé, Azma K, Claire S, nous avons, avec succès, continué la lutte après notre licenciement.

Elie et ses parents qui m'ont soutenu durant ce long périple.

Antoine Gharbi m'a montré comment appliquer concrètement les graphes pour réaliser un jeu de l'oie.

Avec Lydia Gouevic nous avons discuté couches et biberons.

Gilles Junquet et Antoine G. m'ont déchargé de quelques heures pour que je développe des outils pédagogiques.

Lucy Kang Ethe m'a bien aidé pour l'anglais.

José Manrique, m'a encouragé en me disant qu'il ne doutait pas de ma réussite dans cette entreprise.

Daniel-Alain Régent m'a rapidement encouragé quand je me lançais sur le simulateur de paysages.

Martin Saldariaga m'a aidé à maîtriser le compilateur C de Borland.

Claire Salmon, Daniel-Alain Régent, Philippe Epaud, Eric Bellot, Françoise ont été pour moi des exemples de pédagogues.

Olivier Schmidt-Chevalier, Papy Logo, m'a beaucoup aidé pour réaliser un interprète Logo.

7) La section Hand-Ball au Paris Sport Club

Ghislain et le petit Sexe, deux commerciaux qui nous ont donné une leçon de relations humaines.

Alain Gombard : la poignée de main énergique d'une personne positive.

Avec Thierry Lavoux nous avons fait un bon bout de chemin ensemble.

Jean-Jacques Mahn : ensemble nous avons pris nos responsabilités.

Olivier Trémois m'a donné envie avec son travail et son théorème sur le traitement du signal.

8) Famille

Fleur - M'enfin, c'est malheureux d'être bonne en maths et de faire de la littérature.

Tom : c'est bien, continue de t'accrocher ainsi.

Axel, Corentin, Gaëtan, Odran, rassurez-vous, pour vous les études seront certainement moins longues.

Laurence : c'est bien pour elle et les enfants qu'elle vienne s'occuper d'eux pendant que j'enseigne à la Philotechnique.

Marie : Bon, ça a été long, mais tu vois c'est presque fini !

Francis : à discuter à gauche à droite avec toi de d'autres utilisateurs de l'informatique, j'ai beaucoup appris.

Eric : Cool et discret. Sage et généreux, il dit ce qu'il pense et il fait ce qu'il dit.

Véro : la maman courage. Soeurette merci pour toute ton aide.

La famille Joseph Moulet : Les escapades avec Joël, Frigé-Crème avec Jean-Paul et Génie, la sculpture sur plomb avec Gilbert, les discussions avec Jean-Luc, Josiane. Odette et Claude, son bonhomme courageux.

Merci à mes parents, de m'avoir tant aidé. J'ai certainement fait une partie de ce travail pour eux.

Ch. 3 Table des matières

Préliminaires

Ch. 1 Résumé synthétique de cette thèse....5
Simulations d'agents sociaux situés, en survie, dans un monde digital....5
1) Cadre mathématico-informatique : 5
2) Problématique : livrer à l'utilisateur un outil : une plate-forme....6
2.A) Fournir un outil pratique....6
2.B) Notre cible : l'informatique distribuée et les sciences humaines et sociales....7
3) Modéliser les agents classés dans cette taxinomie....7
4) État de l'art : 9
4.A) Situer ce travail selon un point de vue mathématique....9
4.B) Situer ce travail selon le point de vue de l'intelligence artificielle (IA)....9
5) Méthodes : construire un modèle de traitement cognitif au moyen d'A6S....11
5.A) Problèmes de représentation des connaissances induits par notre approche....11
5.B) Méthode : une recherche ouverte, une démarche éclectique 12
5.C) Notre démarche : ratisser large pour sélectionner....13
5.D) Démarche éclectique : sans vergogne, nous empruntons tout azimut....13
5.E) Aspect holistique du fonctionnement soma+psyché d'un agent....14
5.F) Concept utilisés : nécessité d'un éclectisme....15
6) Résultats : une plate-forme de simulation, fournie avec des exemples....16
6.A) Comme résultats, nous fournissons des exemples de simulations....16
6.B) Comme résultats, nous fournissons des outils pédagogiques....17
6.C) Comme résultats, nous modélisons des phénomènes de communication....17
7) Évaluation des résultats : 19
8) Conclusion : un travail de recherche fondamentale....19
9) Perspectives....20
Ch. 2 Remerciements....24
1) Jury....24
2) La faculté Paris 8 – Vincennes→St-Denis....24
3) Société Francophone de Mémétique....25
4) Enseignement Publique Informatique (EPI)....25
5) L'association Philotechnique....25
6) L'informatique à Paris pour Les Jeunes....26
7) La section Hand-Ball au Paris Sport Club....26
8) Famille....27
Ch. 3 Table des matières....28

Problématique, histoire et état de l'art, concepts, méthodes

Ch. 4 Notre problématique....37
1) Travail théorique et pratique sur la simulation....37
2) Nous proposons une gamme de simulations, destinée à la recherche....37
2.A) Notre gamme de simulations est destinée à la recherche en informatique....37
2.B) Cette gamme de simulations constitue un outil pédagogique....38
3) Proposer une taxinomie, un classement arborescent de simulations....39
3.A) Deux critères pour classer les simulations fournies 39
3.B) Notre but est de proposer une progression de simulations adaptées....40
3.C) Quelle méthode guide le choix des branches qui sont ajoutées à l'arbre....41
Ch. 5 Histoire du traitement de la complexité en IA....42
1) Au début : des chercheurs peu conscients du problème de la complexité....42
1.A) L'enthousiasme des pionniers est battu en brèche par la réalité des faits....42
1.B) Le subterfuge de Éliza de Weizenbaum met à mal le test de Turing....42
1.C) Les praticiens des sciences humaines modèrent l'ardeur des informaticiens....43
2) Histoire de l'art, le structuralisme : tout rattacher à une structure typée....47
2.A) Introduction : tout rattacher à une structure [Jak 1981]....47
2.B) Définissons les notions de structure et de structuralisme....47
2.C) La logique : une représentation structurée et statique du système/monde 47
2.D) Applications des structures et du structuralisme....48
2.E) L'impasse de la recherche de la grammaire d'une langue....48
2.F) Le classement pour obtenir une taxinomie de structures ?....49
3) Histoire : Le fonctionnalisme : une représentation dynamique de l'action....50
3.A) Définir le fonctionnalisme en philosophie de l'esprit (computationnalisme)....50
3.B) Modèle et implémentation du fonctionnalisme : l'automate à états finis....50
3.C) Représentation au moyen de la logique : passer du statique au dynamique....50
3.D) RC dynamique : la solution des systèmes formels 51
3.E) Les simulateurs, maintenant nous allons focaliser sur les simulateurs....51
Ch. 6 Synoptique général de différents simulateurs....52

1) Fonctionnement des simulateurs simplifiés ou dissymétriques....52
1.A) Les simulateurs basés uniquement sur le choix de l'action....52
1.B) Les simulateurs basés uniquement sur la réponse du support....53
2) Les simulateurs de jeux digitaux : ils sont partiellement dissymétriques....54
3) Simulateurs mixtes/équilibrés munis d'une représentation symbolique....54
3.A) Introduction : travailler avec de simulateurs mixtes/équilibrés....54
3.B) Deux domaines qui peinent à faire fonctionner des simulateurs équilibrés....54
3.C) Conclusion : d'autres techniques pour réaliser des simulateurs équilibrés....55
Ch. 7 Six simulations, du programme à l'agent situé sub-symbolique....56
1) La simulation au moyen des langages de programmation classique....56
2) Simulation au moyen des langages à objets (LO) [Leb 1991], [SB 2006]....57
2.A) Les LO sont basés sur deux idées centrales : décentralisation et héritage....57
2.B) La PO apparaît comme une solution au qualification problem....58
2.C) Déconstruire les objets vers le substrat....58
3) Système multi-agents (SMA) : une simulation au moyen des agents....59
3.A) Introduction classique des agents....59
3.B) Incarnation, mise en situation dans le monde....59
3.C) Introduction des agents, tournée vers les phénomènes langagiers, cognitifs....60
4) Simuler, modéliser l'action de l'agent situé, qui change l'ordre du monde....61
4.A) Le concept d'agent situé [Kae 2005]....61
4.B) Détailler la formalisation des AS : 2 mots clés dérivant de la localité....61
4.C) Les outils pour moduler localement un raisonnement....62
5) Simuler au moyen d'agents munis d'un traitement sub-symbolique....64
5.A) Au fil de l'histoire, une opposition symbolique vs neuronal....64
5.B) La description globale de l'état du monde....65
5.C) Modéliser le traitement causal de l'AS, en représentation sub-symbolique....66
5.D) Raisonnement : l'agent rationnel se représente un monde sub-symbolique....67
5.E) Graphe pour implémenter la réponse du support dans ce contexte local....67
6) Simuler au moyen de robots placés dans une pièce....68
7) Quel positionnement pour notre plate-forme ?....68
Ch. 8 État de l'art : introduction de la communication....69
1) Quelles sont les raisons de cette introduction ?....69
2) Trois exemples illustrant cette communication par signe ou symbole....70
Ch. 9 La transmission culturelle au moyen du langage....71
1) Nous choisissons de séparer cognition partagée et transmission culturelle....71
2) Étape 1 : traiter une langue dont la grammaire est figée à un temps t....71
2.A) Analyse syntaxique de la transmission :....71
2.B) Analyse syntaxique + interprétation sémantique :....71
2.C) Traitement de la pragmatique : scénarios et protocoles....72
3) Étape 2 de l'état de l'art : traitement dynamique d'une langue vivante....72
3.A) Quelques précisions initiales....72
3.B) Apprentissage en introduisant une sémantique provenant de l'extérieur....72
3.C) Apprendre avec une sémantique interne : la sanction de l'efficacité....74
4) État de l'art étape 3, le langage transmet des règles épigénétiques....76
4.A) Introduction : modéliser/simuler la transmission mémétique....76
4.B) Simuler avec lissage statistique : le mème est autonome mais vide de sens....76
4.C) Simulateur de l'IA symbolique : forger une identité aux mèmes....77
4.D) Article de Nicolas Journet Les espoirs déçus de la mémétique [Jou 2007]....77
4.E) Interaction et construction en couches donne une épaisseur à l'agent....78
Ch. 10 Les difficultés induites par notre problématique....79
1) Problèmes que la logique éprouve pour représenter les scènes dynamiques....79
2) Problème de la complexité à modéliser la réponse du monde....82
2.A) Introduction, position du problème....82
2.B) En écrivant nos modélisations, nous buttons contre le mur de la complexité....83
2.C) Exemple de cette complexité à modéliser la réponse du monde....83
2.D) Transition vers la suite :....84
Ch. 11 Les concepts utilisés dans ce travail....86
1) Situer, positionner les concepts d'émergence et d'évolution....86
1.A) Introduction....86
1.B) Le désordre....86
1.C) Le chaos....86
1.D) L'agent situé : un comportement causal avec un horizon....87
1.E) L'omniscience....87
1.F) Introduction des concepts d'émergence, d'évolution et de co-évolution....87
2) La sémantique en simulation....89
2.A) Dans les simulateurs avec lissage statistique les mèmes sont dégénérés....89
2.B) Fourniture d'une plate-forme qui ne soit pas dégénérée....89
2.C) Partir d'une base qui possède déjà sa sémantique : celle de la VA....89
2.D) La démarche de construction que nous utilisons....89
2.E) Conclusion....90
Ch. 12 Le concept de la vie artificielle pour une sémantique de l'action....91
1) Introduire la vie artificielle pour obtenir une sémantique de l'action....91
2) État de l'art : Les trois mouvances de la VA (ALife en anglais)....91
2.A) La VA chimique (Wet ALife)....91
2.B) La VA matérielle (Hard ALife)....91
2.C) La VA du logiciel (Soft ALife)....91

3) Le concept de VA pour implémenter une sémantique de l'efficacité....92
3.A) La nourriture, une première évaluation (directe) de l'agent....92
3.B) Le niveau d'énergie, une seconde évaluation (indirecte) de l'agent....93
3.C) L'évaluation du niveau d'énergie donne une sémantique indirecte à l'agent....93
4) Donner un sens à la vie, donner une vie au sens....93
4.A) Exemple d'une machine intelligente insensée....93
4.B) La VA pour donner un sens aux simulations....94
4.C) Des simulations qui n'ont pas de sens, la VA assure un service minimal....94
5) Les conclusions à tirer sur la structure des simulateurs sensés....94
5.A) Mesurer le niveau d'énergie des agents, l'outil pour évaluer un simulateur....94
Ch. 13 Les concepts de base de notre travail....96
1) Introduction : présenter et détailler les concepts de base de notre travail....96
2) Le finalisme : un concept nécessaire pour parler d'action sur le monde....96
2.A) Définir le concept d'action selon le point de vue de la philosophie de l'action....96
2.B) Rappels des conventions avancées précédemment :....96
2.C) Selon la philosophie, le procès sous-tendu par un but devient une action....96
3) Nouveau concept de sens : évaluer le sens d'une fonctionnalité au sein d'un être....97
3.A) Introduire le concept : la sémantique vue comme une fonction de transfert....97
3.B) Appliquer le concept : la sémantique vue comme une fonction de transfert....97
3.C) Conclusion : une sémantique qui possède du sens commun....98
4) Des concepts rattachés à la vie ou l'évolution artificielle....98
4.A) Étape 1 de l'émergence du groupe....99
4.B) Étape 2 de l'évolution de la structure....99
4.C) Étape 3 : concept de la co-évolution....101
5) Du concepts de symbole à celui de sub-symbole....104
5.A) Signe et forme : deux aspects des symboles....104
5.B) Construction des symboles....105
5.C) Définition de la notion de symbole par le dictionnaire (Lalande)....105
6) Symbole de l'informatique formelle vs sub-symbole du connexionisme....105
6.A) Symbole vs sub-symbole à propos de codage, de propagation et traitement....105
Ch. 14 Méthodes utilisées afin d'atteindre nos buts....108
1) Présenter les méthodes retenues....108
2) Rappel rapide de notre problématique....108
2.A) Fournir une plate-forme....108
2.B) Fournir une classification des simulations qui tournent sur la plate-forme....108
3) Le concept de zone privilégiée où de masse critique invite à la simplicité....108
3.A) Définitions....109
3.B) Illustrer cette notion de zone privilégiée en termes de surface....109
3.C) KISS : plus ses simulations sont simples, denses, plus cette zone est large....109
3.D) Conclusion : il existe une zone de fonctionnement privilégiée....110
4) Les méthodes que nous avons retenues, au vu de l'état de l'art....110
4.A) Préciser notre positionnement par rapport à la vie artificielle (VA)....110
5) Les méthodes qui ont guidé la construction de notre taxinomie....111
5.A) Le choix d'un parcours en largeur d'abord....111
5.B) D'un côté nous guidons l'émergence d'une progression (en largeur d'abord)....111
5.C) De l'autre côté, la bionique guide notre classification de simulations....111
6) Modéliser le procès cognitif de l'agent : structure d'arbre ou de réseau ?....114
6.A) Intro : la fonction de traitement cognitif de l'agent est difficile à modéliser....114
6.B) Quelle est l'intérêt d'une représentation arborescente des connaissances ?....114
6.C) Épistémologie de l'approche informatique : du synthétique à l'analytique....114
6.D) Comment traiter de façon arborescente le graphe d'un procès cognitif ?....115

Partie théorique

Ch. 15 Simplifier l'automate fini (AF) pour obtenir un séquenceur....118
1) Pour formaliser l'agent situé, baser notre construction sur l'AF....118
2) Rappels à propos de l'AF (automate à états finis)....118
2.A) Description mathématique de l'AF....118
2.B) Implémentation de l'AF....118
2.C) Ainsi la lecture du ruban ouvre l'automate sur l'extérieur....119
2.D) La structure du graphe de transition....119
3) Présentation de notre feuille de route pour les chapitres à venir....120
4) La restriction de l'AF, à un séquenceur....120
4.A) Description mathématique du séquenceur....121
4.B) Ci-contre nous présentons le schéma de l'implémentation d'un séquenceur....121
4.C) Le séquenceur est fermé au monde : il est solipsiste....121
5) Transition : vers la matérialisation....121
Ch. 16 Vie artificielle, la cellule végétative exécute une séquence digitale....122
1) Nous introduisons une vie artificielle (VA) dans notre modèle(colonne 4)....122
1.A) Analysons l'architecture de la CV(+MS) qui exécute une séquence digitale....123
1.B) Traitement des états somatiques, au sein de la CV(+MS)....124
1.C) Quelles sont les conséquences de cette introduction ?....125
2) Quelques illustrations bioniques de l'introduction de l'énergie....126
2.A) La cellule végétative (CV)....126
3) Le cycle graine-cellule (GC)....127

3.A) Description, présentation générale....127
3.B) L'extérieur de la GC :....127
3.C) L'intérieur de la machine : comment elle est implémentée....127
4) La machine séquenceuse pulsionnelle (MSP)....128
4.A) Description, présentation générale....128
4.B) L'extérieur de la MSP :....128
4.C) L'intérieur de la machine :....128
4.D) Conclusion : quel est l'intérêt de la MSP....128
5) Transition :....129
Ch. 17 Matérialiser le séquenceur en une machine séquenceuse....130
1) Introduction....130
2) Transformer le séquenceur en machine séquenceuse, en le matérialisant....130
2.A) Nous partons du séquenceur....130
2.B) La transformation ϕ : A$-\phi\rightarrow$ MS matérialise le séquenceur A en une MS....131
2.C) La transformation de chaque composante de la transition séquenceur→MS....132
3) Transition d'état du séquenceur → procès de la machine dans le monde....133
3.A) Le système formel (SF) devient une machine séquenceuse (MS)....133
3.B) La transition d'état du séquenceur....134
3.C) Comment est effectué le procès physique qui change la position du pion ?....134
4) Conclusions....134
5) Quelques illustrations de l'application des machines séquenceuses....135
5.A) Le clignotant double....135
5.B) La machine séquenceuse (MS)....135
6) Conclusion....136
7) Perspective, col 4 : mettre en place un fonctionnement autonome interne....137
7.A) Le passage colonne 2→colonne 3 constitue donc une étape intermédiaire....137
7.B) Colonne 3→colonne 4 introduire la vie artificielle et la reproduction....137
Ch. 18 L'agent situé minimal (ASM), son actionneur et son action....138
1) Introduction....138
2) Col 5 introduit le pilotage de l'agent autonome qui remplace le SF initial....138
3) Transition vers l'action de l'agent situé dans le monde....139
3.A) Introduire un mouvement piloté par un étage de décision....139
3.B) Le but des chapitres suivants est d'introduire progressivement l'agent situé....139
3.C) L'agent situé décrit comme l'agent A situé sur un support S....139
3.D) L'agent A6S : Agent Somatique Situé en Survie Sur un Support Simple....139
3.E) Généralisation : le fonctionnement causal décision action d'un agent situé....141
3.F) Application : implémentation concrète de l'architecture interne de l'ASM....142
3.G) Application : les fonctions de perception et de décision chez l'ASM....142
3.H) Application - L'actionneur est activé pour effectuer un procès générique....142
3.I) Le premier actionneur à paraître est celui de mouvement :....143
4) L'assimilateur : i.e. une introduction concrète d'actionneurs et d'action....144
4.A) Rappel sur l'action....144
4.B) Un exemple simple d'assimilateur : le thon (prisonnier dans un filet)....145
4.C) Le trajet du requin s'en va à l'infini....145
5) Une double transition....146
Ch. 19 Représenter formellement l'agent situé (A6S)....147
1) Tableau synthétique pour présenter notre démarche....147
2) Nous travaillons à formaliser une partie du domaine des agents situés : 147
3) Colonne 5 : Description formelle de l'Agent Situé (A6S)....147
3.A) Rappel sur l'agent situé (A6S)....147
3.B) Rappel sur un exemple d'agent situé : l'agent situé minimal (ASM) 147
4) Continuer de formaliser le monde de l'agent situé sur son support....148
4.A) Notre but : continuer de formaliser ce monde régulier....148
4.B) Rappel : la description formelle de l'agent situé....148
4.C) Rappel : le monde M est un quadruplet {S, A, Q, q0} :....148
Ch. 20 Formaliser le monde de l'agent situé : un agent sur un support....150
1) Représenter qt l'état du monde M au temps t 150
2) Comment réduire la description du monde à celle d'un agent sur sa case 150
3) Représenter les relations entre les cases du support....150
3.A) Plongeon dans la réalité physique....150
3.B) Représentation statique du système : méthode extensive 151
3.C) Représentation statique du système agent-support : méthode compréhensive....151
4) Représentation séquentielle du procès de l'agent situé....152
4.A) Introduction générale, globale : le procès consiste en une transition d'état....152
5) Outils pour décrire le fonctionnement axiomatique du procès résultant....153
5.A) Maintenant décrivons le procès de l'agent situé sur un support....153
5.B) Sur plusieurs aspects, le monde de l'agent situé est très régulier....153
5.C) Les règles avec variables décrivent les procès (transitions d'état des actions)....154
6) Définissons, décrivons la prise de décision qui pilote le procès....155
6.A) Traitement de la partie (ψ : qi→act°i)....155
6.B) Traitement de la partie : RéponseDeM(qi, act°i);....155
6.C) Conclusion à propos de la prise de décision de l'action dans l'agent situé....156
Ch. 21 Agents situés : le parallélisme induit les capteurs et la causalité....157
1) Tableau synthétique pour présenter notre démarche....157
2) Nous travaillons à formaliser une partie du domaine des agents situés : 157
3) Colonne 6 : introduire le parallélisme induit les capteurs et la causalité....157

4) Le paradis perdu ou la transition vers les herbivores et les objets....158
5) Mettre en oeuvre deux agents identiques en parallèle....158
5.A) Pour modéliser, nous partons de la définition initiale de l'agent situé :....159
5.B) Et nous plaçons deux agents situés en parallèle sur le même support....159
5.C) Maintenant, explicitons cette notation....159
6) Mettre en oeuvre deux agents différents en parallèle....159
6.A) Pour modéliser, nous partons de la définition initiale de l'agent situé :....160
6.B) Et nous plaçons deux agents différents en parallèle sur le même support....160
6.C) Maintenant, explicitons cette notation....160
7) Mettre en oeuvre N agents en parallèle....160
7.A) Maintenant, explicitons cette notation....160
7.B) La gestion en parallèle de l'action des agents....161
7.C) Introduction des agents passifs....162
8) Résultat : pour deux raisons l'agent situé est non-déterministe....162
8.A) Énoncer les deux raisons du non déterminisme....162
8.B) Conclusion....163
Ch. 22 Aspect théorique : formaliser les Agents réactifs tropiques....164
1) Rappel - construction progressive : de l'automate fini vers l'agent CDI....164
1.A) Nous avons déjà traité ce tableau 1 : de l'automate fini (AF) à l'agent situé....164
1.B) Tab 2, le parallélisme, de l'agent situé(A6S) jusqu'à l'agent BDI rationnel....164
1.C) Présentation mathématique du parallélisme....165
1.D) Le monde M est un quadruplet {S, Σ, Q, q0} :....165
2) Décrivons formellement les agents réactifs tropiques....166
2.A) Introduction : dans la simulation nous faisons apparaître la causalité....166
2.B) Le fonctionnement réactif causal perception décision action....166
2.C) Précisons, définissons, décrivons la prise de décision qui pilote le procès....167
2.D) Préciser la réponse du monde dans le synoptique de l'agent....169
3) Le problème épistémologique du procès et de l'action....169
3.A) Introduire, poser le problème : procès, but, action d'un agent situé réactif....169
3.B) Un agent réactif est indirectement finaliste....170
3.C) Conclusion : le procès devient action....172
4) Bilan - L'agent réactif tropique (agent tropique)....172
4.A) Nous disposons d'un agent réactif, i.e. au comportement spontané....172
4.B) Définition, présentation de l'agent réactif....172
4.C) L'agent réactif tropique s'inscrit dans l'instant, i.e. hors du temps....173
5) Bilan et conclusion :....174
5.A) L'agent situé minimal : base d'une reconstruction....174
5.B) L'agent réactif tropique....174
5.C) À terme : construire en couches progressives vers les agents rationnels....174
Ch. 23 Volet pratique : aspects tropiques de l'agent réactif tropique....176
1) Rappel : notre base de départ....176
2) L'agent réactif tropique sur un espace 1D introduit une causalité discrète....177
2.A) Introduction du parallélisme....177
2.B) Les objets que nous introduisons sur les lieux bloquent l'agent....177
2.C) Rappel : limite de profondeur des interactions....178
2.D) Introduction du sens : l'agent repart dans l'autre sens....178
2.E) Introduction de la finalité....178
2.F) Intervention du concepteur : introduction de règles de causalité....178
2.G) Conclusion : du procès à l'action....178
3) Agents réactifs tropiques dans un espace 2D : une démarche causale riche....179
3.A) La finalité demeure mais le concepteur introduit des règles plus complexes....179
3.B) Introduction d'une grande richesse....179
3.C) Introduction d'états externes : 179
3.D) Introduire l'influence à distance....180
3.E) L'agent tropique : brique de base de l'agent pulsionnel....184
4) L'agent tropique, brique de base pour d'autres agents....184
5) Transition vers l'agent pulsionnel, puis vers l'agent intentionnel....184
Ch. 24 De l'agent réactif tropique à l'agent réactif pulsionnel (AP)....185
1) Au départ : le forçage instinctif d'un état somatique interne....185
1.A) Des états hystérétiques....185
1.B) Aspect somatique : un forçage instinctif d'un état corporel....185
1.C) Importance du niveau d'énergie dans ce scénario de vie artificielle....186
2) Quels sont les rôles et effets de ces états ?....186
2.A) Des états finalistes....186
2.B) L'AP : N agents tropiques en parallèle, commutés par des états internes 186
2.C) Agent au comportement séquentiel....186
3) Application A : le cycle effort/réconfort....187
4) Application B : deux exemples d'alternateur....187
4.A) L'agent tropique : brique de base de l'agent pulsionnel....187
5) Conclusion : 188
5.A) Quelle représentation du monde pour un agent pulsionnel ?....188
5.B) Perspective à terme : en couches progressives vers les agents intentionnels....188
5.C) Annoncer la double transition de l'intentionnalité....188
Ch. 25 De l'agent intentionnel à l'agent Intentions....190
1) L'agent intentionnel....190
1.A) Introduction....190
1.B) Définir l'agent intentionnel....190

1.C) Quel type de connaissance est représenté au sein d'un agent Intentions ?....191
2) Communication pour transmettre une info intentionnelle entre agents....192
2.A) Introduction....192
2.B) Transfert de symbole entre agents intentionnels....193
2.C) Transfert entre agents d'une information codée sur plusieurs symboles....193
3) Conclusion : nous affirmons l'importance des agents Intensions....194
Ch. 26 Les agents Croyances pour simuler et anticiper....195
1) Implémentation des agents Croyances....195
1.A) Se baser sur les agents situés....195
1.B) S'accrocher à une description de structures....195
1.C) Précisons les notions de croyances, désirs et intention chez l'agent CDI....195
1.D) Pour simuler et anticiper il nous suffit de disposer d'agents Croyances....196
2) Communication pour transmettre une information entre agents....197
2.A) Introduction....197
2.B) Transfert de symbole entre agents....197
2.C) Transfert entre agents d'une information de lieu, de position....197
2.D) Transfert de règles entre agents....197
2.E) Remarque....199
3) Retour sur le synoptique de la simulation....199
3.A) Cycle élémentaire perception décision action....200
3.B) Le cycle de simulation fait intervenir la réponse de l'agent et du monde....200
3.C) Un séquenceur pour simuler à profondeur n....200
3.D) Passer de simulation sur l'ordinateur à imagination par l'agent Croyances....200
3.E) Fonctionnement en anticipation....202
3.F) Anticipation à profondeur unaire....202
3.G) Anticipation n-aire (à profondeur n)....204
4) Conclusion....205
Ch. 27 L'agent Croyances Désirs Intentions utilisé pour planifier....206
1) Introduction : l'agent réactif....206
1.A) Position du problème....206
1.B) Agents réactifs avec un choix heuristique dans les règles....206
2) L'agent CD (Croyances Désirs)....207
2.A) Plus de possibilités chez un agent intentionnel....207
2.B) Vie artificielle et pression sélective....207
2.C) Solution partielle : l'agent Croyances Désirs utilisé en proto-planification....208
2.D) Analyser cette solution partielle : l'agent CD utilisé en proto-planification....208
2.E) Remarque conclusion....209
3) L'agent Croyances Désirs Intentions (CDI) utilisé en planification....210
3.A) Introduction....210
3.B) Décrivons la position du problème....210
3.C) Pour cette tâche nous devons disposer d'un Agent CDI....210
3.D) Détaillons un peu le calcul de la combinatoire d'une planification....210
3.E) Exemple de planification : le problème du singe et de la banane....211
4) Implémenter la planification en IA symbolique classique....211
4.A) Combinatoire d'une planification utilisant la force brute....211
4.B) Les heuristiques permettent d'élaguer dans les arbres....212
5) Une représentation sub-symbolique pour une planification bon marché....214
5.A) Remarque à propos des êtres : agents CDI + méta-regard....214
5.B) Synoptique d'un traitement pré-cognitif minimaliste....214
5.C) Hypothèse forte ou faible....216
5.D) Conclusion :....216
Ch. 28 La cognition chez l'agent rationnel....217
1) Simuler un traitement rationnel cognitif sur des agents situés sociaux....217
1.A) Tout d'abord définissons la cognition de l'agent rationnel....217
2) Projet : les agents situés pour effectuer un traitement rationnel cognitif....218
3) Situer le traitement rationnel cognitif sub-symbolique....218
3.A) Deux orientations du traitement rationnel cognitif de l'IA symbolique....219
3.B) Conclusion....220
4) Représentation sub-symbolique pour un traitement rationnel cognitif....221
4.A) Notre définition du traitement rationnel cognitif....221
4.B) La cognition chez l'agent rationnel....222
4.C) Implémenter un exemple de ce traitement rationnel cognitif bon marché....223
4.D) Conclusions sur cet exemple de traitement rationnel cognitif bon marché ?....223

Partie pratique

Ch. 29 Classifier des simulations d'agents situés sub-symboliques....225
1) Problématique :....225
1.A) Quel est notre but ?....225
1.B) Fournir une progression d'outils classés selon une complexité croissante....225
2) État de l'art....226
2.A) Notre progression de simulations s'inscrit dans l'état de l'art de l'IA....226
2.B) Notre taxinomie reprend celle de Chomsky et la prolonge vers les agents....226
3) Les concepts mis en avant....227
4) Critère de la progressivité pour évaluer la qualité de notre classification....227

4.A) Chercher un critère pour quantifier la progressivité de notre classification....227
4.B) Traduction mathématique de critère de progressivité....227
5) Quelles démarches pour fournir une classification de simulations....228
5.A) Simplifier, pour épurer la structure, trouver de nouveaux outils universels....228
5.B) Une représentation et un traitement sub-symboliques de l'information....228
5.C) Méthodes de la VA pour évaluer une organisation mise en situation....229
5.D) Construire en couches concentriques pour obtenir une classification progressive....229
6) Quels résultats avons-nous obtenus ?....230
6.A) Le premier résultat consiste à fournir une classification arborescente....230
6.B) La taxinomie présente un tronc linéaire et une triple fourche....230
7) Nous obtenons une émergence déterministe....230
7.A) Résultats sur le plan concret....231
8) Évaluer ce résultat : vérifier la qualité de cette progression....232
9) Conclusion :....233
10) Perspectives....233
10.A) Fournir à l'utilisateur des résultats sous forme de courbes graphiques ?....233
10.B) Évaluer la complexité d'une simulation....233
Ch. 30 0DCelluleVégétative - La cellule végétative (CV)....235
1) Introduction de la cellule végétative....235
2) Outils, méthodes pour implémenter cette CV....235
3) programmes, méthodes et implémentation....236
3.A) Outils informatiques utilisés pour développer les .exe de cette progression....236
3.B) Programmation objet....236
3.C) Le programme d'initialisation : main.c....236
3.D) Modélisation informatique des relations que CV entretient avec l'énergie....238
4) Pas de résultat brut à présenter....239
5) Conclusion....239
6) Transition....240
7) Perspectives....240
Ch. 31 1DCorridor : un domaine où l'agent agit, avance....241
1) Situer l'étape dans la taxinomie....241
2) Introduire une représentation simple pour simuler....241
2.A) Représenter connaissances et actions dans les simulations de la taxinomie....241
2.B) 1DCorridor : le choix d'une représentation simple....241
2.C) Théorie : construire le domaine, l'interpréter, diverses conventions....242
2.D) Application pratique des graphes 1D....244
2.E) Le mobilier du monde se réduit à un ASM (agent situé minimal)....245
2.F) Représenter les relations du mobilier avec son support....246
2.G) Représenter les actions de l'agent sur le mobilier....246
3) Détailler l'implémentation informatique de la simulation....247
3.A) Détailler l'implémentation informatique du domaine (le support)....247
3.B) Détailler l'implémentation informatique du mobilier....249
3.C) Détailler l'implémentation des relations du mobilier↔support....250
3.D) Détailler l'implémentation des actions de l'agent sur le mobilier....250
4) Présenter les résultats (sans les interpréter)....252
4.A) Présenter le programme à l'utilisateur....252
4.B) Pas de résultat brut à présenter....252
5) Conclusion....253
6) Perspectives....253
Ch. 32 2D ASM⊥ : 2DAgentSituéMinimalOrthogonal....254
1) Introduction....254
1.A) Distribuer aléatoirement l'énergie....254
1.B) Situer l'étape dans la taxinomie....254
2) Quels outils informatiques pour modéliser cette étape ?....254
2.A) Modélisation du monde (support + mobilier)....254
2.B) Représenter le domaine (le support)....255
2.C) Présentation de la mise en oeuvre du programme par l'utilisateur....256
3) Résultats....256
3.A) Présenter les résultats....256
3.B) Interpréter les résultats....257
4) Conclusion....257
5) Perspectives....258
Ch. 33 1DCueilleur, l'agent situé fait le geste de cueillir sa nourriture....259
1) Situer l'étape dans la taxinomie....259
2) Introduire une représentation simple dans la simulation....259
2.A) Dans une simulation de l'agent réactif, le programmeur doit :....259
2.B) 1DCueilleur : un être causal qui constitue un vrai agent situé....259
2.C) Représenter le domaine (le support)....260
2.D) Représenter le mobilier du domaine....260
2.E) Représenter les relations du mobilier avec son support....261
2.F) Représenter les actions de l'agent sur le mobilier....262
3) Détailler l'implémentation informatique de la simulation....262
3.A) Détailler l'implémentation informatique du domaine (le support)....262
3.B) Détailler l'implémentation informatique du mobilier du domaine....262
3.C) Détailler l'implémentation des relations du mobilier↔support....263
3.D) Représenter les actions de l'agent sur le mobilier....264

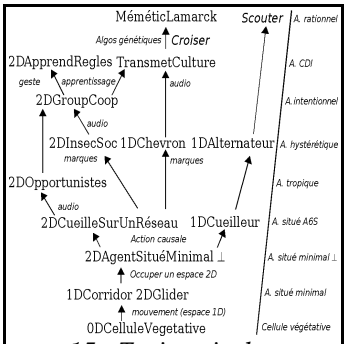
3.E) Représenter la prise de décision des actions de l'agent sur le mobilier....264
4) Présenter les résultats (sans les interpréter)....265
4.A) Présentation du programme à l'utilisateur....265
4.B) Résultats....265
5) Conclusion....266
6) Perspectives....266
Ch. 34 2DCueilleSurUnRéseau, chercher sa nourriture parmi les obstacles....267
1) Introduction....267
1.A) Passer à un monde 2D....267
1.B) Situer cette étape dans notre taxinomie....267
2) Matériels, méthodes et implémentation....267
2.A) Comment pour modéliser cette étape ?....267
3) Présenter les résultats (sans les interpréter)....269
3.A) Lancement, présentation et utilisation du programme....269
3.B) Résultats....269
4) Conclusion : vers une complexification....270
5) Discussion : l'action de l'agent réactif est-il sous-tendu par un but....271
5.A) Définition de la notion de procès....271
5.B) Définition de la notion d'action....271
5.C) Introduction de finalité implicite indirecte....271
5.D) Conclusion....272
6) Perspectives....272
Ch. 35 1DAlternateur, agent pulsionnel qui commute entre deux buts....273
1) Introduction....273
1.A) Mémoriser en marquant le monde....273
1.B) Situer cette étape dans notre taxinomie....273
2) Matériels, méthodes et implémentation....274
2.A) Quels outils informatiques pour modéliser cette étape ?....274
3) Présenter les résultats (sans les interpréter)....275
3.A) Lancement, présentation et utilisation du programme....275
3.B) Résultats....275
4) Conclusions par rapport à différentes sémantiques 276
4.A) Introduction....276
4.B) Sémantique dénotationnelle....276
4.C) Sémantique de l'efficacité d'une fonction, d'une couche, d'une structure....277
4.D) Perspectives....278
Ch. 36 2DOpportunistes : le groupe et la communication passive....279
1) Introduction....279
1.A) Situer cette étape dans notre taxinomie....279
1.B) Le groupe induit la communication....279
2) Matériels, méthodes et implémentation....280
2.A) Quels outils informatiques pour modéliser cette étape ?....280
3) Présenter les résultats....281
3.A) Lancer, présenter et utiliser le programme....281
3.B) Résultats....281
3.C) Interprétation des résultats....282
4) Conclusion : un agent tropique bien simple....282
Ch. 37 2DInsecSoc : la communication active du groupe 283
1) Introduction....283
2) Matériels, méthodes et implémentation....283
2.A) Quels outils informatiques pour modéliser cette étape ?....283
2.B) Simuler la communication par phéromones....284
3) Présenter les résultats....284
3.A) Lancement, présentation et utilisation du programme....284
3.B) Résultats....285
3.C) Interprétation des résultats....286
4) Conclusions....286
Ch. 38 2DGroupCoop : le groupe et la communication globale active....287
1) Introduction....287
2) Matériels, méthodes et implémentation....288
2.A) Modélisation du support : 288
2.B) Modéliser l'être comme un agent intentionnel hystérétique....288
3) Présenter les résultats....290
3.A) Lancement, présentation, utilisation du programme....290
3.B) Résultats pour 2DGroupCoopNat....290
3.C) Résultats pour 2DGroupCoopArt....291
3.D) Interprétation des résultats....292
4) Conclusions....292
Ch. 39 Apprentissage : 2DApprendRègles....293
1) Introduction de l'apprentissage....293
2) Problématique et démarche : traiter un exemple simple d'apprentissage....293
2.A) Présenter des constituants de cette simulation....293
2.B) Le parent chemine à l'est en appelant l'enfant qui apprend en le suivant....293
3) Outils et méthodes pour implémenter les modélisations....294
3.A) Modélisation du support : 294
3.B) Modélisation de l'agent apprenant....295

3.C) L'agent apprend à piloter son déplacement au milieu des obstacles....296
3.D) L'algorithme de poursuite : comment l'apprenant suit-il son leader ?....297
4) Résultats des trois apprentissages obtenus sur ces trois configurations....297
4.A) Résultats pour l'enseignement-apprentissage sur un support à 4 voisins....297
4.B) Enseignement-apprentissage sur un support à 6 voisins (config. facile)....299
4.C) Enseignement-apprentissage sur un support à 6 voisins (config. difficile)....300
5) Conclusions....304
5.A) Make it simple, stupid : simple is beautifull....304
5.B) Outil pédagogique....304
5.C) Transmission de règles épigénétiques....305
6) Transition....305
7) Perspective....305
Ch. 40 TransmetCulture : un parent transmet des règles à un enfant....306
1) Introduction....306
2) Problématique : introduire une communication langagière simplissime....306
3) Les modélisations utilisées dans cette simulation....306
3.A) Modélisation du support....306
3.B) Modélisation du mobilier : les matières....307
3.C) Modélisation du mobilier : l'agent....307
4) Implémenter ces fonctionnements....309
4.A) Implémenter le support, un tableau unaire de chevrons....309
4.B) Ne pas programmer la communication mais en simuler le résultat....309
4.C) Implémenter des agents : rajouter des champs qui sont testés à l'exécution....309
4.D) Implémentation de la reproduction....310
5) Présenter les résultats....311
5.A) Où les agents se reproduisent....311
5.B) Les agents communiquent....311
5.C) Les collisions entre agents....311
5.D) Analyse des résultats....312
6) Conclusions à propos de la structure du support....313
7) Transition....313
8) Perspective....313
Ch. 41 MémétiqueLamarck, transmettre le jeu de règles qu'on a créé....314
1) Introduction : effectuer un traitement cognitif facile à implémenter....314
1.B) Position de cette simulation dans notre progression de modèles....316
2) Quels outils informatiques pour modéliser cette simulation ?....316
3) Comment programmer deux espèces différentes d'agents....316
3.A) L'espèce initiale : des agents primaires inconséquents....316
3.B) Implémenter des agents, rajoutons des champs qui sont testés à l'exécution....317
4) Concrètement comment programmer la chose ?....318
4.A) L'espèce initiale : des agents primaires inconséquents....318
4.B) À la naissance, des êtres immatures....318
4.C) Une espèce mieux adaptée : les agents secondaires finauds et intelligents....318
4.D) Il existe encore deux espèces hybrides, mais médiocres....319
5) Présenter les résultats....322
5.A) Où nous voyons la mutation se produire....322
5.B) Discussion annexe : darwinisme ou lamarckisme....322
6) Conclusion : peu et beaucoup....323
Ch. 42 Conclusion....325
1) Agent situé : léger et simple pour représenter et traiter les données....325
1.A) Langages macroscopiques : représentation centralisée des données/procès....325
1.B) Les langages objets : décentralisation, héritage mais abstraction....325
1.C) Introduction de l'action : l'agent est plongé dans le monde où il agit....326
1.D) Introduction de l'agent situé....326
2) Dans le cadre des agents situés, nous reconstruisons notre plate-forme....327
2.A) Au début, nous commençons par le cycle de la cellule végétative....327
2.B) Nous décomposons notre classification arborescente en trois parties....327
2.C) Quel est notre but ?....327
3) Nous proposons un système, i.e. une taxinomie de simulations....328
3.A) Les propriétés de ce système de simulations d'agents situés sur un support....328
3.B) Conséquence conclusion....329
3.C) Chacun des systèmes présent dans notre taxinomie sert à :....329
4) Conclusion finale....330
Ch. 43 Bibliographie....331
Ch. 44 Index thématique....341

Ch. 4 Notre problématique

Fournir une plate-forme de simulations d'agents situés modélisant des phénomènes sociaux.

Notre but est de construire une plate-forme de simulations d'agents situés, et de la fournir avec une gamme d'exemples à des chercheurs qui voudraient modéliser différents phénomènes : cognitifs, de communication ou langagiers.



15 - Taxinomie des simulations proposées

1) Travail théorique et pratique sur la simulation

Nous proposons un travail théorique sur la simulation.

Tout d'abord, dans un travail théorique, sur un plan mathématique puis logique informatique, nous montrons comment, en partant du séquenceur et de l'automate à états finis, nous pouvons construire un agent situé.

De plus, nous traitons ces différents sujets selon un point de vue théorique. Nous travaillons à les rattacher aux réalisations classiques en mathématiques, informatique, sciences sociales et épistémologie.

Aspect pratique : proposer une gamme de simulations et la plate-forme correspondante

Ensuite, dans une partie pratique, nous proposons, classées dans une taxinomie, une *progression de simulations*, concrétisée par plus d'une douzaine de programmes exécutables, livrés avec le code. Les premières modélisations fournies sont organisées selon une structure en pelures d'oignons, ensuite elles se répartissent suivant trois thèmes : communication, apprentissage et cognition.

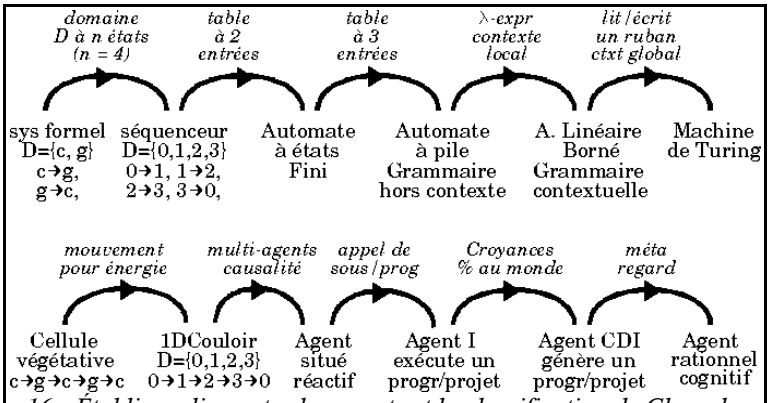
Puis nous fournissons la plate-forme informatique implémentant ces simulations. Comme une poupée gigogne l'être situé à l'extérieur incorpore son prédécesseur, dont il hérite. Finalement il dispose tout le code de ses *ancêtres*. De plus, le code fourni est écrit en C, mais est structuré comme du C^{++} : il est modulaire et adaptable.

2) Nous proposons une gamme de simulations, destinée à la recherche

2.A) Notre gamme de simulations est destinée à la recherche en informatique

Informatique : gamme liant la programmation par agents à la classification de Chomsky

Elle sert à structurer, ordonner la mouvance des outils de programmation : nous les classons selon les critères classiques en informatique et obtenons une taxinomie intéressante. D'abord son tronc est constitué par une progression basée sur les types de Chomsky (automate fini, automate à pile, langage contextuels). Ensuite, sa ramure est constituée par les langages objet et leurs évolutions (objets, agents, agents cognitifs).



16 - Établir un lien entre les agents et la classification de Chomsky

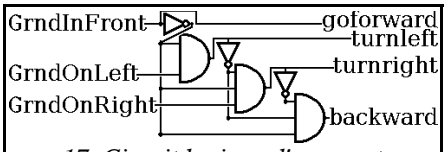
Nous proposons une réelle progression, intéressante sous deux aspects : d'abord, elle est construite formellement et ensuite, elle est différente, elle part du séquenceur et finit avec les agents situés sociaux (et un peu rationnels).

Cette gamme est utilisable en sciences humaines (psychologie, sociologie).
Elle modélise ou fait émerger des phénomènes de sciences sociales, i.e. cognition, psychologie, linguistique, sociologie. Nous présentons en particulier des simulations de phénomènes cognitifs et de phénomènes langagiers.

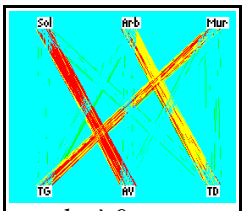
2.B) Cette gamme de simulations constitue un outil pédagogique

a) Un outil pédagogique qui fournit un agent dont la structure interne est accessible

À la base, l'agent est construit à partir des mathématiques.
L'utilisateur peut soulever le capot de la voiture, le moteur est visible. La structure de chaque agent utilisé dans les simulations est accessible : il est construit à partir d'outils mathématiques, les systèmes formels, puis les fonctions logiques combinatoires de Boole, et la logique séquentielle.



17 Circuit logique d'un agent fourageur



couche à 9 neurones

Ensuite l'agent est construit au moyen d'outils informatiques.
Son soma et sa psyché sont simulés par un programme informatique symbolique. Finalement, dans nos simulations les plus élaborées, situées en haut de l'arbre, nous modélisons sa psyché par une représentation et un traitement hebbien, i.e. sub-symbolique (ci-contre à gauche, une simple réseau constitué d'une couche de 9 neurones).

b) Les programmes exécutables fournissent une simulation

Nos exécutables simulent des agents situés, or dans cette simulation, tout est observable.
À partir du moment où le chercheur réussit à simuler un phénomène, i.e. à le faire tourner sur un de nos modèles, il obtient des résultats : d'abord chaque modélisation effectuée est programmée sur un simulateur dont tous les rouages sont accessibles. Ensuite, elle tourne basiquement sur des outils épurés qui modélisent la vie, l'agent, l'information, la cognition, la rationalité, alors l'expérimentateur récupère des résultats à ce propos, i.e. qui sont formulés dans des termes anthropomorphes.

Nous simulons plus particulièrement des phénomènes cognitifs puis langagiers.
Les phénomènes cognitifs puis langagiers constituent notre domaine de recherche. Ainsi, dans la progression de simulations que nous proposons, nous avons particulièrement imaginé des simulations où ces domaines apparaissent, et s'avèrent facilement observables. Mais rien n'interdit à l'utilisateur d'en imaginer d'autres et de les simuler selon ses centres d'intérêt personnels.

c) Des chapitres de ce mémoire peuvent servir de polycopiés pour un cours d'IA

Notre but est que des enseignants utilisent certaines chapitres de ce mémoire comme des polycopiés afin de donner un cours d'IA.

Nous souhaitons rendre accessible la simulation au moyen d'agents situés.
Certains chapitres de ce travail fournissent des explications qui se veulent claires sur la simulation au moyen d'agents situés, et nous faisons un effort de rédaction dans ce sens.

Nous essayons de proposer des simulations pratiques, dont les agents soient simples et légers.
Ainsi, dans notre progression, toutes les premières simulations sont basées seulement sur des agents réactifs, qui sont simples à cause de leur structure. Nous y trouvons quelques agents CDI, seulement vers les dernières simulations.

Nous fournissons le code des programmes informatiques, lui aussi est simple.

Il est écrit en C selon une architecture inspirée de C++. Selon une démarche implicite d' héritage, le code d'une simulation peut être réutilisé ailleurs avec seulement quelques variations.

Nous fournissons des illustrations : un petit dessin vaut mieux qu'un long discours.

Les explications sont illustrées par des dessins dédiés, dont la plupart est en noir et blanc, ce qui permet des polycopiés légers à stocker sur un fichier et faciles à imprimer. Cependant, quand pour des raisons de clarté nous ne pouvions pas dessiner en nuances de gris, quelques dessins sont réalisés en 16 couleurs. Finalement, nous avons adjoint quelques recopies d'écran qui s'avèrent très parlantes.

3) Proposer une taxinomie, un classement arborescent de simulations

Le choix des simulations recrutées au sein de notre gamme est large, précisons maintenant sur quels critères elles sont choisies et ensuite classées.

3.A) Deux critères pour classer les simulations fournies

Avec cette gamme de simulations, nous fournissons à l'utilisateur une progression d'exemples de modélisations. Chacune d'elles est orientée, classifiée dans une arborescence, une taxinomie, selon deux critères couplés :

a) Critère de la structure, plus visible dans la partie théorique

La gamme des simulations proposées dans cette progression modélise la structure des êtres, depuis le séquenceur, en passant l'agent situé, jusqu'au groupe social.

1) Une progression en couches concentriques vers l'individu reconstruit l'évolution de l'AF jusqu'à l'agent situé.

Nous guidons une émergence de la structure des agents. En partant du séquenceur et de l'AF, nous empilons des couches concentriques, pour reconstruire jusqu'à l'agent situé.

2) À partir de l'agent situé, construction de l'agent intentionnel.

À partir de l'agent situé, qui est seulement réactif et tropique, nous ajoutons progressivement une fonction de mémorisation, afin de construire l'agent intentionnel. Au début, avec l'agent pulsionnel, cette mémorisation est implicite : elle est réalisée par le corps somatique ou marquée sur le monde. Ensuite, avec l'agent intentionnel, elle est effectuée explicitement par une fonction mémoire élémentaire interne à l'agent.

3) À partir de l'agent intentionnel, construction de l'agent CDI.

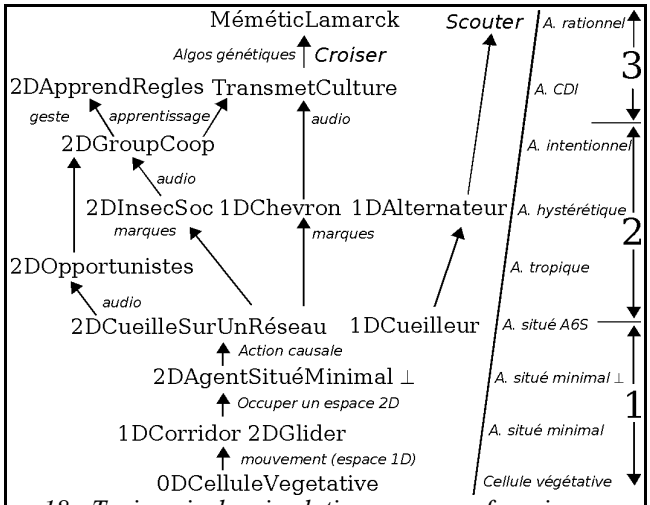
La mémoire de l'agent intentionnel s'améliore : elle grandit, se spécialise et reçoit un type : croyance, désirs et intention.

b) Le critère de l'organisation sociale (plus visible dans la partie pratique)

Dans cette classification, l'organisation sociale est d'abord centrée sur l'individu, puis elle s'organise autour du groupe :

1) l'individu : émergence de l'être, depuis la cellule végétative jusqu'à l'agent situé.

D'abord l'individu : notre démarche consiste à guider une émergence de l'agent, depuis la cellule végétative jusqu'à l'agent situé. Ainsi, après la cellule végétative, chacune des simulations de l'étape 1, modélise un individu isolé, qui apprend à se déplacer et collecter de la nourriture.



18 - Taxinomie des simulations que nous fournissons

Partant des agents situés ou bien intentionnels, quand nous introduisons le parallélisme, dans le monde apparaît la multitude informe. Quand nous rajoutons aux agents des capteurs et actionneurs pour communiquer, grâce à la communication, la masse s'organise progressivement en groupe social.

2) le groupe d'agents hystérétiques[4] : simuler des phénomènes cognitifs, de communication puis langagiers.

Au stade de l'agent hystérétique, nous commençons par modéliser des phénomènes cognitifs simples, puis de communication élémentaire, pour finalement aborder des phénomènes langagiers simples : nous simulons seulement une communication primaire de symboles décrivant le monde.

3) Le groupe : après l'agent intentionnel, la classification se sépare en communication, apprentissage et cognition.

Grâce à sa communication plus développée, l'agent CDI pratique une communication secondaire, culturelle, qui transmet des règles de conduite. Ceci induit dans l'organisation correspondante de notre taxinomie des simulation, une structure arborescente qui se ramifie dans trois directions.

- D'abord, tout simplement, des agents qui communiquent de faits à propos du monde.

- Ensuite, une communication parent-enfant de règles de conduite, d'attitude culturelles (scripts, scénarios, protocoles), ce qui induit l'apprentissage : l'enfant apprend un comportement de son parent.

- Et finalement, selon un point de vue global, en considérant le groupe comme une entité macroscopique, nous constatons que sa communication interne fait qu'il progresse, qu'il s'adapte au monde, parce qu'il fait évoluer sa culture. Alors nous considérons que le groupe qui communique effectue un traitement de l'information, et qu'il produit un résultat qui se concrétise dans un nouveau savoir culturel codé par de l'information.

3.B) Notre but est de proposer une progression de simulations adaptées

Chaque simulation de notre progression est juste et dépouillée.

- Chaque simulation est juste dans le sens de *sound*, bien adaptée, i.e. à chaque fois que nous rajoutons une fonctionnalité à la structure, elle est validée par une augmentation de son efficacité.

- Chaque simulation est dépouillée : elle ne comporte pas de partie, de fonctionnalité qui soit inutile au fonctionnement général de l'agent.

4 Agents hystérétiques : du grec husterein (retarder). Ce sont des agents qui mémorisent l'information. Ils la gardent la retiennent.

Le critère d'admissibilité d'une simulation dans notre gamme est celui de la continuité de la progression obtenue.

Il importe, qu'au sein de ce classement, chacune des simulations s'inscrive bien à sa place, sans gap, i.e. sans faire apparaître de rupture dans la continuité de la progression obtenue.

3.C) Quelle méthode guide le choix des branches qui sont ajoutées à l'arbre

Au début, la nécessité nous guide dans la construction du tronc de notre arborescence, qui est ainsi déterministe.

Par exemple, au début de la construction de notre monde, nous utilisons un support de lieu, alors, au vu de son architecture, l'utilisation du déplacement se révèle une nécessité : dans la modélisation de nos agents, le mouvement apparaît dès le début, de façon inéluctable et donc déterministe.

Ensuite, selon une démarche bionique, l'observation de la nature guide nos choix.

Nous introduisons dans nos modèles, des attitudes copiées dans la nature : marquage du support, organisation en groupe, communication audio.

Finalement, quand viendra l'imagination exubérance de la nature, nous serons éclectiques et opportunistes.

Par la suite, quand les simulations vont se transporter vers des mondes plus variés, apparaîtra le phénomène des niches écologiques. Dans ce cas le choix des simulations à placer dans notre classement deviendra de moins en moins déterministe. Il deviendra éclectique car nous devrons faire le tri parmi les formes qui vont apparaître. Et si surviennent des formes intéressantes que nous n'avons pas prévues, alors nous devrons être opportunistes et les accepter.

Ch. 5 Histoire du traitement de la complexité en IA

Dans ce chapitre nous positionnons les systèmes rationnels cognitifs par rapport à la complexité.

Dans les systèmes à base d'agents rationnels cognitifs [Gue 2003], la complexité des structures à mettre en oeuvre est tellement grande qu'elle devient difficile à appréhender. Avant de recenser les difficultés induites par notre problématique dans cette histoire de l'art, nous retraçons le fil conducteur du combat que les générations de chercheurs ont mené contre la complexité.

Finalement, à l'issue de ce chapitre, nous préconisons d'utiliser les systèmes formels, i.e. leur représentation et leur traitement fonctionnel de l'information.

1) Au début : des chercheurs peu conscients du problème de la complexité

Au début de l'ordinateur, les informaticiens qui travaillent à créer des systèmes artificiellement intelligents [Ric 1987] se montrent inconscients de la complexité du problème qu'ils abordent. Voyons quelques exemples illustrant les errements de cette époque.

1.A) L'enthousiasme des pionniers est battu en brèche par la réalité des faits

En 1956, au Dartmouth College, John McCarthy réunit les pères fondateurs de l'IA, (Herbert Simon, Allen Newell, Seymour Papert [Pap 1981], Marvin Minsky [Min 1967] [MP 1972], Claude Shannon, Arthur Samuel).

Constatant la croissance exponentielle de la puissance de traitement des ordinateurs et des programmes, à l'instar du programme de jeu de dames de Samuel [Sam 1958], il prédisent qu'avant 10 ans l'ordinateur battrait l'homme dans les jeux majeurs.

Pour les échecs, cette prévision était déjà malheureuse. En effet, nous savons maintenant que cet événement s'est seulement produit avec 30 ans de retard, le 10 février 1996 à Philadelphie, où l'ordinateur Deep Blue d'IBM à battu Garry Kasparov. le champion du monde en titre.

Pour le jeux de go, ces pionniers se sont largement trompés, car, à l'heure actuelle aucun ordinateur ne peut battre les meilleurs maîtres. Il nous faudra encore attendre un moment [BH 2003] !

1.B) Le subterfuge de Éliza de Weizenbaum met à mal le test de Turing

Cerner le problème de la cognition, définir la notion d'intelligence, beaucoup s'y sont essayés, et même les plus grands ont éprouvé de la difficulté face à cette complexité. Ce fut aussi le cas de Alan Turing.

Le test de Turing [Tur 1950].

Alan Turing, bien que conscient de la difficulté à cerner la notion d'intelligence, s'attache à définir des critères pour tester les aptitudes de l'ordinateur dans ce domaine. Il met en avance son fameux *test de Turing* :

Grâce à l'interface fournie par un télétype, un humain discute avec un correspondant afin de le tester.

- Si l'utilisateur réalise rapidement qu'il échange avec une machine, alors l'ordinateur à échoué à montrer de l'intelligence.
- Mais si l'ordinateur répond si justement que le testeur ne parvient pas à distinguer s'il converse avec un humain ou un artefact, alors on conclut que la machine a passé le test avec succès.

Le subterfuge de Éliza de Weizenbaum [Wei 1966].

D'abord Weizenbaum se place intelligemment dans le cadre biaisé de la psychanalyse Rogérienne[5] [Pia 1964]. Selon les spécifications de cette thérapie, le soignant se montre plein d'empathie à l'égard du malade. Loin de juger les actions qu'on lui décrit, il est compatissant, et ne fait que reformuler les déclarations de son patient.

Par exemple le thème T est relancé au moyen de la structure : *Parlez-moi de T.*
- Patient : *J'ai toujours eu un problème avec mon père.*
- Psychanalyste : *Parlez-moi de votre père.*

On trouve encore une relance de la forme : *Qu'est ce qui vous fait penser que T.*
- Patient : *Vous dîtes des non-sens.*
- Psychanalyste : *Qu'est ce qui vous fait penser que je dis des non-sens ?*

Dans ce cadre biaisé, l'implémentation de Éliza devient simple.

Nous constatons qu'à l'intérieur de ce cadre, pour le psychanalyste, tenir une conversation consiste principalement en une reformulation des dires du patient. Ainsi Weizenbaum implémente un programme qui utilise des mots clés pour détecter des structures récurrentes de la conversation et ré-agence les phrases du patient pour les reformuler.

Quand il échoue à trouver pas des mots-clés, il relance la conversation par des généralisations construite au moyen d'attachements sémantiques : mère → famille, père → famille.
- Patient : *Je crois que ma mère me déteste.*
- Psychanalyste : *Parler moi de votre famille.*

Comment Éliza met à mal le test de Turing ?
La chose apparaît très clairement au travers d'une des formulations de relance.
- Patient : *Pour moi, c'est vraiment difficile.*
- Psychanalyste : *Je vous comprends.*

Dans cette réponse mécanique de la machine, l'utilisateur ressent une empathie de son soignant, pourtant nous savons qu'en fait, il y projette sur la réponse du psychanalyste, un sens qui est totalement absent.

Finalement nous concluons que Turing est pris en défaut : l'ordinateur passe son test, sans pour autant être intelligent. En effet, Éliza est confondue avec un humain car le patient perçoit son correspondant comme une personne très empathique. Cependant, sémantiquement parlant, on peut affirmer que le programme informatique ne comprend rien de son patient : avec Éliza, Feigenbaum met à mal le test de Turing.

1.C) Les praticiens des sciences humaines modèrent l'ardeur des informaticiens

Au début de l'ordinateur, les techniciens de l'informatique focalisent sur leurs machines et négligent un peu trop les sciences humaines : le nez sur leur guidon, ils peinent à établir des passerelles entre les disciplines et cherchent peu à synthétiser leurs approches. Ceci explique que certaines grandes découvertes utilisées en IA ont été effectuées par des psychologues : les réseaux de neurones de Hebb [Heb 1949], et les réseaux sémantiques de Quillian [CQ 1969].

Rapidement les pionniers débordent des domaines de l'arithmétique et de la gestion, pour déplacer leur centre d'intérêt vers les sciences humaines : le langage, la cognition... Cette ardeur naïve des mathématiciens et informaticiens offusque les humanistes qui ne manquent pas de les mettre en garde.

a) La chambre chinoise de Searle [Sea 1980]

Sans rentrer dans les détails, disons que Searle en 1980, demande comment un ordinateur symbolique, qui ne fait que de manipuler des règles formelles, peut tirer un sens du traitement qu'il effectue. En effet, le questionnement de ce philosophe est compréhensible, car il se place au niveau basique du système formel qui applique mécaniquement les règles : à ce niveau si bas, il ne peut voir émerger du sens. Il devrait plutôt se placer selon un point de vue plus élevé, plus global.

Pour nous, ce questionnement de Searle à propos de la sémantique, vient à point. Dans notre recherche, sur les actions des agents situés [PLP 2002], nous travaillons à faire apparaître du sens dans les procès effectués : nous les sous-tendons par des buts, afin qu'ils deviennent des

5 Cet adjectif est construit d'après le nom du psychanalyste de *Carl Rogers*.

actions. Pour analyser si les actions basiques que ces êtres effectuent, atteignent les objectifs que nous leur fixons, nous devons changer de point de vue, i.e. nous placer à un niveau macroscopique global.

Tout d'abord, avec les systèmes à agents [Fer 1995], une solutions simple consiste à placer des êtres en situation de survie sur une grille où il cherchent de la nourriture. Le changement de point de vue, le positionnement à un niveau d'observation global qui permet l'évaluation de la justesse de leur comportement, se fait indirectement au moyen de la vie artificielle : le programme comptabilise la nourriture qu'ils trouvent et consomment. Ainsi, il finit par éliminer ceux qui jeûnent trop, ce qui simule une mort par inanition.

Généralement, en IA classique, ce traitement s'effectue depuis un point de vue macroscopique, appelé *niveau méta* ou encore *méta-regard* [Pit 1990]. Il s'implémente au moyen d'une programmation symbolique très lourde, et s'applique aux démarches de simulation et de planification. Mais Searle, de son propre avis, ne connaît pas grand chose à l'IA et il ignorait ces solutions. Il ne pouvait donc pas comprendre comment, en changeant de point de vue, en prenant un peu de hauteur, en utilisant des règles de haut niveau à propos de règles de bas niveau, on pouvait effectuer des traitements sémantiques au moyen d'un ordinateur.

b) Hubert Dreyfus [Dre 1972], philosophe, professeur à Berkeley

" *La compréhension stricto sensu implique tout un sens commun. Et comme l'on ne sait pas cerner de façon adéquate cette question, les programmes d'IA relèvent de la contrefaçon* "...

Le sens commun, la psychologie du quotidien ne sont pas si naïfs que ça.

Ils correspondent au savoir appris par interaction ou reçus à propos du monde. Ils concernent notre relation au monde (physique naïve), des protocoles de coopération avec un partenaire ou le groupe (psychologie sociologie naïve). Même s'il sont parfois qualifiés de naïfs, tous ces domaines ne sont pas si simplistes que ça. Ainsi, appréhender et décrire le sens commun et la psychologie du quotidien [Jac 1992], représente donc une tâche très lourde, et Dreyfus a raison de nous mettre en garde.

Conscient de cette difficulté à modéliser le sens commun nous le construisons en couches concentriques.

Dans l'arbre des simulations que nous proposons, nous classons des agents construits progressivement (bottom up), depuis la cellule végétative initiale jusqu'à l'agent CDI rationnel. Partant de la base, nous commençons par le tronc, qui correspond à la construction en couches concentriques de l'agent situé réactif. Puis vient l'embranchement correspondant à la construction de l'agent tropique, puis de l'agent intentionnel. Enfin les branches partent vers trois directions : communication, apprentissage et traitement de l'information [Car 2004].

Une fois muni de ces agents, nous oeuvrons à recueillir le fruit de notre travail. En effet, des informations de sens commun que nous recherchons, sont représentées plus ou moins implicitement, plus ou moins directement, dans la structure de nos agents. Selon la complexité de chacun, cette information correspond à la représentation que l'être se construit progressivement du monde, du groupe, (du partenaire et de lui-même)[6].

La difficulté à aller chercher les informations de sens commun codées au sein des agents.

En plus, une partie du problème consiste déjà à savoir où chercher cette information codée au sein de l'agent.

Donc, d'abord, dans la partie 1 de notre taxinomie, qui correspond au tronc de notre arborescence, nous commençons à traiter principalement les interactions simples de l'agent qui se déplace dans le monde physique. Chez cet agent réactif tropique, la représentation du monde est difficilement accessible, car elle est cachée, synthétisée, au fond de ses règles de pilotage.

Puis, dans la partie 2 de notre taxinomie, qui correspond à la fourche de notre arbre, nous traitons un peu des premières interactions avec le groupe ou avec les autres. Chez cet agent pulsionnel, puis intentionnel, la représentation du monde demeure générale[7] et peu typée. Mais

6 La représentation du partenaire et la conscience de soi ne sont pas disponibles chez nos agents qui sont très simples. En effet, les êtres que nous simulons possèdent une représentation plus ou moins implicite du monde et des autres, mais ne vont pas jusqu'à se représenter leur partenaire et eux-même.

7 Ici la représentation du monde est générale. Nous hésitons pour savoir quel type attribuer au digit qui code cette information : *état du monde* ou bien *état interne*. En cela, l'exemple typique est le message *alerte*, qui

par contre, elle est codée explicitement dans une mémoire élémentaire que l'agent gère : il la positionne ou l'efface au moyen d'instructions explicites (faim/nourriture, froid/chaleur, soif/eau, peur/prédateur).

Enfin, dans la partie 3 de notre taxinomie, correspondant aux branches de notre arbre, nous présentons des simulations qui illustrent et modélisent des phénomènes de communication, apprentissage et traitement de l'information. Chez ces agents CDI, la représentation du monde devient très visible. D'abord, elle est codée explicitement dans une mémoire de grande capacité que l'agent gère (agents Croyances). De plus il la positionne ou l'efface au moyen d'instructions explicites.

Prospective : ce que nous n'avons pas fait. Ensuite, il reste à continuer dans cette direction, à utiliser des agents plus riches, afin de progresser en vers les agents SOAR et leurs interactions cognitives complexes [New 1990].

En décrivant un sens commun à propos du monde physique nous buttons contre le *qualification problem* [Car 1959].

La première étape pour donner un sens commun à un agent est de lui enseigner le monde physique. Et, pour ce faire, les chercheurs buttent sur le *qualification problem (inscrutabilité de l'identité radicale)*, une difficulté décrite très tôt en IA.

Afin d'éviter ce problème nous utilisons les jeux digitaux [Hau 1985][Hau 1989] et l'interaction entre agents.

Au départ, nous commençons en travaillant sur un monde digital, muni de lois physiques simplifiées. Ainsi les relations que nous établissons entre le support et son mobilier sont si simples que nous évitons ce problème. Malgré cet aménagement osé, dans ce monde simplifié à outrance, les interactions entre les agents conservent une épaisseur et gardent encore du sens.

Ensuite, pour donner du corps à notre travail, nous devons enrichir notre modèle. Alors, en le construisant progressivement en couches concentriques, nous nous attachons à ne pas casser notre jouet, i.e. à toujours éviter ce problème grave de l'identité radicale. La chose peut se faire en imaginant à chaque étape des mondes spéciaux où les agents demeurent simples alors que les interactions prennent de l'épaisseur.

c) Bar-Hillel critique le mémorandum de Weaver sur la traduction automatique

En juillet 49, le mathématicien américain Warren Weaver, publie son memorandum dans lequel il préconise d'utiliser des ordinateurs pour traduire le langage naturel, d'une langue à une autre [Wea 1949]. Conscient de la difficulté du problème, il propose des solutions pour mettre au point une méthode plus fructueuse que la simple traduction mot à mot.

L'impact de son intervention est dû non seulement à ses compétences largement reconnues en mathématiques et dans le domaine de la traduction automatique, mais surtout aux relations qu'il entretient avec les décideurs politiques dans les agences internationales. En effet, dans cette période de guerre froide les services de contre-espionnage et surtout la NSA croulent sous les documents russes à traduire en anglais afin de les exploiter. Alors, l'aide hypothétique de l'ordinateur est bien venue : de gros crédits sont débloqués pour financer son initiative.

Mais, malgré les gros moyens mobilisés, les travaux piétinent. En 1960, puis en 1964, Yehoshua Bar-Hillel, mathématicien, philosophe et linguiste, critique ces travaux dans un papier très explicite : *A Demonstration of the Nonfeasibility of Fully Automatic High Quality Translation*, [Bar 1960], [Bar 1964] qui critique cette démarche et amène les USA à réduire sévèrement les crédits qui lui sont attribués :

> *During the past year I have repeatedly tried to point out the illusory character of the fully automatic high quality translation ideal even in respect to mechanical determination of the syntactical structure of a given source-language sentence. Here I shall show that there exist extremely simple sentences in English—and the same holds, I am sure, for any other natural language—which, within certain linguistic contexts, would be uniquely (up to plain synonymy) and unambiguously translated into any other language by anyone with a <u>sufficient knowledge</u> of the two languages involved, though I know of no program that would enable a machine to*

décrit aussi bien le monde : *un prédateur rôde*, que l'état interne correspondant : *il faut faire attention*.

> *come up with this unique rendering unless by a completely arbitrary and ad hoc procedure whose futility would show itself in the next example.*

On retrouve encore dans ce texte l'argument du savoir contextuel, du background culturel qui renvoie toujours au sens commun, que nous avons déjà évoqué et traité dans ce chapitre.

2) Histoire de l'art, le structuralisme : tout rattacher à une structure typée

2.A) Introduction : tout rattacher à une structure [Jak 1981]

En 1929, le Cercle linguistique de Prague préconisait pour la linguistique *une méthode propre à permettre de découvrir les lois de structure des systèmes linguistiques et de l'évolution de ceux-ci*. À l'instar de ces linguistes, pour éviter les discussions qui durent jusqu'à la nuit, nous introduisons un peu de rigueur dans notre travail et l'attachons, l'ancrons à une étude des structures du monde (support et agents). Nous focalisons surtout sur celle de l'agent.

2.B) Définissons les notions de *structure* et de *structuralisme*

- Une structure est une *entité de dépendances internes*[8].
- D'après Émile Benveniste [Ben 1966] [Ben 1974], s'appuyant sur les travaux de Saussure et du Cercle de Prague auxquels il fait largement référence, le structuralisme est l'hypothèse selon laquelle on peut étudier une langue en tant que structure.

2.C) La logique : une représentation structurée et statique du système/monde

a) Présenter la logique classique

Par *logique*, ici nous entendons la *logique classique*, celle qui part de l'algèbre de Boole [Boo 1854], qui est prolongée par De Morgan et Lewis Carol [Car 1886], et qui va jusqu'au calcul des proposition ou des prédicats [Car 1896].

b) Le système mathématique plongé dans la réalité devient un monde

Dans les applications de la vie courante, le système mathématique abstrait s'incarne dans un monde (ou un micro-monde) et dans les simulations de notre plate-forme nous le décrivons au moyen d'un sous ensemble de la logique.

c) La logique utilisée pour une représentation des connaissances est <u>statique</u>

En RC, la logique est pratique pour décrire l'état du monde.

Dans notre travail, nous utilisons la logique en RC car elle est pratique. Elle nous sert à représenter une situation, i.e. l'état du monde, sous la forme d'une structure.

Nous utilisons une version réduite de la logique pour décrire l'état du monde.

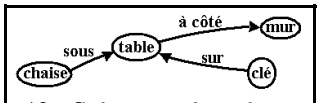
Dans les simulations de notre plate-forme, nous utilisons une version réduite de la logique, i.e. qu'avec ceci :

- Des relations vraies à propos du monde.
- La fonction logique $\wedge$ comme connecteur afin de rassembler ces relations au sein d'une liste.
- Des constantes, chacune représentera un objet du monde.

Nous délaissons les quantificateurs, et l'implication.

Exemple de description d'un monde structuré, au moyen de relations.

La scène *S* ci-contre est représentée au moyen d'un graphe. Nous pouvons facilement le transcoder en logique. Nous convenons d'écrire les relations i.e. les prédicats en minuscules et de commencer les constantes par une majuscule.



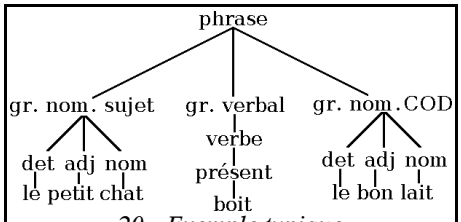
19 - Scène représentée au moyen d'un graphe

S = sous(Chaise, Table) $\wedge$ àcôté(Table, Mur) $\wedge$ sur (Table, clé).

8 Louis Hjelmslev (1899-1965), linguiste danois qui a prolongé les réflexions de F. de Saussure en fondant la glossématique.

d) L'inférence du *modus ponens* n'est pas dynamique mais statique

Dans le cadre de la logique classique, l'inférence effectuée au moyen du *modus ponens* induit un changement dans la représentation, mais pour autant elle demeure statique et ne représente pas le mouvement.



20 - Exemple typique d'une application du structuralisme : l'analyse syntaxique d'une phrase.

L'inférence de la logique classique, réalisée par le *modus ponens* est incrémentale : elle enrichit notre connaissance du monde. En fait elle sert à raffiner notre information. Elle rajoute un fait à son propos, mais elle demeure statique, elle ne change pas la moindre relations entre les objets du monde. Ainsi elle ne peut pas représenter le mouvement.

2.D) Applications des structures et du structuralisme

Regardons quelques exemples d'applications des structures et du structuralisme.

Tout d'abord nous citons l'exemple classique de l'analyse syntaxique d'une phrase du langage naturel (exemple ci-contre) [Cho 1969], [Bob 1969].

Ensuite nous pouvons citer l'exemple tout aussi classique de l'analyse syntaxique du code d'un programme structuré. Toutefois nous ajoutons que quand le programme est typé, le compilateur effectue en plus une analyse sémantique en vérifiant la compatibilité des types passés en arguments des fonctions et procédures.

Puis nous citons l'exemple de la construction d'une *formule bien formée* en logique (calcul des prédicats) qui elle aussi fait intervenir des contraintes syntaxiques et sémantiques. Par exemple, les fonctions logiques $\wedge$ $\vee$ $\neg$ en plus de contraintes syntaxiques, font aussi intervenir des contraintes sémantiques : elles ne peuvent pas directement recevoir des paramètres sous forme de constantes ou de variables, mais ces dernières doivent d'abord transiter par des prédicats qui fournissent en sortie des valeurs binaires acceptées par les fonctions logiques.

Enfin nous citons la hiérarchie de Chomsky [Cho 1969], dans laquelle il classifie en fonction de leur structure, les différents types de langage et le système formel symbolique correspondant qui les traite.

Un exemple qui introduit une première transition vers le traitement dynamique de l'information.

Finalement citons les hypothèses de Chomsky sur l'organisation initiale du cerveau, car nous les reprendrons dans la suite de notre thèse : il suppose que la morphogénèse y construit une structure innée dont la fonction est d'effectuer l'analyse et le traitement du langage.

Ainsi, selon ce positionnement, il dépasse le structuralisme : il n'en reste pas à l'analyse *statique* de l'information mais il passe à son traitement dynamique. En effet, dans sa grammaire générative, il fait intervenir des règles de réécriture qui transforment la forme profonde de la représentation de la phrase.

En conclusion, cette évocation nous permet d'annoncer une transition marquée par la notion de traitement, de procès. Elle correspond à un passage, depuis une analyse statique de l' information, vers son traitement dynamique, que nous développons dans le prochain chapitre.

2.E) L'impasse de la recherche de la grammaire d'une langue

Chaque société humaine qui utilise une langue vivante doit en connaître la grammaire

Une langue vivante sert à communiquer. Mais, puisqu'elle est basée sur des conventions arbitraires, elle est infiniment mouvante. Alors, afin que tous se comprennent, il est nécessaire de convenir de règles à appliquer entre utilisateurs. Ceci explique la nécessité de disposer de la grammaire d'une langue vivante. Donc, le rôle social des linguistes consiste, entre autres, à définir la grammaire du langage.

Afin d'obtenir la grammaire d'une langue, la solution n'est pas d'analyser plus finement le langage.

Mais, par définition, une langue vivante change constamment : sa grammaire évolue chaque jour. Et cela est encore plus vrai à notre époque où, de plus en plus rapidement, les techniques font évoluer notre façon de communiquer.

Par ailleurs, afin de décrire finement une syntaxe, un grammairien doit fournir plusieurs années d'effort. Mais, à peine son travail est terminé, qu'il est déjà démodé. Ainsi, le linguiste qui s'attache à fournir une grammaire précise du langage, remplit le tonneau des Danaïdes : il doit sans cesse recommencer.

Cette direction constitue une impasse.

La solution serait d'analyser dynamiquement le langage naturel.

Ainsi, certains chercheurs travaillent à déterminer la grammaire d'une langue dynamiquement, i.e. par ordinateur. Ils lui fournissent de gros corpus du langage, et la machine effectue rapidement l'analyse, sans trop demander d'interventions humaines. C'est ce que font, au LIASD, Gilles Bernard [Ber 2000], Jean-Jacques Mariage [Mar 2006] et Anna Pappa [Pap 2009].

2.F) Le classement pour obtenir une taxinomie de structures ?

Regardons un structuraliste dont le travail consiste à rechercher des structures. Elles peuvent apparaître, par exemple dans les règles de grammaire d'une langue, ou dans les systèmes de traitement de l'information. Mais comment procède-t-il ? À l'instar des biologistes qui font de l'anatomie comparée, il analyse les architectures des objets étudiés, et les classe, en fonction de leurs structures, au sein de taxinomies.

C'est une démarche qui fonctionne à peu près, mais quand l'objet de l'étude est la grammaire de la langue ou le traitement de l'information, le scientifique doit s'attacher de longue année à un recensement servile et systématique des structures qui apparaissent : l'affaire se révèle être un travail de bénédictin, simplement parce que ces domaines sont en perpétuelle évolution.

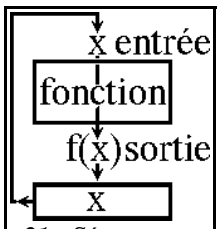
Mais alors, que faire, comment procéder ? Dans la suite de cette thèse nous allons évoquer quelques pistes : au lieu d'attaquer globalement le problème, nous proposons de le traiter progressivement, de sérier les choses en utilisant le critère de la phylogenèse, i.e. de l'évolution des espèces vivantes. Bien sûr nous focalisons sur les étapes concernant les apparitions du traitement causal de l'information, de la communication, du traitement rationnel cognitif et du langage.

3) Histoire : Le fonctionnalisme : une représentation dynamique de l'action

3.A) Définir le fonctionnalisme en philosophie de l'esprit (computationnalisme)

En philosophie de l'esprit, le computationnalisme est une théorie fonctionnaliste qui, pour des raisons méthodologiques, conçoit l'esprit comme un système de traitement de l'information. Elle compare la pensée à un calcul (en anglais, computation) et, plus précisément, à l'application d'un système de règles (Hilary Putnam [Put 1992] et Jerry Fodor [Fod 1975]).

Le fonctionnalisme/computationnalisme tire son nom du terme *fonction*, chacune d'elles est plus qu'une structure, car elle effectue un traitement caractérisé par la donnée de $y = f(x)$, i.e. de sa sortie y en fonction de son entrée x. Et, plus généralement, f est caractérisée par la donnée de $y = f(x_1, x_2 \ldots x_i)$, i.e. de sa sortie y en fonction de ses entrées x_i.



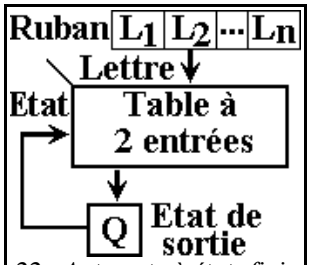
21 - Séquenceur basé sur y=f(x)

Initialement, dans le chapitre précédent, nous partons du structuralisme qui focalise seulement sur une structure afin de la classifier dans une taxinomie (de structures). Puis dans ce chapitre, nous définissons le fonctionnalisme sur la base de la fonction. Ensuite, nous lui rajoutons une mémoire qui enregistre y l'état du système. Ainsi nous disposons d'un outil plus puissant qui représente sa transformation, depuis son état x jusqu'à l'état $y = f(x)$: maintenant, nous somme en présence d'une représentation dynamique du traitement de l' information.

3.B) Modèle et implémentation du fonctionnalisme : l'automate à états finis

Regardons un exemple d'implémentation d'un traitement fonctionnel : l'automate à états finis (automate fini) [Min 1967].

Afin d'illustrer le traitement de l'information effectué par un système fonctionnaliste, nous utilisons un automate fini (AF). Outil de base de l'informatique, il nous sert à implémenter les systèmes les plus simples. Il est constitué d'une table à deux entrées et d'un registre mémoire.



22 - Automate à états finis

La table implémente la fonction.

La table fonctionne comme un tableau à deux entrées $y = f(x_1, x_2)$, où x_1 est une lettre en provenance du ruban, et x_2 l'état actuel de l'automate.

Le registre mémorise l'état du système.

Le registre mémorise l'état du système sous la forme d'un symbole. Il permet les transitions d'état du système.

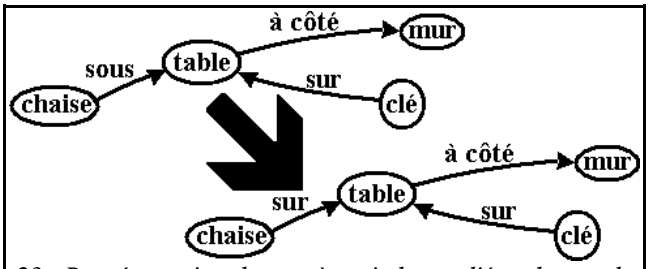
3.C) Représentation au moyen de la logique : passer du statique au dynamique

a) La variation de relations pour représenter le procès

Nous savons représenter l'état du monde au moyen de plusieurs relations entre ses composantes. Convenons d'appeler *procès* le changement d'état du monde, i.e. le changement d'une (ou plusieurs) de ces relations. Donc la représentation d'un procès se fait simplement en changeant une ou plusieurs relations entre les composantes du monde. Par exemple, dans l'illustration ci-contre, la chaise qui était sous la table, passe sur la table.

b) La logique utilisée pour représenter l'action

La logique éprouve une difficulté dans la RC dynamique, elle peine à représenter le procès/l'action qui change l'état du monde. Bien sûr cette représentation peut tout de même s'effectuer au moyen du *modus ponens* [Car 1886] [Car 1896], mais le prix à payer est lourd, il faut introduire la dimension temporelle dans toutes les relations et dans ce cas, on butte sur le problème du cadre (le frame problem - [CH 1969] J. Mc Carthy and P. J. Hayes). Voyons cela plus en détails.



23 - Représentation du procès qui change l'état du monde

c) La représentation des connaissances en logique

En logique, quand nous faisons de la déduction, nous listons l'ensemble des faits vrais à propos de l'univers. Nous les relions implicitement dans notre tête, par le connecteur logique ET i.e. $\wedge$.

Par exemple, prenons le syllogisme classique : *Tous les hommes sont mortels, Socrate est un homme, donc Socrate est mortel*. Durant ce raisonnement, en partant du fait que *Socrate est un homme*, nous déduisons que *Socrate est mortel*. Après cette déduction, nous obtenons les deux faits suivants :

homme(Socrate); | Ici, le connecteur ET n'est pas matérialisé,
mortel(Socrate); | Il est donc est implicite

Ce qui correspond en logique à la formule :

homme(Socrate) $\wedge$ *mortel(Socrate);* | Ici, nous avons écrit explicitement le connecteur ET.

d) Les problèmes de la représentation des actions en logique

Dans le cadre de la logique, il est possible de simuler des actions. Dans la partie *Difficultés induites par notre problématique*, nous détaillons les obstacles à franchir pour arriver à nos buts : le problème d'inconsistance, le problème du cadre i.e. le frame problem.

3.D) RC dynamique : la solution des systèmes formels

La solution des systèmes formels à règles de réécriture ou de production [Hof 1985].

Les systèmes formels, munis de règles de réécriture ou de règles de production, amènent une solution à notre problème de RC dynamique, car ils permettent de décrire les changements d'état du monde, sans subir le problème du cadre (frame problem).

Deux exemples de systèmes formels : l'automate à états finis (AF) et l'agent situé.

Dans notre partie théorique, nous montrons comment effectuer de la simulation avec le premier de tous les systèmes formels, qui pour nous est l'AF. Puis nous montrerons comment le transformer en agent situé.

3.E) Les simulateurs, maintenant nous allons focaliser sur les simulateurs

Maintenant nous allons focaliser sur les simulateurs car ils répondent à nos attentes. Ils sont fonctionnels et réalisés au moyen de systèmes formels [Cho 1956] : avec eux nous pourrons simuler des actions.

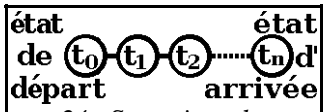
Ch. 6 Synoptique général de différents simulateurs

Note introductive : nous écrivons à la fois un état de l'art des simulateurs et leur histoire de l'art.

Dans ce chapitre, au lieu de faire un strict état de l'art sur les différents simulateurs, nous avons cru bon de retracer leur évolution au fil du temps. Donc notre travail n'est pas seulement un état de l'art, mais constitue aussi une histoire de l'art de la simulation.

Nous présentons la démarche de simulation de l'évolution du monde .

Tout d'abord, avant de travailler sur les simulateurs, nous présentons leur mode de fonctionnement simplifié, basé sur les systèmes formels [Gil 1962] [Hof 1985] :



24 - Synoptique de la simulation

Soit au temps t_0, un monde M, dans l'état E_0, *simuler* son évolution au fil du temps consiste à calculer et décrire les états E_1, E_2... E_n qu'il prendra aux temps t_1, t_2,... t_n.

Dans les simulateurs qui font intervenir un agent, le calcul de l'état au temps t_{n+1} en fonction de celui au temps t_n comporte deux étapes :

- Au départ, avant l'action, l'état du monde est connu, alors on calcule l'action que l'agent va effectuer, en fonction de cet *état avant.*
- Quand l'agent a décidé l'action à effectuer, alors il l'effectue : l'ordinateur calcule la réponse du monde à l'action de l'agent, i.e. il détermine l'état d'arrivée, qui devient le nouvel état du monde.

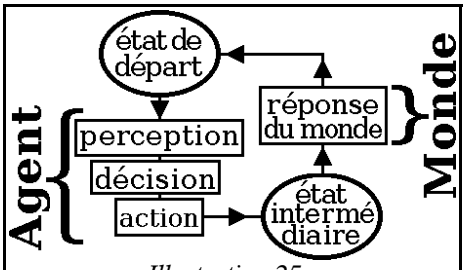


Illustration 25:

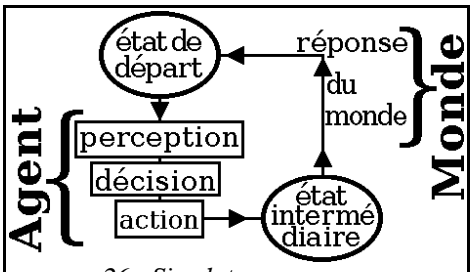
1) Fonctionnement des simulateurs simplifiés ou dissymétriques

Afin de mieux cerner le fonctionnement des simulateurs, nous regardons un peu les différents modèles existants. En les classifiant nous comprenons mieux leur synoptique de fonctionnement. Dans ce paragraphe nous regardons les simulateurs dissymétriques, i.e. ceux dont une fonction à disparu.

1.A) Les simulateurs basés uniquement sur le *choix de l'action*

Tout d'abord, nous focalisons sur la prise de décision et traitons en premier les simulateurs basés uniquement sur le *choix de l'action.*

En classifiant les simulateurs, nous en trouvons qui sont constitués uniquement d'une fonction *choisissant la transformation* à effectuer. Chez eux, la partie réponse du monde est occultée. Ainsi il ressort de ce fonctionnement que, à chaque pas de simulation, le traitement se réduit à une étape unique : *choix de l'action.*

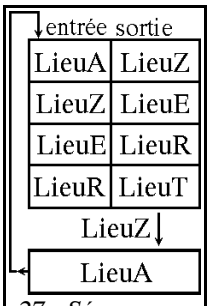


26 - Simulateur avec une réponse du monde occultée

Le séquenceur est le simulateur le plus simple.

Au sein du séquenceur, la prise de décision se fait au moyen d'un tableau à un index. Elle consiste à choisir le prochain état du système en fonction d'un unique paramètre : l'état actuel de l'automate.

Par contre, la réponse du monde à l'*action* de l'AF est totalement occultée. En effet, elle consiste simplement en un changement d'état du système, et il est déterministe et inéluctable. Selon ce mode de fonctionnement, il n'est pas prévu que l'automate fini donne son avis sur l' action à effectuer : il change d'état, un point c'est tout.

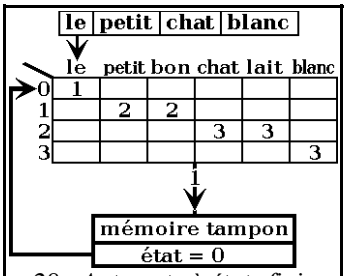


27 - Séquenceur

L'automate à état finis (AF) [Cho 1956].

Au sein de l'AF, la prise de décision effectuée par l'agent est clairement visible car elle se fait au moyen d'une table à deux entrées. Elle consiste à choisir le prochain état du système en fonction de deux paramètres : l'état actuel de l'automate et le premier symbole de son ruban d'entrée.

De même, comme pour le séquenceur, la réponse du monde à son action est totalement occultée.



28 - Automate à états finis

Conclusion : Deux exemples de simulateurs basés uniquement sur le *choix de l'action.*

Nous venons d'exhiber deux exemples où la réponse du monde à l'action de l'être est totalement occultée : il agit et le changement se fait, sans que le moindre incident soit envisagé dans le déroulement de l'action.

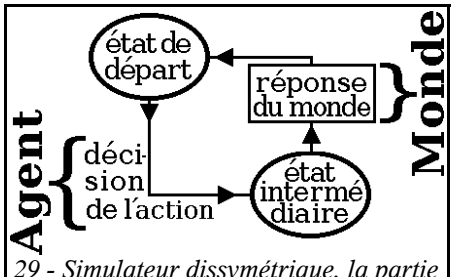
1.B) Les simulateurs basés uniquement sur la réponse du support

En classifiant les simulateurs, nous en trouvons qui sont constitués uniquement d'une fonction déterministe qui transforme le monde. Dans leur synoptique, la partie *décision de l'action* est atrophiée ou inexistante car il effectue toujours le même traitement. Par exemple, les deux simulateurs que nous évoquons ci-dessous effectuent un traitement mathématique (équation aux dérivées partielles).

La rencontre avec ce type de plate-forme constitue pour nous l'occasion de réaliser qu'au sein d'un simulateur, existe une fonction *réponse du support*. Ainsi, lors de l'exécution classique d'un pas de simulation, après l'étape *choix de l'action* se déroule ensuite une étape qui change l'état du système en fonction du changement choisi.

Méthodes des différence finies [Pis 1980].

Dans ce travail, nous choisissons de ne pas distinguer le fonctionnement de la méthodes des différences finies de celle des éléments finis. Le lecteur est donc invité à passer au paragraphe suivant qui traite seulement ce dernier sujet.



29 - Simulateur dissymétrique, la partie décision de l'action est inexistante.

Modélisation par éléments finis [ZT 1991].

Dans les simulateurs de systèmes physiques la décision de l'action de l'agent disparaît totalement : le système évolue tout seul, guidé de façon déterministe seulement par les lois mathématiques qui le caractérisent.

Donc, nous focalisons seulement sur la réponse du support, et en l'occurrence sur celle du système physique, qui est calculée au moyen de l'outil mathématique (méthodes des différence finies, modélisation par éléments finis).

Conclusion : Deux exemples de simulateurs basés uniquement sur l'*exécution de l'action.*

Le gros intérêt de la théorie des modèles, c'est qu'elle ne met pas en scène d'agent, elle court-circuite le calcul de la décision de l'action, pour mettre en avant le concept de réponse du monde, qui devient ici réponse du système différentiel.

2) Les simulateurs de jeux digitaux : ils sont partiellement dissymétriques

Après les simulateurs totalement dissymétriques, nous étudions maintenant des simulateurs qui sont seulement partiellement dissymétriques. Nous focalisons sur l'exemple des jeux digitaux[Hau 1985] [Hau 1989], chez qui la partie *action de l'agent* prédomine sur la *réponse du support*.

Recherche de l'action à exécuter par l'agent.

Dans les jeux digitaux, l'action à exécuter est calculée par chaque joueur qui doit choisir entre différentes solutions. Donc son élaboration peut être complexe. Par exemple, dans une partie d'échecs par correspondance, elle peut durer des jours ou des semaines.

Calcul de la réponse du support.

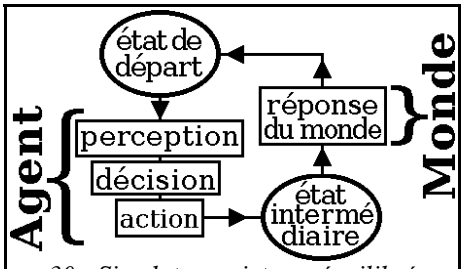
Par contre l'élaboration de la réponse du support est déterministe : elle calculée seulement en fonction des règles du jeu, alors ce traitement est effectué, soit par le joueur, soit par le meneur de jeu, soit par l'adversaire. Ainsi, cette opération est facile, alors elle est presque passée sous silence (sauf quand les joueurs se disputent !).

En conclusion, dans les jeux digitaux, nous constatons une focalisation sur le choix de l' action, au détriment de la réponse du support.

3) Simulateurs mixtes/équilibrés munis d'une représentation symbolique

3.A) Introduction : travailler avec de simulateurs mixtes/équilibrés

Maintenant, le moment est venu pour nous de travailler avec des simulateurs mixtes, ou encore équilibrés, i.e. avec, d'un côté, une simulation de la prise de décision de l'action à effectuer par l'agent, et avec, de l'autre, une simulation de la réponse du monde, quand il est sollicité par cette action effectuée par l'agent.



30 - Simulateur mixte ou équilibré

Ci-contre est illustré un *simulateur équilibré* possédant une structure mixte qui se dédouble en deux parties : la première choisit, calcule l'action à effectuer ; la seconde interprète cette action à exécuter, et transforme le monde selon ses caractéristiques propres.

Les simulateurs mixtes utilisent plutôt une représentation symbolique de l'information.

En effet, ils attribuent une place à la décision de l'action. Cette prise de décision complexe est facilement réalisée au moyen de l'ordinateur qui est symbolique. À la fin de ce chapitre, dans le paragraphe *Au fil de l'histoire, une opposition symbolique vs neuronal* nous racontons comment après 1945 la représentation symbolique à remplacé le traitement analogique. En 1954, à la naissance de l'IA, tous les membres fondateurs travaillaient en symbolique. Cette représentation a dominé le champ de l'IA, jusqu'aux années 80, où est réapparu la représentation connexionniste.

3.B) Deux domaines qui peinent à faire fonctionner des simulateurs équilibrés

En logique mathématique : simuler au moyen de la logique [Car 1886] [Car 1896].

Dans la partie *problèmes induits*, nous décrivons la dérivation au moyen du modus-pones qui permet de simuler au moyen de la logique formelle. Nous expliquons pourquoi elle est limitée par l'inconvénient majeur nommé le *frame problem* ou *problème du cadre* [CH 1969]. Puis nous

précisons comment cet inconvénient peut être résolu au moyen des systèmes formels qui permettent de simuler au moyen des règles de réécritures.

En informatique : simuler au moyen de l'IA symbolique.

Historiquement, on trouve d'abord des simulateurs mixtes basés sur la mouvance des Soar impulsée par Minsky. Elle s'attache à simuler au moyen de la représentation et du traitement symbolique de l'information, mais elle butte sur le *qualification problem* [Car 1959], le problème de *l'inscrutabilité de l'identité radicale* que nous traitons aussi dans la partie *problèmes induits.*

3.C) Conclusion : d'autres techniques pour réaliser des simulateurs équilibrés

Puisque la logique mathématique et l'IA symbolique peinent à faire fonctionner des simulateurs équilibrés, maintenant tournons-nous vers d'autres techniques pour les réaliser.

Ch. 7 Six simulations, du programme à l'agent situé sub-symbolique

Introduction : Nous analysons qualitativement l'évolution des techniques de programmation.
Nous observons l'évolution des systèmes de traitement de l'information qui vont nous servir à implémenter des simulateurs afin d'analyser des évolutions qualitatives.

Six autres techniques pour réaliser des simulateurs équilibrés.
Puisque la logique mathématique et l'IA symbolique peinent à faire fonctionner des simulateurs équilibrés, maintenant passons en revue d'autres techniques, afin d'évaluer leur intérêt :
- La programmation classique (structurée ou non) [Ric 1984].
- Les langages objets [Fer 1989],
- Les agents [Fer 1995], [Gue 2003], [Car 2004],
- Les agents situés [Fer 1995], [PLP 2002],
- Les agents situés sub-symboliques.
- Les vrais robots.

1) La simulation au moyen des langages de programmation classique

Dans l'histoire de la programmation en informatique, nous délaissons les programmes câblés, ceux en langage machine ou en assembleur, et nous en venons aux programmes en langage macroscopique (de haut niveau).

a) Système informatique implémenté en programmation classique

Ici nous décrivons un système de traitement de l'information implémenté au moyen d'un langage de programmation classique. Il peut être structuré ou non, déclaratif ou procédural (Basic [Ric 1984], Lisp [Wer 1985], C [KR 1988] [BG 1988], Pascal).

L'état du système et sa mémorisation.
Quand le système est initialisé, il reçoit des données, i.e. la description de son état, qui est enregistré dans une mémoire.

Définition d'un procès.
Par définition, le procès qu'effectue le système correspond au traitement qu'il pratique sur ses données.

Définition du code d'un programme.
Le code qui décrit le procès à effectuer, correspond au traitement à pratiquer sur les données du système : c'est la liste des instructions à exécuter pour transformer l'état de ce dernier.
De par sa nature, un langage de programmation est conçu pour implémenter un procès, i.e. pour écrire la liste des instructions qui décrivent le procès à effectuer.

L'exécution d'un code effectue un procès.
Puisque le code d'un programme décrit un traitement, un procès, le fait d'exécuter ses instructions effectue le traitement, réalise le procès correspondant.

Centralisation du code et des données.
Dans un langage de programmation classique, le code et les données sont centralisés, i.e. rangés au sein du même système. Et quand le code et les données de certaines applications deviennent conséquents, même si cette concentration pose problème, les adeptes de la programmation classique continuent de coder ainsi.
Regardons maintenant deux exemples de programmation classique :

b) Exemple 1 : simulation implémentée en programmation classique : le séquenceur

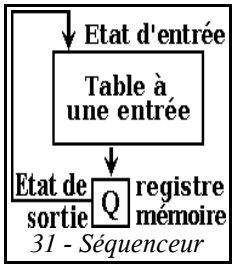
L'état du séquenceur correspond à la sortie du séquenceur. Il est donc enregistré dans son registre mémoire.

Le procès effectué par le système consiste en un changement d'état.

Le code du système correspond à la programmation de la table.

L'exécution d'une des adresses de la table correspond à la réalisation d'un procès élémentaire.

La table qui contient le code et le registre qui mémorise l'état sont centralisés au sein de l'automate.

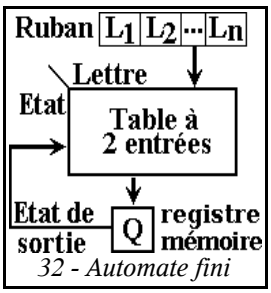


31 - Séquenceur

c) Exemple 2 : simulation implémentée en programmation classique : l'automate fini

Globalement le synoptique de l'AF découle de celui du séquenceur :
- L'état de l'AF correspond à la sortie de son registre mémoire.
- Le procès effectué par le système consiste en un changement d'état.
- Le code du système correspond à la programmation de la table.
- Exécuter une des adresses de la table réalise un procès élémentaire.
- La table et le registre mémoire sont centralisés au sein de l'AF.

Mais nous devons signaler une différence : le séquenceur est solipsiste : il est fermé à l'extérieur et ne regarde que son nombril. Ceci se voit à l'adressage de sa table : elle ne reçoit qu'une entrée, la sortie du registre.



32 - Automate fini

En conclusion, l'AF sort de sa coquille et s'ouvre au monde en lisant un ruban externe. Maintenant sa table reçoit deux entrées, il agit donc en fonction de l'extérieur : son action est devenue causale.

d) Transition

Par la suite, la programmation classique se démode. Dans le paragraphe suivant, nous décrivons langages à objets, avec eux, quand un système grandit, il se décentralise, i.e. il se sépare en plusieurs petites applications, où chacune possède son propre code et ses données personnelles.

2) Simulation au moyen des langages à objets (LO) [Leb 1991], [SB 2006]

2.A) Les LO sont basés sur deux idées centrales : décentralisation et héritage

La décentralisation du code et des données.

Cette démarche consiste à décentraliser code et données. Alors qu'en programmation classique (Basic, Lisp, C, Pascal) un programme est monolithique, il regroupe dans un même bloc le code et les données, la programmation objet (PO), qui est très utile quand on traite une quantité d'êtres indépendants, distribue code et données au sein de chaque objet. Le fait d'utiliser dans le programme cette organisation répartie qui mime celle de la vie réelle, facile le travail du développeur.

L'héritage des structures et du code.

Puisque la spécificité des LO est de mettre en scène beaucoup d'agents, ils ne peuvent être tous différents, donc ils se ressemblent parfois ou souvent. Dans ce dernier cas, les LO fournissent des méthodes d'héritage qui facilitent la description des êtres traités.

Nous devons préciser que dans les simulations suivantes qui animent des agents (situés ou non) l'implémentation de cette démarche est facilitée par la mise en situation et par la vie artificielle. En effet l'enfant qui naît ressemble à son parent, et il hérite naturellement de ses caractéristiques physiques (position, direction...).

2.B) La PO apparaît comme une solution au *qualification problem*

a) Elle décrit le fonctionnement de chaque objet en le basant sur ses composants

Elle décrit chaque objet en l'expansant, i.e. en les représentant sous la forme d'un ensemble de composants. Ainsi, elle donne du corps, de l'amplitude, de l'épaisseur ontologique aux représentations des objets et à leur fonctionnement.

Holisme et épiphénomène [Gan 2006].

Avec les LO, l'utilisateur peut plus facilement mettre les objets en relation et faire apparaître leurs comportements macroscopiques comme des épiphénomènes de leurs interactions microscopiques. Ainsi, lors de l'exécution l'informaticien obtient un comportement holistique du programme. Nous remarquons que le tout est plus que la somme des parties. En effet, au fonctionnement individuel que nous avons programmé chez chaque objet, apparaissent des épiphénomènes, qui viennent se superposer au reste.

La trichotomie décompose le fonctionnement des objets sur trois niveaux imbriqués.

Dans le cadre de cette démarche analytique, les chercheurs (David Marr, Alan Newel puis Daniel Dennett) sont amenés à présenter leur trichotomie, qui décompose le fonctionnement des objets sur trois niveaux imbriqués. Au plus bas niveau de cette empilage de couche, i.e. au niveau de l'implémentation sur le substrat (selon David Marr [Mar 1982]), ces scientifiques représentent les formes élémentaires, les composants atomiques du domaine : le matériel pour Newel [New 1990] et le composant pour Denett [Den 1987].

b) Conclusion : la stratification représente une solution au problème de l'identité

En décrivant les objets en sous-parties, puis en assemblant le tout, à un niveau macroscopique nous construisons un être holistique et récupérons, aux niveaux plus macroscopiques, des épiphénomènes. Ainsi nous pouvons modéliser plus facilement le fonctionnement des êtres riches. Alors la construction des objets en couches stratifiées nous aide à les doter d'un comportement complexe, et constitue une solution au problème de l'identité.

2.C) Déconstruire les objets vers le substrat

La déconstruction des objets en direction, en fonction du substrat se rapproche de la représentation neuronale. Ainsi la représentation des connaissances des LO peut être symbolique comme neuronale, mais elle quand elle prend un démarche analytique, quand elle déconstruit les objets en sous-parties, elle descend vers le substrat et commence à se rapprocher de la représentation neuronale [Smo 1992].

3) Système multi-agents (SMA) : une simulation au moyen des agents

3.A) Introduction classique des agents

Afin d'obtenir une introduction classique à propos des SMA, le lecteur se reportera au livre très complet de Jacques Ferber [Fer 1995].

3.B) Incarnation, mise en situation dans le monde

a) Incarner, mettre en situation, la démarche pour étudier un phénomène langagier.

Dans notre marche vers la simulation/modélisation des phénomènes langagiers, d'abord nous avons montré comment les systèmes formels apportent un outil pour représenter dynamiquement les transformations du système.

Ensuite nous avons montré comment les LO permettent de décentraliser l'information à propos du système vers de multiples objets, et de les décrire au moyen de l'héritage.

Maintenant regardons dans l'histoire de l'IA quelques unes des simulations qui prennent un système formel et l'incarnent dans le monde. Pour ce faire, nous présentons ici, deux réalisations afin d'introduire les agents, puis les agents situés.

b) SHRDLU de Terry Winograd (1968-1970) [Win 1972]

À propos de cette simulation nous focaliserons sur trois aspects notables : le changement de l'état du monde, le traitement du langage naturel, et la notion de *situation dans le monde* (sur un plateau).

Le changement de l'état du monde.

Le robot est placé sur un plateau, au milieu de formes géométriques : cube, pyramides, parallélépipèdes. Il agit, et change l'état du monde en exécutant des actions : d'abord il modifie les relations entre le mobilier et le support, puis les relations entre certains éléments du mobilier.

Le traitement du langage naturel.

SHRDLU est commandé en langage naturel. Il reçoit des énoncés qu'il analyse syntaxiquement, qu'il interprète sémantiquement : il effectue les procès qui lui sont commandés et il change l'ordre du monde,

Introduction de la notion de mise en situation[9]

Le robot est mis en situation sur le plateau. Il bouge les volumes qu'on lui désigne, en les caractérisant par leurs position ou situations relatives à lui-même ou relatives aux autres volumes.

c) Margie : la théorie de la dépendance conceptuelle de Schank et Abelson [SA 1977]

Dans Margie de Schank (1974), nous retrouvons un peu les mêmes aspects. L'action demeure basée sur la variation de relations relatives aux autres ou aux lieux, mais elle peut aussi induire des variations absolues, i.e. des grandeurs physiques (i.e. de niveau d'énergie et de santé).

Mais cependant nous notons quelques différences : les agents ne sont pas situés. Ainsi le programmeur perd la simplicité de la représentation en situation. Alors le monde décrit est bien plus formel et la représentation des agents est plus riche. Ce travail marque la limite supérieure de la direction vers laquelle nous progressons.

9 Introduction de la mise en situation : cette notion sert à introduire les agents situés qui sont présentés au paragraphe suivant.

3.C) Introduction des agents, tournée vers les phénomènes langagiers, cognitifs

En ce qui nous concerne, nous procéderons à une introduction des agents, orientée vers notre centre d'intérêt : les phénomènes langagiers et cognitifs :

Plonger l'agent dans la matière du monde[10].

Dans notre partie théorique, nous montrons comment le passage des LOO aux SMA se fait en incarnant chaque objet dans un agent. Pour ce faire, nous choisissons d'introduire l'agent comme la matérialisation d'un objet, i.e. comme le fait de lui donner chair, de le concrétiser au sein d'un corps. À l'issue de cette incarnation, le système devient le monde composé d'un support et de son mobilier : les agents, les pièces de nourriture et les obstacles.

Deux Remarques :

Chacun des symboles du domaine de l'automate fini s'incarne en un des lieux du monde, où évoluent les agents. Le agents font parti du mobilier. Il est aussi constitué des pièces de matière et des obstacles, qui apparaissent comme des agents aux capacités restreintes.

L'état du système d'objets devient l'ensemble des relations du SMA.

À l'issue de cette incarnation, ce qui était l'état du système devient l'ensemble des relations *agent* ↔ *agent* ou *agent* ↔ *support*.

Introduction de l'action.

Ensuite pour définir l'action, d'abord nous introduisons les relations entre l'agent et le monde, ensuite nous faisons varier ces relations au moyen du procès, et enfin, pour donner un sens à ce dernier, nous le sous-tendons par un but. Finalement, au terme de cette construction, nous obtenons un agent : par définition, il effectue une action quand il change l'ordre du monde afin de satisfaire un but.

10 En fait, ici je paraphrase un peu Teilhard de Chardin.

4) Simuler, modéliser l'action de l'agent situé, qui change l'ordre du monde

4.A) Le concept d'agent situé [Kae 2005]

a) Introduction : de l'automate fini à l'agent situé

Les personnes qui étudient les phénomènes langagiers sentent confusément qu'ils doivent simuler un agent en situation. Ceci consiste en deux aspects : nous avons déjà vu l'incarnation de l'objet, qui a donné l'agent. Il nous reste à mettre ce dernier en situation, afin d'obtenir un agent situé.

Dans le paragraphe précédent, nous avons incarné l'être, l'objet, dans la matière. Il nous reste à le mettre en condition, en situation dans le monde.

Mais en fait, dans l'histoire de l'informatique, le concept d'agent situé arrive tardivement. Ainsi, selon un point de vue chronologique, le concept d'agent situé est venu plus tard. Nous traiterons cet aspect quand nous parlerons des langages.

b) Les trois composantes de l'agent situé : incarnation, mise en situation, décision

Nous avons déjà prolongé les objets des LO en incarnant l'abstrait dans le réel.

Les SMA viennent après la programmation orientée objet, dont ils conservent la démarche de décentralisation du code et des données. Ensuite, ils la prolongent au moyen d'une incarnation du domaine dans la matière. Par exemple, l'incarnation des symboles du domaine mathématique de l'objet se concrétise en un support de plusieurs lieux.

Mettre l'agent en situation dans le monde.

Ensuite, nous utilisons le parallélisme des LO, afin de disposer du mobilier sur ce support ; par exemple : des êtres sont placés au milieu de nourriture et d'obstacles. Cette simulation se modélise en décrivant un système qui anime en parallèle, quelques êtres omnipotents pour fournir des agents, et des êtres atrophiés qui modélisent une pièce de matière pour chaque obstacle ou chaque nourriture. En effet, les agents agissent, mais une nourriture ne se déplace pas ; cependant elle peut être mangée, dans ce cas, elle repousse progressivement.

Ainsi nous obtenons un agent situé : l'agent se trouve en situation sur le support de lieu, en relation avec d'autres agents, et au milieu de pièces de matières (obstacles et nourritures). Mais dans ce cas, pour des raisons physiques, naturelles, il est surtout en relation avec ses plus proches voisins.

À chaque instant l'agent situé prend sa décision au moyen d'une fonction.

En situation dans le monde, grâce à ses senseurs, l'agent perçoit son environnement et il agit en conséquence. La prise de décision de l'agent situé est fonctionnaliste. Elle est effectuée par une table ou par un jeu de règles.

Le fait que naturellement l'agent soit surtout en relation avec son environnement local entraîne qu'il ne prend pas ses décisions en toute connaissance de cause. Dans ce cas, il cesse d'être un tacticien et devient un stratège. Ainsi, nous passons de l'absolu des mathématiques à l'empirisme du langage naturel et de l'homme de la rue.

4.B) Détailler la formalisation des AS : 2 mots clés dérivant de la *localité*

Au départ, avec les langages objets nous disposons déjà d'êtres abstraits qui subissent un procès. Quand nous introduisons les agents, nous matérialisons ces êtres qui alors exécutent des actions.

Afin d'obtenir l'agent situé, nous ajoutons un cadre local dans lequel il prend des décisions à courte vue et change l'ordre de ce micro-monde.

a) Le support à voisinage (effet d'horizon sur la nature du support)

L'agent situé est placé sur un support régulier : chaque lieu l_0 est seulement en relation avec quelques autres l_i. Ce sont ses plus proches voisins au deux sens du terme. Ils sont voisins parce qu'en relation $R(l_0, l_i)$ et aussi car ils sont physiquement proches.

Par exemple, les supports les plus utilisés sont ceux à 2 voisins (cas d'un domaine en dimension 1), 4 voisins (cas classique d'un domaine en dimension 2 très régulier), et parfois 6 voisins (domaine en nid d'abeille).

b) L'effet d'horizon : l'agent situé perçoit seulement son voisinage

Maintenant l'agent est situé dans un cadre physique local, et sa perception est limitée à un horizon proche : c'est l'effet d'horizon : Par exemple, dans nos modélisations, quand l'agent est sur un lieu l_0 alors il ne perçoit que les lieux l_i tels que $R(l_0, l_i)$.

c) Le raisonnement à profondeur limitée

Le raisonnement est contextuel à cause d'une perception limitée.

Puisque la perception de l'agent est limitée par l'effet d'horizon, son raisonnement s'effectue au moyen d'une connaissance du monde réduite, limité à un contexte : c'est le raisonnement contextuel de la *rationalité limitée* de Herbert Simon [Sim 1947].

Le raisonnement est contextuel à cause d'un traitement de l'information, sommaire mais suffisant

Dans la simulation *ApprendRègles*, commentée dans la partie pratique, nous effectuons un test minimal avant de déclencher une règle : l'observation d'un percept active immédiatement un actionneur. Mais il se trouve que ce traitement sommaire de l'information, effectué à cette profondeur très limitée [Kay 1992], s'avère suffisant pour piloter le comportement recherché.

4.C) Les outils pour moduler localement un raisonnement

a) Introduction : moduler localement un raisonnement

L'agent en situation sur son support, dispose d'une perception limitée de son environnement, et pourtant il doit traiter cette seule information afin d'agir. Il se trouve donc dans une situation où il effectue un raisonnement modulé localement.

b) La logique modale des mathématiques

Introduction.

En mathématiques, la logique modale permet de moduler un raisonnement en fonction des propriétés du monde où il est effectué. Elle découle de la théorie de la sémantique des mondes possibles introduite par Hintikka [Hin 1989].

Le contexte local de l'évaluation devient ici le monde particulier où s'effectue le traitement.

Dans ce cadre, l'utilisateur dispose d'un outil pour construire son univers en assemblant plusieurs mondes. Il peut moduler la réponse à l'évaluation de la formule en fonction des spécifications qu'il lui attribue. Donc cette évaluation est contextuelle, elle se fait en fonction de l'univers, selon les relations définies entre les mondes, i.e. en fonction de ceux qui sont visibles, accessibles, dans la situation donnée.

Le raisonnement causal.

Dans cet univers mathématique, l'utilisateur conserve la logique classique qui constitue un outil pour raisonner. Avec elle, il peut simuler et modéliser le raisonnement de l'agent qui décide l'action, au moyen des fonctions logiques booléennes, qui sont bien connues pour être adaptées à cela.

Cependant, il reste que la logique modale est du premier ordre : elle utilise les variables. Certes elle permet des raisonnements généraux, mais les démonstrations, nécessairement effectuées en symbolique, sont lourdes. Cette lourdeur de la logique du premier ordre est encore plus perceptible quand il faut faire de l'apprentissage par généralisation. Dans ces deux notes Plotkin [Plo 1970] [Plo 1970] montre comment généraliser des clauses. Nous, qui avons choisi de présenter une plate-forme simple, ne pouvons pas travailler en symbolique sur ce sujet. C'est pourquoi,

dans *ApprendRègle* de notre partie pratique, nous montrons comment nous généralisons des règles au moyen des agents situés et d'une représentation traitée en sub-symbolique[11].

c) Les agents situés de l'informatique

Introduction.

À l'instar de la logique modale, les agents situés de l'informatique permettent aussi un raisonnement modal : chaque agent adapte son comportement en fonction des propriétés locales de l'environnement où il se trouve.

Le contexte local de l'agent situé devient ici l'environnement ponctuel où l'agent agit.
Dans ce cadre, le programmeur définit l'architecture du monde à sa guise. De même il définit sa réponse à l'action des agents. Alors l'évaluation de l'action est contextuelle, elle se fait en fonction d'un rayon qui définit la porté des relations entre les objets du monde, i.e. en fonction de ceux qui sont visibles dans la situation donnée.

Une plate-forme d'enseignement pour simuler un raisonnement causal facile à implémenter.

En conclusion, nous présentons une plate-forme d'enseignement qui permet de simuler un raisonnement causal facile à implémenter. Dans ce cadre de l'agent situé (sub-symbolique), le chercheur en sciences humaines et sociales peut programmer un comportement causal d'un être, au moyen de règles ou de programmes, et cette simulation, très naturelle apparaît plus simple que celles des mathématiques ou celle de la simulation par LO.

11 Les agents situés sub-symboliques sont traités au paragraphe suivant.

5) Simuler au moyen d'agents munis d'un traitement sub-symbolique

Après avoir introduit cette notion centrale d'agent situé sur un support, avant de regarder comment le modéliser pour qu'il traite ses perceptions, au moyen de la représentation sub-symbolique, nous faisons un rappel historique sur la longue opposition s*ymbolique vs neuronal*.

5.A) Au fil de l'histoire, une opposition s*ymbolique vs neuronal*

Afin de faire un état de l'art sur ce sujet, il est bien difficile de dégager une loi qui résumerait l'opposition s*ymbolique vs neuronal* au fil de l'histoire, car on observe un constant chassé-croisé entre les deux mouvances.

Cependant, nous résumerons l'après-guerre en disant que le neuronal, longtemps éclipsé par l'ordinateur de John von Neumann [Neu 1996], revient en force naturellement parce que les deux mouvances sont complémentaires.

Ensuite, nous pronostiquerons que le neuronal va encore progresser, d'abord car notre but final est d'expliciter le fonctionnement de la cognition, puis en s'appuyant sur les grands progrès technologiques d'investigation du cerveau (scanner, RMN, TEP).

a) À l'origine le traitement et la représentation des connaissances sont analogiques

Le langage parlé : au début il est plutôt analogique.

Assurément, au sein du cerveau de l'homme le traitement et la représentation des connaissances sont analogiques. Au début, le langage parlé est plutôt analogique : il constitue une mouvance amorphe, changeante, aux contours mal délimités. Ensuite, quand il est formalisé par les grammairiens, il reçoit un lexique et une grammaire : il devient formel.

b) Puis vient l'approche formelle de l'arithmétique et des mathématiques

Généralement, dans leurs débuts, l'arithmétique et des mathématiques relèvent plutôt de représentations et traitements formels.

- La logique Aristotélicienne et son syllogisme sont déjà formels (Aristote -384 -322).
- L'abbaye de Port-Royal des champs (1204 – début du 18ième siècle) diffuse sa logique qui prolonge celle de Aristote.
- En 1642, elle voit naître la Pascaline, machine formelle à additionner, construite par B. Pascal.
- En 1703, Gottfried Wilhelm Leibniz Paris, dans un mémoire à l'académie des sciences explique l'arithmétique binaire.
- En 1854, avec son algèbre, Georges Boole introduit la logique formelle [Boo 1854].
- Charles Babbage (1791-1871) avec ses machine à différences et analytique effectue des traitements formels.
- Augustus De Morgan (1806-1871), travaille sur la logique formelle [Mor 1847].
- Puis Lewis Carroll (Charles Dodgson 1832-1898) prolonge ce travail [Car 1886], [Car 1896].
- Gottlob Frege (1848-1925) crée la logique moderne (calcul propositionnel, le calcul des prédicats) [Fre 1923].
- William Russell, Alfr. Whitehead, *Principia Mathematica* [RW 1910] tentent de refonder les mathématiques sur la logique.

c) Le traitement analogique de l'information culmine au milieu du XXième siècle

- Déjà, dans l'antiquité les méditerranéens traitaient une information analogique (astrolabe, calculateur analogique).
- Le moyen-age voit apparaître des planisphères et horloges (d'abord analogiques).
- En 1614 John Neper invente les logarithmes, bases mathématiques des règles à calcul (Edmond Wingate, 1627).
- En 1876 James Thomson invente l'analyseur différentiel, un ordinateur analogique mécanique pour résoudre les équations différentielle par intégration mécanique (les premiers exemplaires furent construit dans les années 1920, 1930).
- 1906, Lee De Forest invente la triode qui marque la naissance de l'électronique (traitement analogique de l'information).

- Pendant la première guerre mondiale, les marines Russes et anglaises disposent déjà de canons asservis par électronique.
- Suite à son travail pendant la seconde guerre mondiale sur l'asservissement des batteries anti-aérienne, en 1948 Norbert Wiener jette les bases de la cybernétique.

d) Après la guerre 39-45 vient la montée en puissance du traitement logique formel

- Après la seconde guerre mondiale, les calculateurs analogiques dominaient, et on ne pensait pas beaucoup aux ordinateurs digitaux.

Mechanical analog computers were very important in gun fire control in World War II and the Korean War; they were made in significant numbers. In particular, development of transistors made electronic analog computers practical, and before digital computers had developed sufficiently, they were commonly used in science and industry.

Tableau 1: Diffusion des calculateurs analogiques, après 1945 (d'après Wikipédia)

-1949, apparaît l'EDVAC, premier ordinateur doté de l'architecture inventée par von Neumann.
- En 1959 Texas Intrument invente le circuit intégré et 1960 Fairchild le procédé Planar.
- En 1956 fondation de l'IA : c'est une IA symbolique.
- En 1969 dans leur livre Perceptrons Marvin Minsky et Seymour Papert montre que le perceptron linéaire ne peut apprendre la fonction OU exclusif, même si on lui ajoute des couches [MP 1972].

e) Le renouveau des réseaux de neurones des années 1980

Cette charge de Minsky et Papert contre le perceptron va retarder de 10 ans les travaux dans le domaine. Cependant certains ont persisté dans cette direction, et, sous l'impulsion de John Joseph Hopfield [Hop 82], les années 80 voient la résurgence de cette mouvance avec de nouveaux concepts :

- Carte auto adaptative pour une classification non supervisée : Teuvo Kohonen [Koh 1982].
- Perceptron multi-couches de Rumelhart [RDG$^+$ 1986] : rétropropagation du gradient de l'erreur dans un réseau à fonction de transfert non linéaire.

En ce qui concerne notre travail de thèse, signalons que certains chercheurs, soucieux de neuro-mimétisme, émettent un doute sur l'aptitude du modèle de la *rétropropagation du gradient* à modéliser le fonctionnement neuronal. D'abord ils avancent qu'on n'y trouve pas de cheminement de l'information à rebours, mais surtout ils font remarquer qu'on y trouve ni le signal à apprendre, ni le signal cible tous deux nécessaires à l'apprentissage.

Ainsi, quand nous présenterons *LearnTrajet,* un des résultats de notre travail, nous soulignerons combien notre modèle hebbien est plus plausible que celui de la *rétropropagation du gradient.*

f) Conclusion : notre travail est à cheval entre les disciplines *symbolique* et *neuronal*

D'un côté notre travail s'attache à modéliser des fonctionnements relevant des sciences humaines (linguistique, psychologie, sociologie, cognition), et de l'autre il tourne sur un ordinateur typiquement von Neumannien !

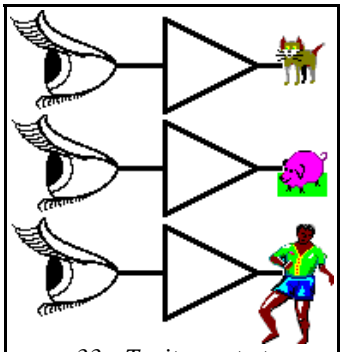
5.B) La description globale de l'état du monde

L'agent situé perçoit le monde, qui, au début se réduit au mobilier sur un support de lieux. Initialement, nous représentons un meuble en situation dans le monde au moyen du couple (meuble, support). Puis, pour représenter un groupe de plusieurs meubles (obstacle, nourriture, agent), nous représentons ce couple initial plusieurs fois en parallèle.

5.C) Modéliser le traitement causal de l'AS, en représentation sub-symbolique

Nous utilisons le cadre des jeux digitaux.

Pour représenter l'agent situé en sub-symbolique, et modéliser son raisonnement causal, de suite nous utilisons le cadre très limitatif des jeux digitaux. Cette restriction simplifie outrancièrement la réponse du support. Ainsi, nous pouvons éviter le problème de l'identité radicale.

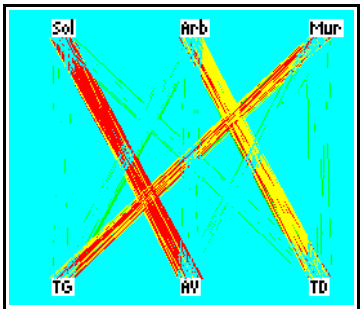


33 - Traitement et représentation sub-symbolique des perceptions

Nous utilisons une perception limitée par un horizon.

Toujours dans le but de simplifier le traitement, nous limitons la perception de l'agent à un horizon de profondeur unité : dans le jeu digital, il ne perçoit que la case devant lui.

En *sub-symbolique* le monde perçu est décrit au moyen de symboles câblés.



34 Jeu de 9 règles de la forme percept→action

Dans ce cadre de cet horizon limité, l'état du monde que l'agent *sub-symbolique* perçoit, peut se limiter à la sortie d'un capteur typé : l'agent connaît le contenu de la case devant lui.

Dans le cadre d'un monde plus riche, l'agent prend sa décision au moyen d'une fonction plus complexe, câblée en sub-symbolique : dans un réseau de neurones, un noeud pondéré sommateur compose et traite plusieurs percepts. Cette démarche correspond à augmenter la profondeur du raisonnement et introduit le raisonnement à profondeur variable décrit plus bas dans ce paragraphe.

Le traitement de l'information à profondeur limitée déclenche une action.

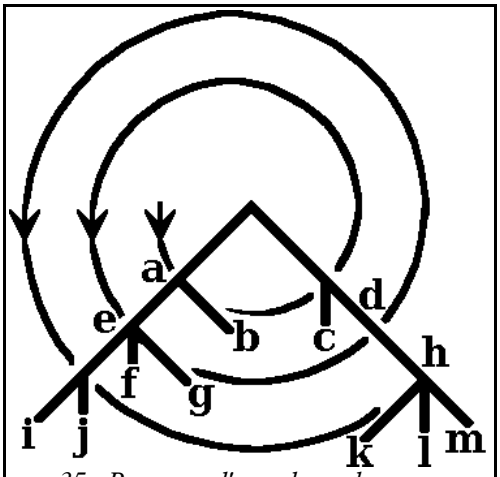
Dans la simulation *ApprendRègles*, commentée dans la partie pratique, nous effectuons un test minimal avant de déclencher une règle : l'observation d'un percept active immédiatement un actionneur. Mais il se trouve que ce traitement sommaire de l'information à cette profondeur très limitée s'avère suffisant pour piloter le comportement recherché.

Le raisonnement à profondeur variable [Kay 1992].

Raisonner à profondeur constante, dans un arbre, consiste à utiliser des règles de la même complexité. Nous avons déjà cité l'exemple de règles de la forme *percept→action* (illustration ci-dessus à gauche).

Le raisonnement à profondeur variable correspond à l'idée de traitement dans un arbre, selon un parcours en largeur d'abord (ci-contre). Il consiste à utiliser un jeu de règles dont la structure est de plus en plus complexe, i.e. dont la profondeur est de plus en plus grande.

Cette stratégie de traitement, Daniel Kayser propose de l'appliquer pour le traitement des données décrivant le monde au moyen du langage naturel. Ici nous l'appliquons à des données reçues par l'étage de perception de notre agent.



35 - Parcours d'un arbre selon une stratégie en largeur d'abord

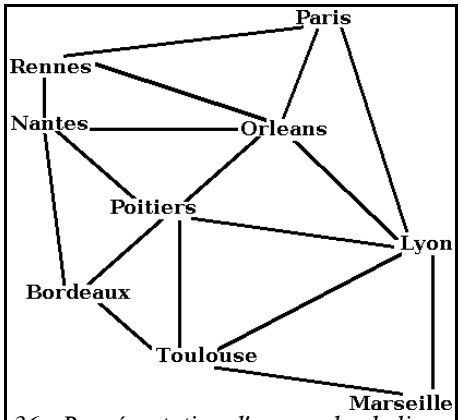
5.D) Raisonnement : l'agent rationnel se représente un monde sub-symbolique

Simuler un agent possédant une représentation sub-symbolique du monde.

Maintenant nous modélisons un être plus complexe qui, dans sa mémoire, représente en sub-symbolique les connaissances de la topologie d'un monde simple. Plus précisément, nous modélisons ce savoir sous la forme d'un graphe de lieux.

Simuler un raisonnement sur un monde représenté par un graphe de lieux.

Nous modélisons un être qui, dans sa tête raisonne sur une représentation du monde effectuée sous la forme d'un graphe de lieux. Il simule son déplacement dans ce graphe.



36 - Représentation d'un graphe de lieux

Une fonction mathématique modélise la prise de décision sub-symbolique.

Dans le cadre de cette représentation sub-symbolique, l'agent possède une direction *d*, il sait sur quelle case *x* du graphe il se situe, et il connaît la nature des cases voisines.

Pour modéliser le raisonnement qu'il effectue dans le cadre de ce traitement sub-symbolique, cette position *x* est mémorisée dans un registre. La prise de décision de l'agent en fonction de sa position *x* et sa direction *d*, peut être modélisée par la fonction mathématique classique $f : (x, d) \rightarrow f(x, d) = x'$; où x' est la prochaine position de l'agent.

Sur le plan informatique, le support du monde est un graphe de lieux, le domaine de la variable *x* est fini car il se situe dans un sous-ensemble des entiers. Alors, en représentation symbolique, le traitement de cette fonction instanciée s'implémente au moyen d'un dictionnaire, plus précisément d'une table à deux entrées. Sinon, en représentation sub-symbolique, il se représente par des pointeurs, et se traite en *tirant* sur celui qui correspond au lieu et à la direction où se trouve l'agent.

5.E) Graphe pour implémenter la <u>réponse du support</u> dans ce contexte local

Puisque nous sommes dans le cadre des agents situés, leur localité, i.e. le fait qu'ils sont situés dans un environnement limité par l'effet d'horizon, s'implémente en *tirant sur les pointeurs d'un graphe*. L'agent est positionné sur un des noeuds, où ils dispose de tous les liens vers les noeuds voisins.

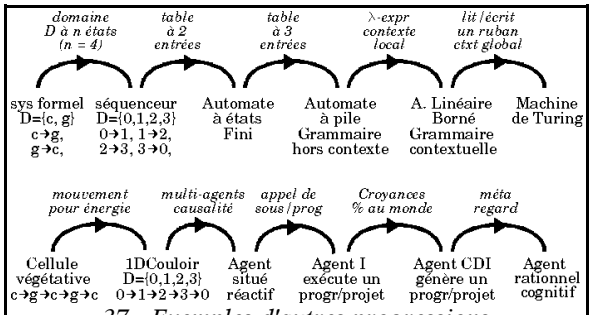
Ainsi, à l'instar de Stanley J. Rosenschein [RK 1986] et Kaeblin [Kae 2005], nous pouvons conclure que la prise de décision effectuée en tirant sur des pointeurs d'un réseau, i.e. de manière fonctionnelle, permet une implémentation <u>sub-symbolique</u> d'un traitement logique. Alors, de cette façon, nous pouvons effectuer un raisonnement causal sans devoir représenter formellement les règles de déduction du calcul des prédicats, i.e. sans utiliser des règles qui porteraient des variables.

En effet, quand l'agent est positionné sur le réseau, implicitement sa variable de lieu est instanciée. Comme il est toujours situé sur le réseau, c'est toujours le cas, donc le traitement mathématique peut s'effectuer au moyen du calcul des propositions, i.e. sans utiliser de variables.

Pour plus de détails sur ce sujet, le lecteur est invité à se reporter au chapitre *LagentCDIPourPlanifier* de la partie théorique. Il commente l'exécutable *Scouter* qui est fourni avec notre progression de simulations.

6) Simuler au moyen de robots placés dans une pièce

À chaque étape de notre progression de simulations, nous avons avancé vers le réalisme : *cellule/séquenceur*, *automate*, *automate à pile/programme*, *objet*, *agent*, *agent situé*, *agent situé sub-symbolique*. Normalement, en continuant cette marche vers le réalisme, nous devrions aller jusqu'à simuler au moyen de robots placés dans une pièce.



37 - Exemples d'autres progressions

D'autres l'ont fait. Nous signalons en particulier Agnès Guillot et Jean-Arcady Meyer [GM 2008], parce que nous reprenons leur démarche de bioniciens qui s'attachent à construire des robots en copiant les inventions de la nature.

Pourtant nous arrêtons en cours de route car nous pensons pouvoir modéliser des phénomènes sociaux et langagiers sans devoir engager des moyens aussi lourds. Finalement, pour viser ce but, nous préférons essayer une approche mathématico-informatique.

7) Quel positionnement pour notre plate-forme ?

Le positionnement sub-symbolique, large et peu profond de notre plate-forme d'enseignement/recherche s'oppose directement au modèle économique des plate-formes symboliques industrielles.

En effet, il semble que les plate-formes industrielles mettant en scène les agents situés [AAH 2009] s'avèrent économiquement viables seulement dans des cadres dédiés, pointus, et spécialisés. Ainsi, sur le site du *Journal of Artificial Societies and Social Simulation*, nous ne trouvons aucune référence à des travaux directement orientés vers une recherche fondamentale sur les phénomènes cognitifs ou langagiers [JAS 2010].

À l'inverse de la démarche industrielle, notre plate-forme sub-symbolique est simple afin de ratisser large. Ainsi elle est peu profonde, légère, et si élémentaire qu'elle pourra modéliser les relations socio-économiques communes. C'est pourquoi elle s'avère intéressante pour l'enseignement et la recherche.

Ch. 8 État de l'art : introduction de la communication

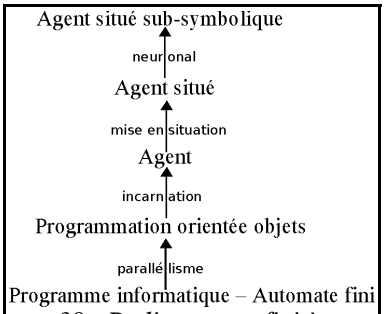
1) Quelles sont les raisons de cette introduction ?

Introduction.

Dans le chapitre précédent, en utilisant le parallélisme, nous introduisons le groupe d'objets. Puis en les incarnant, nous fabriquons des agents, que nous mettons ensuite en situation sur un support : nous avons construit un groupe d'agents situés. En ajoutant la dimension neuronale nous passons à une représentation sub-symbolique.

À ce stade nous sommes arrivés au stade de l'agent situé A6S[12] (2DCueilleSurUnRéseau ou 1DCueilleur de la figure ci-dessous)

Maintenant nous choisissons d'introduire la communication dans notre progression.



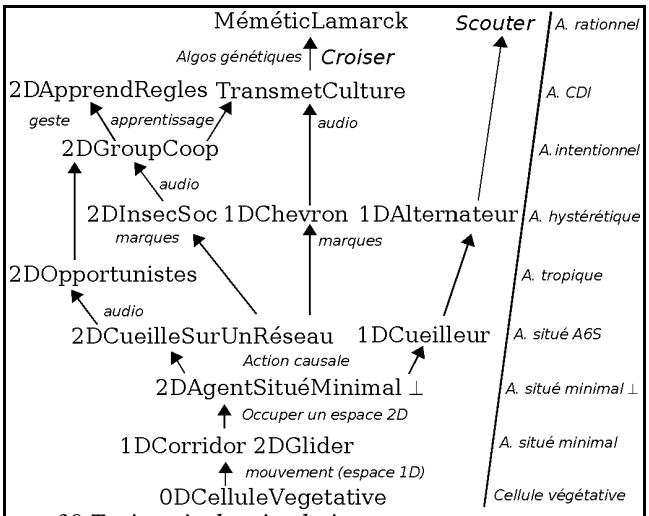
38 - De l'automate fini à l'agent situé sub-symbolique

Une justification par la sémantique.

En fait, l'introduction de cette communication au sein du groupe d'agents est sensée. Elle est justifiée par sa sémantique, i.e. par l'augmentation de l'efficacité qu'elle induit : sa mise en oeuvre améliore le fonctionnement du groupe d'agents, qui assure ainsi sa pérennité.

Rajouter de nouvelles branches à notre nœud.

Pour construire l'arbre ci-contre, nous ne cherchons pas à prolonger la branche de l'agent situé que nous venons de construire, mais, arrivés au nœud des A6S, nous cherchons plutôt à lui ajouter de nouvelles branches. Alors, dans une démarche éclectique nous trouvons l'occasion de nous diversifier vers quatre directions, dont trois types différents de communication.



39 Taxinomie des simulations que nous proposons

Construction de notre progression.

Maintenant nous parcourons cet arbre, afin de relever les simulations et construire notre progression. D'abord, en partant de la cellule de base, nous relevons le tronc jusqu'à la fourche de l'agent situé. Ensuite, nous parcourons les arborescence selon une stratégie *en largeur d'abord*.

Nous introduisons une communication rapide et superficielle.

Attention, au stade de notre progression, nous n'introduisons pas de suite dans notre plateforme, une simulation où les agent disposeraient du langage. Nous utilisons seulement une communication superficielle, non langagière. Contrairement au langage, elle ne transmet pas de règles de comportement au moyen de paroles articulées, mais seulement des informations à propos de l'état du monde. Et ces dernières sont simplement codées au moyen de signes ou de symboles.

La raison d'être de cette démarche s'explique par la notion de cognition distribuée. Les agents, répartis sur le support, ne perçoivent que leur environnement, car leur perception est li-

12 A6S : Agent Somatique Situé en Survie Sur un Support Simple

mité à un horizon très petit. Donc, ils ont intérêt à communiquer afin de partager leurs informations, et de mieux vivre de ses ressources.

Cette communication est très simple, c'est pourquoi, selon un point de vue de bionicien [GM 2008], elle apparaît très tôt dans la phylogenèse. Par exemple, les insectes l'utilisent déjà quand ils communiquent au moyen des phéromones [Cas 2006] [OZ 2005].

2) Trois exemples illustrant cette communication par signe ou symbole

Dans notre progression de simulations nous présentons trois illustrations de cette communication :

Les insectes sociaux.

Des insectes communiquent au moyen de phéromones. L'agent à l'origine de l'information marque le sol au moyen du médium : une phéromone de stress. Quand un agent la perçoit, cette marque induit chez lui un comportement rudimentaire : elle inhibe son action.

Le groupe coopératif.

Les agents sont en situation de survie sur un support 2D où la nourriture est distribuée par gros tas très espacés. Les agents se dispersent et partent à sa recherche : ils se répartissent sur le support. Le premier qui en trouve appelle les autres. Une fois que le groupe a mangé toute la nourriture, le cycle recommence.

L'apprentissage de règles.

Un couple parent-enfant est placé sur une grille parsemée d'obstacles réguliers. L'enfant doit apprendre à s'y déplacer, son parent le guide : il se déplace sur la grille en l'appelant. L'enfant le suit en se tournant en direction de l'appel, puis en avançant.

Ainsi, l'enfant apprend à se déplacer parmi les obstacles en analysant les actions qu'il doit effectuer afin de suivre son père.

Ch. 9 La transmission culturelle au moyen du langage

1) Nous choisissons de séparer *cognition partagée* et *transmission culturelle*

Au sein de notre progression de simulation nous choisissons de séparer *cognition partagée* et *transmission culturelle*. Nous y avons déjà introduit la première car, très tôt, elle permet aux agents soumis à l'effet d'horizon, d'échanger des données à propos du monde. Ainsi, ils partagent leur connaissance à ce sujet.

Maintenant, à la fin de notre progression de simulations, nous introduisons plusieurs modélisations de la transmission culturelle. Nous la définissons comme celle qui permet aux agents de transmettre des règles épigénétiques au moyen de la communication, et particulièrement, du langage.

Ceci nous amène à faire l'hypothèse que le langage pourrait apparaître comme un détournement des outils initiaux de communication. Nous avançons que le langage réutilise les mécanismes qui très tôt ont permis aux agents de traiter la transmission et l'échange d' informations à propos du monde. Ils s'en servent pour des transferts culturels, afin de transmettre oralement, des règles épigénétiques de comportement.

2) Étape 1 : traiter une langue dont la grammaire est figée à un temps *t*

Regardons les étapes marquantes du traitement du langage naturel (TLN). Pour l'instant, nous choisissons de nous arrêter à l'étape 1, celle du traitement du langage dans le cadre d'une langue vivante, dont la grammaire du moment est figée à un temps *t*.

2.A) Analyse syntaxique de la transmission :

Analyse syntaxique d'une phrase.

Noam Chomsky [Cho 1956], [Cho 1969] montre que l'ordinateur peut effectuer automatiquement l'analyse syntaxique d'une langue (parsing).

Analyse syntaxique du discours .

Hans Kamp [Kam 1983] montre comment effectuer l'analyse du discours (gestion des anaphores au sein de plusieurs phrases contextuelles). Il traite principalement les aspects syntaxiques, et aborde très peu la sémantique.

2.B) Analyse syntaxique + interprétation sémantique :

Shrdlu de Terry Winograd [Win 1972].

Dans son micro-monde des blocs, Shrdlu, le robot virtuel de Terry Winograd, interprète sémantiquement l'analyse syntaxique qu'il effectue sur les ordres reçus en langage naturel.

L'analyse des fables de Gérard Sabah [Sab 1978] [Sab 1988].

Gérard Sabah s'attache à dégager une certaine régularité dans le scénario des fables, mais ce travail est difficile. En fait, il se trouve confronté au problème de l'inscrutabilité de l'identité radicale (qualification problem) [Car 1959]. Nous détaillons ce problème ci-après, dans le chapitre : *Les difficultés induites par notre problématique.*

2.C) Traitement de la pragmatique : scénarios et protocoles

Avec leurs scripts et scénarios Schank et Abelson [SA 1977] prennent en compte la pragmatique, i.e. la dimension culturelle qui intervient dans les protocoles des activités des groupes sociaux.

3) Étape 2 de l'état de l'art : traitement dynamique d'une langue vivante

Transition du traitement statique au traitement dynamique de la langue

Jusqu'à maintenant nous avons présenté le traitement de la grammaire d'un langage qui est supposé figé, i.e. d'une langue morte. Maintenant, nous devons présenter le traitement d'une langue vivante.

3.A) Quelques précisions initiales

Maintenant, l'objet de notre étude est la langue vivante.

Ici nous présentons l'apprentissage dynamique de la syntaxe et de la sémantique d'une langue vivante. Maintenant nous traitons la langue vivante, dont la grammaire est changeante. Alors son évolution continuelle démode rapidement le travail des linguistes. Afin de palier ce problème, à intervalle de temps réguliers, les chercheurs utilisent l'ordinateur pour effectuer rapidement une analyse. Dans ce cas, elle est qualifiée de *dynamique*.

L'information fournie ou échangée, porte un contenu sémantique codé sur différents media.

Pour que l'apprenant puisse comprendre l'échange et en extraire une information, le message transmis doit porter un contenu sémantique. Ainsi, selon la simulation informatique effectuée, pour introduire le sens dans la modélisation, plusieurs média sont utilisés, accompagnés des méthodes automatiques correspondantes. Pour illustrer cet aspect, voici une liste, non exhaustive, de quelques exemples :

- Dans le cadre d'un protocole d'apprentissage, l'enseignant fournit le sens en évaluant le savoir acquis par l'étudiant.
- La sémantique est fournie au travers de gros corpus du langage à apprendre.
- La sémantique provient indirectement de la sanction létale de vie artificielle.
- Le sens émerge de l'interaction : l'agent apprend en se cognant aux autres, ou en jouant à un jeu simulé par l'ordinateur.

L'apprentissage à effectuer peut être divers.

Durant ce travail, l'apprentissage à effectuer peut être de différents types et porter sur divers contenus. Par exemple, l'ordinateur liste le lexique d'une langue, ou bien il étudie sa syntaxe. De même, le contenu du message échangé peut être constitué de faits à propos du monde, ou de règles de conduite à tenir selon la situation donnée.

3.B) Apprentissage en introduisant une sémantique provenant de l'extérieur

Maintenant revoyons tous ces différents apprentissages en détail :

a) Le maître fournit la sémantique : il enseigne à l'apprenant un lexique

Durant cet apprentissage par émergence, la sémantique provient de l'enseignant : selon un protocole riche voire complexe l'enseignant fournit la sémantique en sanctionnant l'apprenant.

Dans la mouvance de Steels[Ste 1996] et Kaplan : Tao Gong, Andrea Puglisi, Vittorio Loreto et William S.-Y. Wang [GLW+ 2008].

Résultats.

Noam Chomsky [Cho 1956], [Cho 1969] considère que le cerveau humain possède de façon innée, l'aptitude à traiter la grammaire des langues naturelles. Cette explication constitue une avancée dans le sens où elle montre que le lexique d'une langue peut être appris au moyen d'un ordinateur. Elle implique que les agents exécutent un programme de taille conséquente

définissant le protocole d'apprentissage où interviennent deux partenaires. Donc, à l'instar de Chomsky, elle présuppose dans le cerveau l'existence d'une fonction initiale.

Commentaires.

Cette étape qui s'attache à expliquer l'émergence du lexique constitue un pas de plus vers l'explication de l'origine du langage. Cependant, il reste aux chercheurs de cette mouvance à expliquer la provenance innée de cette fonction cérébrale, i.e. de montrer comment son câblage émerge dans le cerveau au moyen de la morphogénèse.

b) Sémantique d'une structure définie par ses relations avec les autres structures

Rappel sur la notion de sémantique (des types ou des catégories).

Selon un point de vue classique, qui est aussi utilisé en programmation, au sein du synoptique d'un fonctionnement, traitement global de l'information, la sémantique d'une fonction ou d'une structure est contenue, codée, définie dans les relations qu'elle entretient avec les autres catégories de fonctions ou de structures.

c) La sémantique est fournie au travers de gros corpus du langage à apprendre.

Analyse syntaxique superficielle (Shalow parsing)

Dans ces deux exemples qui nous évoquons ici, la sémantique est fournie indirectement par la donnée de gros corpora du langage à traiter. Dans une première étape, une analyse syntaxique à faible profondeur des phrases dégage des régularités parmi les relations entre les constituants de la phrase. Ainsi, la donnée de ces relations entre structures définit une sémantique. Puis, dans un second temps, l'information portée par ces relations est utilisée de deux façons différentes dans chacun des exemples :

- Anna Pappa utilise ces résultats pour tagger [PAP 2009].

La sémantique dégagée des relations au sein des mots de la phrase sert à tagger ces derniers et/ou à construire un lexique.

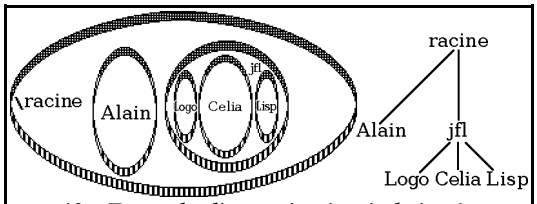
- Gilles Bernard [BER 2000] et Jean-Jacques Mariage [MAR 2006] utilisent ces résultats pour classer.

Les relations sémantiques relevées entre les mots de la phrase servent à les classer par émergence, au moyen des réseaux de neurones auto-organisateurs de Tuvo Kohonen [KOH 1982], [KOH 1984], [KOH 2001].

d) Sémantique spatiale fournie par la donnée de relations entre des conteneurs clos

Précisons notre domaine de travail : les conteneurs 2D.

Dans notre mémoire de DEA (J-François Lucas - [Luc 2004]), nous travaillons sur un support construit au moyen de conteneurs 2D, par exemple des enclos ou clôture. Dans ce cadre, deux conteneurs ne peuvent pas voir leur clôture se chevaucher, ils sont disposés soit l'un à côté de l'autre, soit l'un dans l'autre.



40 - Exemple d'organisation imbriquée de conteneurs clos

Nous nous donnons une sémantique du domaine.

À l'intérieur de ce registre de relations, de dispositions, nous nous donnons une liste de relations entre plusieurs conteneurs. Ceci détermine une architecture du domaine, et construit un micro-monde, qui est souvent arborescent.

Alors, dans ce domaine, nous pouvons utiliser un interprète en pseudo langage naturel pour :

- Modéliser le fonctionnement cognitif d'agents coopératifs : la donnée des relations entre conteneur caractérise l'architecture du domaine. Cette dernière détermine les interactions entre les conteneurs et l'agent qui s'y promène. A chaque instant élémentaire, en fonction de sa position, le joueur scrute son environnement et décide quelle action effectuer. Alors, en fonction du mouvement qu'il commande à l'interprète, l'interaction s'effectue : l'agent réussit ou non à se déplacer dans la structure.

- Animer des agents reproduisant des phénomènes linguistiques. L'agent est situé dans un domaine déterministe avec lequel il interagit. Toutes ces interaction sont causales et L'agent peut commenter ses perceptions, ses choix d'action et leur résultats. Ainsi, il verbalise continuellement la perception visuelle de sa situation (je suis dans tel lieu, je vois ceci), le commentaire de l'action qu'il effectue (maintenant je fais ceci) et son compte-rendu (j'ai réussi à effectuer mon action, ou bien j'ai échoué à la faire et j'ai perçu telle réaction de mon environnement).

En conclusion, dans cette simulations, nous obtenons deux résultats : une sémantique des conteneurs, et une modélisations de quelques phénomènes linguistiques.

La donnée de la structure du domaine suffit à caractériser la topologie du support, ce qui détermine les interactions du joueur avec son environnement. Ainsi la liste des relations entre les conteneurs suffit à caractériser la sémantique des lieux qui est correctement capturée par cet énoncé.

La structure imbriquée des conteneurs, et celle de l'agent réactif sont tellement épurées que rien n'y est superflu. Les interactions de l'agent situé avec son environnement sont peu bruitées et très empreinte de causalité. En conséquence tous les traitements qu'il effectue sur ses informations internes sont tellement justes et nécessaires qu'ils font sens. Alors leur simple verbalisation modélise un phénomène linguistique.

3.C) Apprendre avec une sémantique interne : la sanction de l'efficacité

Apprentissage au moyen d'une sémantique interne apportée par la sanction de l'efficacité.

a) La sémantique provient des parties jouées selon les règles d'un jeu formel

En utilisant cette méthode pour introduire la sémantique dans le système, les programmeurs font moins d'hypothèses fortes : ils introduisent dans le système moins de données initiales.

En effet, l'avantage d'une telle modélisation repose sur le fait que le concepteur peut initialiser le système en lui fournissant peu de données. Il place l'agent en situation sur un plateau de jeu, et la justesse de son action est évaluée en fonction de sa situation et de la règle du jeu.

En conclusion, fournir la règle du jeu introduit la sémantique dans le système au moyen d'une définition en compréhension (et pas en extension).

Des Agents Cognitifs Situés Apprennent à désigner des Actions.
Dans ce travail sur le lexique, les agents cognitifs situés apprennent à désigner des actions. L'apprentissage se fait en jouant à un jeu formel simulé par l'ordinateur. La sémantique est fournie par les interactions qui surviennent au fil des parties jouées : J. Poudade, L. Landwerlin, P. Paroubek [PLP 2002].

La méthode de Monte-Carlo pour jouer au go de Bernard Helmstetter [Hel 2006].

Dans ce travail, l'ordinateur apprend à jouer au go en simulant. Il imagine une séquence de jeu pendant laquelle, il joue les coups au hasard (mais en respectant les règles du jeu). L'apprentissage s'effectue ensuite en évaluant le résultat obtenu, et par l'intérêt de la séquence jouée.

Cette méthode illustre typiquement ce chapitre. Dans ce travail, les hypothèses faites à priori par le programmeur sont extrêmement faibles. D'abord la sémantique est fournie par les règles du jeu, mais surtout le calcul des coups par le joueur est simple puisqu'il joue les coups sans réfléchir.

Ari le programme de Jean Méhat [MC 2010].

Ari est un programme capable de jouer à de jeux généraux. Au début de la partie, la règle du jeu est fournie sous la forme d'équations logiques du premier ordre. Donc ici, les seules données extérieures qui fournissent une sémantique au système, sont clairement identifiables. Pour jouer, Ary utilise la méthode de Monte-Carlo.

b) La sémantique provient de l'évaluation de l'efficacité de l'action

Apprentissage au moyen d'une sémantique apportée par l'évaluation de l'efficacité de l'action (exemple : distance *parent* ↔ *enfant*).

ApprendsRègles : la simulation de l'apprentissage d'une conduite de déplacement, J-François Lucas [Luc 2010].

Elle modélise un agent situé, qui suit sa mère, au travers d'une grille. Ainsi il apprend à se déplacer sur le support.

Pas d'hypothèse forte dans cette simulation, sa sémantique est évaluée au moyen de la distance *parent* ↔ *enfant.*

Nous sommes en présence d'une simulation très simple qui se situe en plein dans le bon créneau. D'un côté elle n'est pas triviale car elle simule une transmission culturelle *parent* ↔ *enfant*, de l'autre elle n'explose pas sous la combinatoire car elle est très épurée.

L'avantage d'une telle modélisation repose sur le fait qu'elle est effectuée sans supposer d'hypothèses fortes :

- Le protocole de cette simulation est très léger, la transmission de règles s'effectue au moyen de gestes corporels.
- L'objet de cet apprentissage est de transmettre un comportement, i.e. un jeu de règles de déplacement dans le monde.
- La forme d'apprentissage des règles est très simple, c'est un protocole de construction en largeur d'abord.
- La sanction, i.e. la contre-réaction consiste aussi en une hypothèse faible : l'enfant est rassuré quand il se tient auprès de son parent.

4) État de l'art étape 3, le langage transmet des règles épigénétiques

État de l'art étape 3, le langage pour transmettre des mèmes, des règles épigénétiques

4.A) Introduction : modéliser/simuler la transmission mémétique

L'échange culturel court-circuite la transmission génétique.

De par sa nature, la transmission culturelle propage des règles comportementales entre générations. Elle est beaucoup plus efficace que la transmission du patrimoine génétique effectuée lors de la reproduction parent→enfant. En effet, cette dernière est lourdement handicapée par la nécessité de coder l'information transmise dans l'ADN, qui par nature, résiste aux mutations.

Introduction de la notion de règle épigénétique.

Selon l'étymologie, ce terme signifie *sur la génétique*. Il s'agit donc d'une couche qui se situe au-dessus de la génétique et qui en est indépendante. De par sa nature, la transmission culturelle est épigénétique, car elle se situe au-dessus de la génétique, dans le sens où elle la court-circuite : elle est plus rapide qu'elle, et donc plus efficace.

Mème = règle épigénétique.

Nous convenons d'introduire un terme nouveau en reprenant le vocabulaire de la mémétique [Jou 2005]. Ainsi, à l'instar de Richard Dawkins [Daw 1987], nous nommons *mème* une *règle épigénétique*.

Problématique : comment construire des simulateurs capables de modéliser des transmissions mémétiques.

De façon générale, dans le cadre de notre thèse nous cherchons à modéliser les phénomènes cognitifs et langagiers. Parmi les directions de travail envisageables, nous avons exploré la transmission mémétique.

En effet, sa nature en fait une orientation intéressante. D'abord, elle consiste en un acte langagier épuré dont les rouages sont visibles et dissécables. Ensuite, elle porte un sens immédiatement confirmé par son receveur : quand il incorpore la règle reçue à ses comportements, il y trouve un bénéfice.

Donc, au sein de notre plate-forme de simulations, nous avons cherché à construire des simulateurs qui modéliseraient des transmissions mémétiques.

Ébaucher un état de l'art des simulateurs mémétiques.

Avant de se lancer dans cette tâche ambitieuse, il est naturel de regarder l'histoire des sciences et d'ébaucher un état de l'art de quelques simulateurs mémétiques.

4.B) Simuler avec lissage statistique : le mème est autonome mais vide de sens

La mémétique consiste aussi en un changement radical de point de vue. Son regard n'est pas celui du linguiste ou de l'informaticien qui se place au niveau du système et étudie les échanges culturels entre agents. Au contraire, à l'instar de Nicolas Copernic, le méméticien change de référentiel et prend le mème pour objet central de son étude. Quand il procède ainsi, il s'intéresse à sa diffusion, à sa propagation de cerveau à cerveau.

Alors, le modélisateur est soumis à une tentation forte : utiliser l'argument du *lissage statistique* pour attribuer au mème des règles lissés de comportement. Ainsi il obtient un mème autonome, dégagé de l'influence de son vecteur, i.e. dont la propagation devient indépendante de l'agent qui le porte. Ainsi elle est réduite à des lois mathématiques, fonctions de grandeurs physiques vides de sens : le temps, l'espace, le nombre de propagations.

En fait, quand le méméticien choisit ce modèle de diffusion, quand il place ces mèmes autonomes dans un tel simulateur, il construit un système réducteur donc le fonctionnement est

lisse et vide de sens : en simplifiant trop, il a jeté le bébé avec l'eau du bain, il a écrasé à plates coutures l'épaisseur ontologique du mème.

C'est le cas de l'assimilateur mémétique de Pierre-Alain Cotnoir [Cot 2007] : il fournit une simulation anémiée, qui s'avère peut-être valable pour effectuer des enquêtes de sociologie (le domaine professionnel de M. Cotnoir), mais qui inutilisable pour modéliser des traitements cognitifs.

4.C) Simulateur de l'IA symbolique : forger une identité aux mèmes

Simulateur de l'IA symbolique, pour donner une épaisseur ontologique aux mèmes en leur forgeant une identité.

La problématique.

Puisqu'en simplifiant trop on obtient une simulation anémiée, les chercheurs de l'IA ont tenté et tentent de cerner l'identité des mèmes au moyen d'une description symbolique. Mais, pour la modéliser, ils buttent contre le *qualification problem* [Car 1959] : il est extrêmement difficile de décrire radicalement l'identité des êtres au moyen d'un programme informatique qui recréerait leur comportement, ou d'une structure de données qui illustrerait leur architecture.

Le simulateur Minimème (1994 – inachevé).

Ainsi, à Paris 6, en 1994, dans l'équipe Miriad que dirigeait Jacques Ferber, Stéphane Bura a travaillé sur un tel simulateur mémétique *Minimème*, malheureusement, il n'a jamais terminé sa thèse.

Le simulateur MemSim (2003 – inachevé).

De même, à la SFM (société francophone de mémétique), Pascal Jouxtel, Charles Mougel travaillent aussi sur *MemSim*, un simulateur mémétique, qui demeure à l'état de projet. Lors du cinquième séminaire de Mémétique francophone qui s'est déroulé à Paris les 27, 28 et 29 juin 2008, j'ai dû leur dire que selon moi, ils buttaient sur le *qualification problem* : le problème de l'inscrutabilité de l'identité radicale.

JFMemSim (2006/Java – 2007/C++).

Par contre, en 2006-2007, nous avons présenté un simulateur mémétique qui effectue quelques transferts et fusions simples de mèmes. Ils ne les lisse pas, mais s'attache à leur modéliser une identité (il est présenté plus en détail dans le paragraphe suivant).

Mais il est très bridé, non paramétrable, Pascal Jouxtel [Jou 2005], nous a demandé de le rendre paramétrable. Gilles Bernard nous a demandé d'en faire une plate-forme universelle, mais tout cela demeure à l'état de projet car la tâche est lourde. Cependant nous pensons que la plate-forme de simulations que nous proposons débroussaille le chemin dans cette direction.

4.D) Article de Nicolas Journet *Les espoirs déçus de la mémétique* [Jou 2007]

Dans son article, *les espoirs déçus de la mémétique*, Nicolas Journet constate qu'aux États-unis la communauté mémétique baisse les bras, et il termine son article en expliquant combien les méméticiens ont besoin de réaliser des simulateurs mémétiques :

> ... Malgré sa reprise annoncée, le *Journal of memetics* a cessé de paraître depuis et n'a pas été remplacé. Évidemment, cela ne veut pas dire que les méméticiens aient tous renoncé. D. Dennett, S. Blackmore, le biologiste Derek Gatherer, le cybernéticien Francis Heylighen ne cachent pas leur attachement à l'idée de même. Ils ont l'espoir de voir la théorie s'affirmer, ne serait-ce que dans une *niche*, celle de la simulation par exemple. En effet, le jour où un programme de simulation permettra de prévoir le succès mémétique d'une idée, la théorie aura gagné le droit de refaire parler d'elle. En attendant, la mémétique a du plomb dans l'aile.

Sur le blog [Jou 2010], nous pouvons lire la réponse de Pascal Jouxtel, fondateur de la SFM Société Francophone de Mémétique, et auteur du livre *Comment les systèmes pondent* [Jou 2005].

4.E) Interaction et construction en couches donne une épaisseur à l'agent

Ici nous présentons une partie de notre recherche, i.e. celle concernant les retombées de notre plate-forme de simulation, en direction des simulateurs mémétiques.

a) JFMemSim : un simulateur mémétique

Deux directions pour un simulateur mémétique : interaction d'agents et agents construits en niveaux

L'intelligence est dans l'interaction.
Ce n'est pas un mystère, depuis longtemps les chercheurs[13] de l'IAD le disent : *l'intelligence est dans l'interaction*. Notre simulateur modélise des êtres qui ne sont pas lisses, mais qui présentent plutôt une épaisseur ontologique. Cette modélisation est obtenue en décrivant, au niveau le plus bas, deux sortes d'interactions : *agent* ↔ *support* et *agent* ↔ *agent*.

Émergence à un niveau macroscopique d'un comportement décrit à un bas niveau.
L'architecture de chaque agent est construite selon structure en pelures d'oignons, i.e. en couches concentriques.
Notre simulateur n'est pas anémié, il fait interagir des agents munis d'une épaisseur ontologique. Pour les modéliser, il décrit leurs interactions à un niveau microscopique afin de récupérer un comportement riche qui émerge à un niveau macroscopique.

Nous avons déjà évoqué cet aspect, le lecteur le trouvera dans les concepts, à propos des trichotomies.

b) JFMemSim : annoncer les résultats présentés en détail dans la partie pratique

Voyons quelles étapes ont été franchies dans cette direction.

MemeticTransmet.
Simulation d'une transmission mémétique langagière.

MemeticLamarck.
Simulation d'algorithme génétiques dans le cadre d'un lamarckisme des idées.

LearnTrajet encore appelé TransmetRègle.
Simulation de transfert culturel au moyen de gestes corporels.

c) Perspectives pour JFMemSim

JFMemSim demeure un projet.
L'idée de réaliser un vrai simulateur mémétique demeure un projet.

Projet de protocole de troc en deux agents vivant dans des situations différentes.
Il y a déjà 2 ans (LIASD - axe 2 AIVD, 23/06/08) que j'ai signalé ce projet de troc entre deux agents. Derrière, bien sûr, il y a l'échange et la relation commerciale. Le troc fait intervenir les protocoles et donc on retombe dans la pragmatique : les scénarios de Schank [SA 1977].

13 Partick Albert, directeur de la société Ilog, lors de sa conférence, pour l'anniversaire de l'IA, à l'ARC, Paris 2006.

Ch. 10 Les difficultés induites par notre problématique

Les problèmes à surmonter avant d'atteindre nos buts

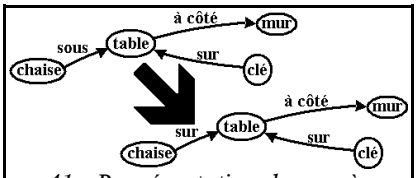
1) Problèmes que la logique éprouve pour représenter les scènes dynamiques

a) La variation de relations pour représenter le procès

Nous savons représenter l'état du monde au moyen de plusieurs relations entre ses composantes. Convenons d'appeler *procès* le changement d'état du monde, i.e. le changement d'une (ou plusieurs) de ces relations. Donc la représentation d'un procès se fait simplement en changeant (ou plusieurs) de ces relations entre les composantes du monde. Par exemple, dans l'illustration ci-contre, la chaise qui était sous la table, passe sur la table.

b) La logique utilisée pour représenter l'action

La logique éprouve une difficulté dans la RC dynamique, elle peine à représenter le procès/l'action qui change l'état du monde. Bien sûr cette représentation peut tout de même s'effectuer au moyen du *modus ponens*, mais dans ce cas, le prix à payer est lourd, nous devons introduire la dimension temporelle dans toutes les relations et alors nous buttons sur le problème du cadre (le frame problem).



41 - Représentation du procès qui change l'état du monde

Voyons cela plus en détails.

c) La représentation des connaissances en logique

En logique, quand nous effectuons une déduction, nous listons l'ensemble des faits vrais à propos de l'univers, en les reliant implicitement dans notre tête, par le connecteur logique ET i.e. $\wedge$.

Par exemple, prenons le syllogisme classique : *Tous les hommes sont mortels, Socrate est un homme, donc Socrate est mortel*. Au moyen de ce raisonnement, en partant du fait que *Socrate est un homme*, nous déduisons que *Socrate est mortel*. Après cette déduction, nous obtenons les deux faits suivants :

homme(Socrate); | Ici, le connecteur ET n'est pas matérialisé,
mortel(Socrate); | Il est donc est implicite

Ce qui correspond en logique à la formule :

homme(Socrate) $\wedge$ *mortel(Socrate);* | Ici, nous avons écrit explicitement le connecteur ET.

d) Le problème de l'inconsistance de la simulation par la logique

Dans le cadre de la logique, il est possible de faire de la simulation. Alors certains, qui désirent tout voir au travers de ce prisme, ont souhaité procéder avec elle. Mais ainsi, la simulation ne se réalise pas directement. Voyons donc pourquoi il faut ajouter au traitement logique, une petite fonctionnalité, la dimension temporelle, qui s'avère ensuite lourde de conséquences.

Quand nous utilisons la logique pour faire de la simulation, au niveau de la base des faits à propos du domaine (de l'univers), survient un problème d'inconsistance, qui provient de ce que la logique travaille selon un processus incrémental des connaissances à propos du monde[14].

14 Ici, dans la suite de notre exemple, nous sommes dans le cadre de la logique incarnée dans le réel car nous faisons de la simulation. Alors, à la place de la terminologie *domaine,* nous emploierons celle de *monde*, qui nous semble plus adaptée que le terme *univers*.

e) Prenons un exemple très simple pour illustrer l'inconsistance

Supposons que nous voulons simuler le déplacement de Toto, qui va de la maison au jardin. Au départ, la base de faits contient seulement le prédicat :

lieu(Toto, Maison); | Toto est dans la maison

Sans rentrer dans les détails, nous supposons que nous avons écrit une règle de transformation, qui simule le déplacement de Toto. Ainsi, à la fin de ce processus, le résultat obtenu simule le fait que Toto soit maintenant dans le jardin, au moyen du prédicat suivant :

lieu(Toto, Jardin); | Toto est dans le jardin

Parce que la logique est un processus incrémental, maintenant la base de faits devient :

lieu(Toto, Maison); | Toto est dans la maison
lieu(Toto, Jardin); | Toto est dans le jardin

Ce qui correspond en logique à :

lieu(Toto, Maison) ∧ lieu(Toto, Jardin); | *Toto est ds la maison et Toto est ds le jardin*

Arrivé à ce stade on écrit l'axiome de non-ubiquité : *un élément du monde ne peut pas être en deux lieux à la fois* :

x ≠ y ⇒ ¬(lieu(z, x) ∧ lieu(z, y)); |*si x≠y donc z ne peut pas être dans x et ds y*

Ensuite on écrit que la maison n'est pas dans le jardin, et réciproquement

Maison ≠ Jardin | *La maison est différente du jardin*

Utilisant l'axiome du *modus ponens*, en prenant x=Maison, y=Jardin, z=Toto, on déduit un nouvel axiome :

Maison≠Toto ⇒ ¬(lieu(Toto, Maison)∧lieu(Toto, Jardin));Si Maison est ≠ de Jardin, alors |*Toto ne peut pas être dans Maison et Jardin.*

Cet axiome permet rapidement de déduire :

¬(lieu(Toto, Maison) ∧ lieu(Toto, Jardin)); |*Toto ne peut pas être dans Maison et Jardin.*

Sachant que par ailleurs nous avons déjà les faits suivants :

(lieu(Toto, Maison) ∧ lieu(Toto, Jardin)); |*Toto est dans maison et Toto est dans jardin*

En posant A= (*lieu(Toto, Maison) ∧ lieu(Toto, Jardin)),* on déduit les deux faits suivants :

¬A ∧ A; | On déduit tout et son contraire.
| Catastrophe, le système est inconsistant !

f) Aménager la simulation en logique pour éviter l'inconsistance

Quand la logique est utilisée en simulation, l'axiome de non-ubiquité induit un problème d'inconsistance. Heureusement, un aménagement permet de l'éviter. Il suffit d'introduire la dimension temporelle dans le système, i.e. il nous suffit de dater les faits. Dans ce cas, simuler le déplacement de Toto, qui va de la maison au jardin donnera par exemple :

lieu(Date1, Toto, Maison); | À la date 1 Toto est dans la maison
lieu(Date2, Toto, Jardin); | À la date 2 Toto est dans le jardin

Ce qui, écrit en logique, produit alors une base de faits qui demeure consistante.

g) Dater les fait pour éviter l'inconsistance induit le problème du cadre

Nous avons vu qu'avec la logique, la simulation tombe sous les coups de l'inconsistance de sa base de faits. Nous avons précisé qu'un aménagement très simple, l'introduction de la dimension temporelle, permet de l'éviter. Tout semble pour le mieux dans le meilleur des mondes, mais, en simulation, cet ajout suffit à faire apparaître un nouveau problème grave : le problème du cadre (frame problem), i.e. de l'arrière plan, qui dans la réalité physique demeure immobile lors d'une simulation, mais doit cependant être traité au niveau de la représentation.

Pour fixer les idées, prenons une simulation dans un monde de 1000 objets. Si au premier plan un agent déplace un objet, alors, à chaque intervalle de temps, le processus doit re-dater tous les objets : celui qui bouge au premier plan, mais aussi tous les objets immobiles en arrière plan. De plus, à chaque fois, il doit ajouter ces 1000 nouveaux faits obtenus, à la base de connaissances.

En conclusion, rien qu'à cause de cette simple modification, la simulation dans le cadre logique explose sous la combinatoire.

Certes on peut trouver des solutions. L'idée est de créer une gestion journalisée des faits transformés par les actions. Mais, la méthode logique munie de cet aménagement, même si elle est plus fine car elle prend bien en charge le temps, demeure cependant lourde, comparée à la simplicité de la simulation au moyen des systèmes formels.

h) Conclusion : délaisser la logique et utiliser les systèmes formels

En conclusion, nous venons d'expliquer pourquoi, afin d'implémenter notre travail, nous délaissons la logique mathématique et utilisons les systèmes formels.

2) Problème de la complexité à modéliser la réponse du monde

Puisque le monde consiste en un support et son mobilier, nous traitons de de la difficulté à modéliser la réponse du support et de son mobilier(nourriture, obstacles et agents).

2.A) Introduction, position du problème

Introduction du concept général de *fonction*, qui se trouve au centre du fonctionnalisme.

D'abord, avec le fonctionnalisme, i.e. le computationnalisme, nous introduisons la notion abstraite de fonction.

Dans les cas les plus simples, en mathématiques, c'est tout simplement la fonction classique : y = f(x) ; et en informatique elle est fournie par une table qui possède une entrée dans un séquenceur où y = f(y), et deux entrées chez l'automate fini (AF) où y = f(x, y);

L'utilisation d'une mémoire introduit le séquentiel i.e. la dimension temporelle.

Pour simuler nous devons introduire le temps dans nos modèles. Par exemple, les simulations avec le séquenceur et l'AF le font intervenir sous la forme d'une mémorisation effectuée par le registre tampon.

Introduction du procès : le changement d'état du monde.

Avec l'introduction de la dimension temporelle, nous passons d'une RC statique du domaine à une RC dynamique. Au moyen d'une opération nommée procès, où l'être change l'état du domaine.

Ce changement est obtenu au moyen de deux outils : la fonction et la mémoire. Dans le séquenceur, l'état au temps t+1 est calculé en fonction de l'état au temps t. Avec l'automate à état fini, il tient aussi compte de l'environnement. Le rôle du tampon est de mémoriser le nouvel état du système.

Ce changement ne constitue pas encore une action, car il n'est pas intentionnel, i.e. qu'il n'est pas sous-tendu par un but.

Le domaine mathématique est plongé dans la réalité physique du monde.

Nous passons de l'abstrait au concret, en incarnant le domaine dans la réalité du monde. Ainsi, le domaine devient le monde, les états/symboles du domaine deviennent des lieux, l'état du système devient le lieu où est l'être. Le procès qu'effectue l'être consiste en un changement de lieu, mais ce n'est pas encore une action.

L'introduction d'un support particulier induit l'automate situé.

Nous introduisons l'automate situé en utilisant dans le modèle un support sous la forme d'un domaine de lieux présentant deux caractéristiques spéciales : d'abord les lieux sont nombreux, ensuite, chacun d'eux est relié seulement à ses voisins les plus proches. La relation qu'un lieu entretient avec ses voisins est limitée par un effet d'horizon de profondeur 1 : il ne voit que ses plus proches voisins.

De par sa nature un automate prend ses décisions au moyen d'une table. Maintenant, à cause du grand nombre de lieux du domaine, celle-ci grandit selon une loi en n^2 (où n est le nombre de lieux du support). À cause du profil particulier du domaine utilisé, elle est creuse et appelle une simplification. Ceci explique que, chez les agents, la décision est prise au moyen de règles.

Nous introduisons l'action en la définissant comme un procès finaliste.

En introduisant l'énergie dans le système nous y introduisons l'action en la définissant comme un procès finaliste. C'est seulement à ce moment que l'être devient un agent, car selon l'étymologie *agere=agir*, l'agent agit.

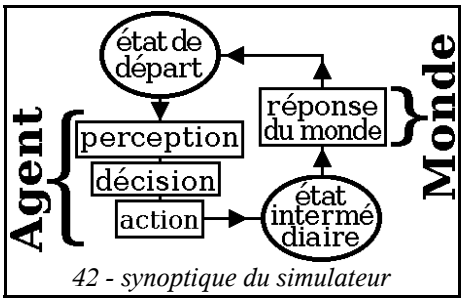
L'introduction du parallélisme induit le groupe.

Pour l'AF, nous travaillons déjà avec une fonction de deux variables, mais rapidement le monde devient complexe quand on introduit le parallélisme, i.e. si on y modélise des obstacles, de la nourriture et d'autres agents. Dans ce cas, nous buttons contre le mur de la complexité.

2.B) En écrivant nos modélisations, nous buttons contre le mur de la complexité

Finalement ces modification complexifient le système.

En conclusion : l'introduction de ces trois concepts *fonctionnalisme, parallélisme* et *agents situés* n'est pas anodine. Elle induit une lourde conséquence : le poids de la complexité.



42 - synoptique du simulateur

Démarche de simulation : nous devons modéliser le monde (support et mobilier).

Dans une démarche de simulation nous devons modéliser, i.e. décrire le synoptique de fonctionnement du système, i.e. du simulateur. Mais comment capturer l'essence du monde afin de le modéliser au moyen d'une fonction ?

À la recherche du synoptique de fonctionnement du simulateur.

Pour décrire le fonctionnement du simulateur, nous devons le décomposer en deux sous-fonctions :

- la réponse de l'agent en fonction de l'état du monde
- la réponse du support en fonction des actions de l'agent.

Comment modéliser les réponses : agent(monde) et support(actions).

Mais comment modéliser la réponse de l'agent vu l'état du monde, et la réponse du support vu les actions de l'agent ? Pour cela, il faut utiliser les mathématiques et recenser servilement la fonction de chacun des êtres en interaction avec les autres. Cela consiste en un travail de bénédictin.

Pour recenser les réponses de l'agent et du monde, nous tombons sur des problèmes de complexité.

Rien que quand nous voulons modéliser le fonctionnement d'agents réactifs situés nous tombons sur des problèmes de complexité. De plus, la difficulté sera encore plus grande quand nous essaierons de dégager le fonctionnement de la cognition.

2.C) Exemple de cette complexité à modéliser la réponse du monde

Maintenant, pour illustrer ce problème à modéliser la réponse du monde, nous traitons un exemple important: *le qualification problem,* i.e. la difficulté qu'il y a pour décrire et cerner l'identité radicale[15] des objets, agents et êtres.

a) Ce problème est apparu en I.A. symbolique[16]

Le problème de l'inscrutabilité de l'identité radicale est vraiment un gros problème qui, à l'heure actuelle, demeure encore relativement insoluble. Il est apparu très tôt en I.A. symbolique. Mc Carthy, créateur du Lisp et fondateur en 1956 de l'I.A., l'a signalé dès 1959.

b) Définition du *qualification problem*

Classiquement, en I.A. symbolique, il est fastidieux, voire impossible de capturer, d'analyser, de décrire le fonctionnement d'un monde au moyen d'une procédure/fonction ou d'un jeu de règles. Cette difficulté est presque insoluble. Elle est désignée en anglais sous le nom de *qualification problem* et en français sous celui d' *inscrutabilité de l'identité radicale*. Ce problème correspond à l'énorme difficulté rencontrée pour scruter et rédiger les règles qui

15 En anglais : the qualification problem. Pour cela voir Mc Carthy [Car 1959] (celui-là même qui a créé lisp).

16 In philosophy and AI (especially, knowledge based systems), the qualification problem is concerned with the impossibility of listing all the preconditions required for a real-world action to have its intended effect. It might be posed as how to deal with the things that prevent me from achieving my intended result. (wikipedia)

décrivent le comportement des agents complexes. Il apparaît surtout dans les simulations de groupes d'agents évolués où il devient impossible d'écrire de façon exhaustive les procédures simulant le comportement des entités.

c) Ex. 1 illustrant cette difficulté : lister les préconditions pour allumer la lumière

Par exemple, Toto fait le geste d'allumer la lumière, mais il existe des dizaines de raisons, que le programme doit tester, pour savoir si cet enfant va réussir dans son entreprise : dans le quartier, il y a une coupure générale d'électricité, le disjoncteur de l'appartement a sauté, l'ampoule est défectueuse ou absente, l'interrupteur commande autre chose, avant de faire ce geste il glisse ou il a une crampe…

d) Exemple 2 pour illustrer cette difficulté : décrire un modèle universel de chat

Voici une autre façon, assez formidable (dans le sens classique du terme) d'illustrer la difficulté rencontrée pour cerner, au sein d'un modèle, le comportement d'un agent complexe. Supposons que nous devions décrire un chat. Selon le contexte, il devrait apparaître différemment, car il peut être :

- L'animal de compagnie qui apporte de l'affection à une personne âgée.
- Le félidé, terrible chasseur sanguinaire, pour une souris.
- Une forme sombre qui, un soir de pluie, traverse une rue en pente, devant un cycliste débutant.
- Le chat noir, qui passe sous une échelle un vendredi 13, et fait peur à une personne superstitieuse

e) Inscrutabilité de l'identité radicale

En conclusion, l'identité d'un être est relativement inscrutable au niveau symbolique. D'abord elle apparaît comme un épiphénomène du niveau du composant, ensuite, elle est si riche, si diverse, qu'elle change selon l'éclairage du thème abordé.

L'identité d'un être, vue comme un épiphénomène du niveau du composant.

En I.A., si on quitte le niveau du composant, pour se placer à l'étage symbolique, i.e. si on observe selon un point de vue macroscopique, l'identité radicale d'un être devient inscrutable, car elle apparaît comme un épiphénomène, émergent de l'interaction des composants. À ce moment, la multiplicité des phénomènes induits est tellement riche que leur description exhaustive devient fastidieuse.

L'identité d'un être varie en fonction du contexte.

L'identité d'un être est relativement inscrutable au niveau symbolique. Nous avons déjà cité l'exemple du chat noir, nous pouvons encore citer celui de l'eau, car ici encore, son identité varie en fonction du contexte :

- Pour un chimiste, c'est la molécule H_2O.
- Pour un plombier, elle constitue son fond de commence.
- Elle sauve la vie d'un voyageur assoiffé dans le désert.
- Pour un marin, elle fait flotter les bateaux.
- Pour un pêcheur, elle constitue le lieu de vie des poissons.
- Pour le gamin, elle signifie *barboter*.
- Pour le surveillant de baignade, elle constitue un danger.

La conséquence de ce comportement à facettes fait que le code écrit pour un contexte, ne peut être récupéré pour un autre : il faut rajouter une facette à la description de l'être. On peut encore considérer que, même si on a décrit plusieurs facettes de l'identité, si on change un peu l'éclairage du domaine, une nouvelle facette apparaît, et il faut à nouveau décrire et ajouter des comportements à l'identité de la chose.

2.D) Transition vers la suite :

a) Pour la clarté de l'exposé, juste annoncer, sans développer

Dans cette présentation de notre travail, à cette étape nous choisissons de déjà présenter globalement les grands traits de la méthode que nous utilisons.

b) Deux solutions : simplifier le monde, l'évolution/émergence artificielle

Ici, nous annonçons deux solutions, sans les développer

Notre solution au qualification problem : simplifier le monde utilisé.

- Support : utiliser un support digital et régulier : une grille ou un graphe à mailles carrées.
- Objets : utiliser des objets digitaux et réguliers, par exemple des pions sur le support.
- Relations : au début, seulement utiliser les relations de lieu, avec au plus un pion par lieu.

Notre solution à la complexité de la modélisation de la structure des agents : l'évolution et l' émergence artificielle.

Dans notre recherche, nous travaillons sur les agents. Pour classifier leurs différentes structures et leurs traitements de l'information correspondants, nous proposons une autre démarche : celle de l'évolution et de l'émergence artificielle. Elle consiste à partir des formes de vie les plus simples, et à essayer de les faire évoluer. Quand nous réussissons à les améliorer, nous regardons comment nous avons fait évoluer leur structure.

Évidemment, dans la suite de la présentation, nous préciserons et expliciterons cette démarche.

Ch. 11 Les concepts utilisés dans ce travail

Après avoir présenté notre problématique, nous avons raconté l'histoire et décrit l'état de l'art informatique. Ensuite nous avons pointé les difficultés induites par notre mission, nous commençons à les résoudre en décrivant les concepts généraux qui éclairent et orientent notre travail.

1) Situer, positionner les concepts d'émergence et d'évolution

Nous cherchons comment se situent, se positionnent les concepts d'émergence et d'évolution par rapport aux mathématiques.

1.A) Introduction

Nous venons d'expliquer comment l'utilisation de l'émergence nous permet de modéliser une structure complexe selon une construction en couches concentriques.

Mais selon un point de vue épistémologique, le concept d'émergence sur lequel nous focalisons présente un positionnement très original. Au sein d'une *échelle de l'ordre* qui s'étagerait entre deux extrêmes : du désordre au déterminisme, il se situe très bas : quand nous travaillons avec l'émergence, nous cherchons à organiser le chaos.

C'est pourquoi, maintenant nous allons expliquer ces concepts d'*émergence* et de *chaos*, car ils éclairent notre positionnement et expliquent notre démarche.

1.B) Le désordre

Tout au bas de l'échelle de l'ordre, nous trouvons le désordre, le hasard. Théoriquement, il correspond à l'absence totale d'ordre, au hasard le plus complet.

Le désordre est souvent utilisé en informatique au moyen de la fonction *random* pour initialiser au hasard des situations. C'est aussi ce que nous faisons, par exemple pour positionner les agents dans le monde.

En fait, ici, nous évoquons le désordre surtout pour positionner la borne inférieure de notre *échelle de l'ordre*.

1.C) Le chaos

Définition du chaos.

Dans le langage usuel, le terme *chaos* est relié à la notion de désordre total : ex : *chaos de rochers*, i.e. un système physique imprévisible car il semble n'obéir à aucune loi.

Mais, attention ! Le chaos est ordonné. En effet, il faut se méfier, car un ordre existe dans le chaos. Par exemple, dans un pierrier bien des règles d'ordonnancement existent : celle de la physique des corps solides et pesants.

Par ailleurs, la théorie du chaos traite aussi des systèmes dynamiques rigoureusement déterministes. ces derniers, surtout quand ils oscillent ou divergent, présentent un phénomène fondamental d'instabilité appelé *sensibilité aux conditions initiales.* En pratique, ce problème les rend non prédictibles sur le *long* terme.

L'émergence se complaît dans le chaos.

À la fin de ce paragraphe, nous développerons cette conclusion : à condition de disposer d'énergie, l'émergence se complet dans le chaos car il lui suffit de progresser, et c'est là le concept clé.

1.D) L'agent situé : un comportement causal avec un horizon

Dans le cadre de l'agent situé, on place un être en situation de survie sur un support où il cueille de la nourriture distribuée aléatoirement ou régulièrement.

L'agent agit de façon causale, mais il génère son comportement alors que sa perception est limité par un effet d'horizon. Cette connaissance limitée qu'il dispose de son monde l'empêche de générer un comportement optimal : il agit causalement, mais génère son comportement au moyen d'heuristiques.

1.E) L'omniscience

Dans le cadre de l'omniscience, celui qui raisonne dispose d'une connaissance totale du monde.

Dans de domaine de représentation des connaissances à propos du monde, nous tombons sur les démonstrations de théorème. Effectuées au moyen du *modus ponens*, par essence elle sont parfaites.

Dans le domaine de la génération de l'action, par exemple au moyen de la planification, on tombe sur les tactiques de la planification.

1.F) Introduction des concepts d'émergence, d'évolution et de co-évolution

Après ces pré-requis, introduisons les concepts d'émergence, d'évolution et de co-évolution

a) L'émergence et l'évolution consistent seulement à s'améliorer

À propos de chaos, nous parlions de champs de pierres. Cette image est bien adaptée à ces deux concepts. Quand la vie a littéralement émergé, les amphibiens ont trouvé le chaos de l'estran : des rochers, du sable et de la vase.

À condition de disposer d'énergie, l'émergence se contente du chaos car il lui suffit d'y progresser. Dans ce domaine, notre démarche est loin de la recherche d'un optimum, elle ne consiste même pas dans la recherche d'un maximum local. Le principal et même le suffisant consiste seulement à progresser.

Les exemples classiques illustrant ce thème sont ceux de l'évolution des espèces de Darwin : toute mutation qui va dans le sens d'une meilleure adaptation du mutant à son milieu est sélectionnée.

b) La notion de co-évolution *mobilier* ↔ *support*

Dans ce cas, la vie fonctionne dans le registre de l'évolution, elle est seulement dans la recherche d'une amélioration (montée de gradient). Elle n'est pas encore dans celui de l'optimisation, car elle ne cherche pas un maximum, même local. Elle est loin de la recherche de la perfection des théorèmes.

Si on regarde plus finement les choses, nous sommes en présence de deux modèles couplés : d'un côté la simulation de l'agent situé dans le monde, et de l'autre celle de la réponse du monde aux actions des agents.

Certes Darwin a pointé l'évolution des espèces, mais il ne faut pas oublier que la vie transforme le monde et qu'ainsi nous pouvons regarder le monde comme un système couplé *mobilier* ↔ *support* en co-évolution.

Ce point de vue se concrétise dans trois paradigmes très actuels :

La notion de niche écologique.

Ce thème est déjà évoqué dans le chapitre intitulé *notre problématique*. Il s'agit de la co-évolution du couple *mobilier* ↔ *support* où le support reste constant et où le mobilier, à savoir l'ADN et/ou la culture du groupe évolue pour s'adapter au support i.e. au milieu.

La notion d'activité située.

Comme son nom l'indique, il s'agit du activité qui se déroule sur un site donné, i.e. sur un lieu particulier du support. Dans ce cas, le domaine est particulièrement marqué pour une activité donnée. Et l'agent situé perçoit ces marques, et prend en compte ces stimuli extérieurs qui

guident son activité. Exemple : un circuit d'apprentissage, un lieu d'initiation, une chaîne de traitement ou de construction d'artefacts, un lieu d'échange social (le lien d'échange de nourriture, le script du restaurant).

Le fait d'exercer cette activité sur ce site présente un avantage pour l'individu et/ou pour le groupe. Elle est donc renforcée et sujet à évolution. Ainsi on obtient une co-évolution du couple *mobilier*↔*support*. : La qualité du marquage de l'activité augmente. La pondération des règles culturelles qui prennent en compte ce marquage augmente. L'avantage du groupe social qui pratique cette activité située croît.

La transmission culturelle inter-générationnelle et le court-circuit de la culture.

Il s'agit ici du couplage *mobilier*↔*support*, où le mobilier est le groupe culturel, et où le support est la mémoire sociale qui porte la culture. Ce phénomène s'appelle la transmission culturelle inter-générationnelle.

Dans ce cas, le bénéfice que le groupe obtient en utilisant et en améliorant sa culture est tel qu'on est en présence d'une amélioration rapide de cette-ci. Ce phénomène s'appelle le court-circuit de la culture.

2) La sémantique en simulation

Nous travaillons sur la sémantique. En simulation son positionnement peut se résumer ainsi : *donner du sens à la vie ; donner de la vie au sens.*

Après avoir présenté notre problématique, et pointé les difficultés qu'elle induit, nous commençons à les résoudre en décrivant les concepts généraux qui éclairent et orientent notre travail. Maintenant, nous précisons le concept de sémantique de la vie, qui nous aide dans l'analyse et la conception de nos simulations.

2.A) Dans les simulateurs avec lissage statistique les mèmes sont dégénérés

Nous avons déjà expliqué que dans les simulateurs avec lissage statistique le mème est autonome mais dégénéré : les simulations n'y ont que peu de sens, peu d'épaisseur ontologique.

2.B) Fourniture d'une plate-forme qui ne soit pas dégénérée

Dans le chapitre précédant, qui traite de mémétique, nous avons montré la nécessité de fournir une sémantique à un système, quand nous voulons lui faire effectuer un apprentissage.

De façon générale dans une simulation d'apprentissage, si nous voulons passer peu de temps à former l'apprenant, il faut qu'il soit suffisamment autonome pour être capable d'effectuer un auto-apprentissage. L'apprenant doit pouvoir comprendre à mi-mots, sans que l'enseignant soit obligé de lui fournir des explications détaillées.

En conclusion, si nous voulons simuler un système qui effectue un apprentissage quand nous lui fournissons une sémantique, il doit ne pas être dégénéré. En effet, si sa structure est exagérément simplifiée, elle est intrinsèquement incapable de porter sa propre sémantique, et donc de comprendre et assimiler celle qu'on lui fournit.

2.C) Partir d'une base qui possède déjà sa sémantique : celle de la VA[17]

Nous souhaitons construire une progression, pour cela nous devons partir d'une base solide, fiable, sur laquelle nous pourrons construire ensuite. Parmi les briques de base envisageables, nous prenons la cellule végétative. Nous la choisissons pour deux raisons : d'abord elle exécute un cycle de vie très simple, ensuite, comme la vie artificielle (VA), il possède un sens, il porte sa sémantique.

2.D) La démarche de construction que nous utilisons

Une démarche incrémentale.

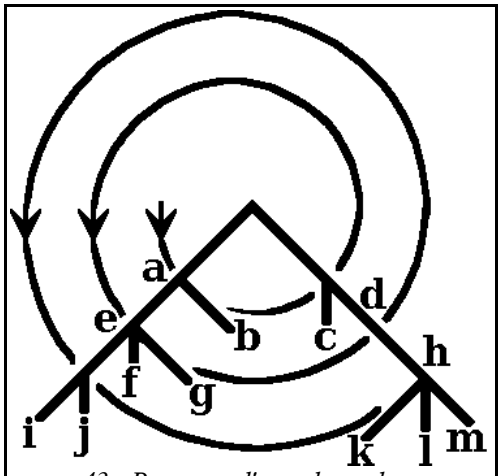
Ensuite, depuis ce niveau de base de la cellule végétative, nous construisons notre progression, selon une démarche incrémentale.

Une démarche éclectique.

Puis, une utilisons une démarche éclectique, i.e. nous choisissons de construire dans toutes les directions, selon une stratégie en largeur d'abord.

Une démarche opportuniste, à touches pointillistes.

L'idée est de travailler à touches pointillistes, d'ajouter à la taxinomie de petits apports : quand l'occasion se présente, quand nous trouvons une



43 - Parcours d'un arbre selon une stratégie en largeur d'abord

17 VA : vie artificielle.

simulation fonctionnant à l'intérieur de la zone privilégiée[18], alors nous l'ajoutons à la progression des simulations.

Ceci se produit quand nous trouvons une niche écologique où placer une simulation mettant en scène un être mutant. Dans ce cas, en partant d'un agent qui est déjà classé dans la taxinomie, nous améliore une de ses fonctionnalités ; ce faisant nous inventons un agent qui se situe un peu plus haut dans la branches de notre taxinomie.

2.E) Conclusion

En conclusion nous pouvons dire que c'est ainsi que nous ajoutons des simulations à notre progression de modélisations.

18 Voir dans le chapitre sur les concepts et les méthodes, le paragraphe intitulé : la zone privilégiée où la masse est critique nous incite à la méthode de la simplicité.

Ch. 12 Le concept de la *vie artificielle* pour une *sémantique de l' action*

(État de l'art : introduire la vie artificielle (VA) pour obtenir une sémantique de l'action)

1) Introduire la vie artificielle pour obtenir une sémantique de l'action

Dans le chapitre sur les *concepts de base utilisés dans notre travail*, par deux fois nous avons pointé l'intérêt de la VA : d'abord, l'introduction de la sanction létale introduit une tension chez un être et les procès qu'il effectue peuvent être appelés actions. Ensuite nous avons vu qu'introduire les critères de la VA dans une simulation permet de lui fournir une sémantique, maintenant nous allons aborder plus précisément la VA et détailler ce dernier aspect :

Critère direct d'efficacité.
- Celui qui est adapté à trouver de la nourriture voit sa réserve d'énergie augmenter vite.

Critère indirect d'efficacité (par effet sommateur de la réserve d'énergie).

- Action de sommation	: celui qui mange de la nourriture voit sa réserve d'énergie augmenter.
- Action de soustraction	: celui qui agit consomme de l'énergie et voit sa réserve diminuer.
- Comparaison à un seuil bas	: s'il elle passe au-dessous du seuil létal, l'être disparaît.
- Comparaison à un seuil haut	: si la réserve d'énergie d'un agent passe au-dessus du seuil de félicité, l'être se reproduit.

2) État de l'art : Les trois mouvances de la VA (ALife en anglais)

La VA s'intéresse aux mécanismes des systèmes vivants reproduits dans des environnements artificiels. Pour nous elle ne constitue par une fin mais seulement un moyen. De façon éclectique nous choisissons dans ses mouvances les aspects qui aident notre recherche.

2.A) La VA chimique (Wet ALife)

Elle traite de la VA sur le plan chimique. Nous choisissons de ne pas traiter cette mouvance.

2.B) La VA matérielle (Hard ALife)

Elle traite de la VA sur le plan du matériel, de la robotique (hardware). Nous choisissons de ne pas traiter cette mouvance.

2.C) La VA du logiciel (Soft ALife)

a) Reproduction par simulation de la VA

Reproduction totale stricte.
Notre but n'est pas de trouver des mécanismes qui reproduiraient entièrement le cycle de la VA tel que le fit Christopher Langton avec les *Boucles de Langton* [Lan 1984].

Reproduction éclectique partielle.
Nous avons retenu certains mécanismes de la VA utilisables dans notre travail :

- Nous utilisons l'illustration des automates cellulaires pour montrer comment ils peuvent servir de substrat aux êtres d'une de nos simulations.

- Nous avons exploré la partie morphogénèse des êtres que nous plaçons dans nos simulations, sans arriver à des résultats probants, alors, dans notre partie prospectives, nous signalons cette direction à éventuellement continuer d'explorer.

b) Différents types de simulateurs utilisés

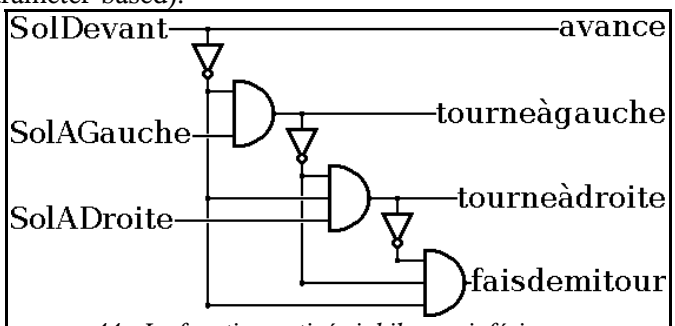
Parmi les différents types de simulateurs utilisés en VA nous retenons plusieurs démarches et les implémentons dans certaines de nos simulations.

Les êtres modulaires (Module-based).

La structure des êtres est modulaire : elle est constituée d'un assemblage de modules de base. Cette organisation est intéressante car on peut être tenté d'associer un module à une fonction et à un verbe (d'action ou de perception).

Les êtres basés sur des paramètres (Parameter-based).

Le fonctionnement des êtres est réglable au moyen de paramètres ajustables. Cette organisation fait penser aux couches de neurones formels dont l'effet est pondéré par des coefficients.



44 - La fonction activée inhibe ses inférieures avec son inverseur

Par exemple, l'agent fourrageur possède une organisation en couches hiérarchiques, où les fonctions prédominantes inhibent les autres. Cette inhibition peut s'effectuer au moyen de coefficients de blocage fournis par les couches supérieures, aux couches subalternes.

Dans l'illustration ci-contre la hiérarchie des fonction est verticale descendante. La première fonction supérieure qui est activée, inhibe ses inférieures au moyen d'une fonction négation (un inverseur).

Les êtres basés sur un réseau de neurones (Neural netbased).

Cette organisation fait penser à la grande mouvance des neurones formels, organisés en couches, mais surtout en réseaux. Nous traitons quelques exemples d'apprentissage simples au moyen de neurones.

3) Le concept de VA pour implémenter une sémantique de l'efficacité

Les mécanismes inhérents à la vie artificielle permettent d'introduire une évaluation de l'efficacité des agents sur deux points différents :

3.A) La nourriture, une première évaluation (directe) de l'agent

Évaluation de l'adresse de l'agent à trouver de la nourriture.

Le fait que l'agent mange un repas constitue un premier outil pour évaluer son aptitude à chercher et trouver de la nourriture. En effet, quand l'agent se débrouille bien, il réussit à se déplacer, à trouver et à manger de la nourriture. Le plaisir qu'il y trouve constitue la sanction directe de ses aptitudes de chasseur-cueilleur.

Évaluation du coût des actions de l'agent situé.

Chaque action que l'agent exécute est sanctionnée par une consommation d'énergie. En fait cette comptabilité consiste en une évaluation directe du coût des actions que l'agent effectue en situation.

3.B) Le niveau d'énergie, une seconde évaluation (indirecte) de l'agent

Description du principe de fonctionnement de ce mécanisme d'évaluation.
Au fil de son activité, si l'agent est bien adapté à son milieu, il accumule plus d'énergie qu'il n'en brûle pour sa quête de nourriture.

Comparer le niveau d'énergie à un seuil létal.
Dans ce cadre, simuler le mécanisme de la mort consiste à comparer ce niveau d'énergie à un niveau bas minimal correspondant à un seuil létal. Quand l'agent passe dessous cette limite, il meurt par inanition.

Comparer le niveau d'énergie à un seuil de félicité.
De même, simuler le mécanisme de la vie consiste à comparer ce niveau d'énergie à un niveau haut maximal qui correspond à un seuil de félicité, de reproduction. Quand l'agent passe au-dessus de cette limite, il se reproduit et donne naissance à un nouvel agent.

Une évaluation indirecte à cause de l'effet sommateur du stockage d'énergie.
Nous gérons le stock d'énergie de l'agent en additionnant les prises de nourriture, et en soustrayant ses dépenses d'énergie. Ainsi nous sommes en présence d'une évaluation indirecte de ses aptitudes. En effet, la mise en réserve de l'énergie permet un effet de lissage : si pendant une période de disette, ce dernier ne trouve pas de nourriture, il peut puiser dans ses réserves afin de jeûner quelques temps, et finalement survivre.

3.C) L'évaluation du niveau d'énergie donne une sémantique indirecte à l'agent

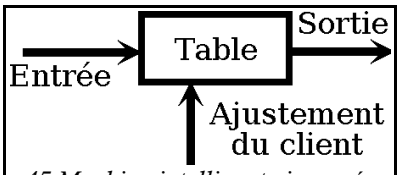
Une évaluation qui donne un sens au comportement de l'agent.
Regardons une simulation où les nombreux êtres sont placés en situation de compétition pour survivre. Si on raisonne avec le bon sens de l'homme de la rue, nous pouvons dire qu'un agent qui survit dispose d'un comportement adapté. Nous concluons que la mesure du niveau d'énergie de l'agent constitue une évaluation qui mesure si son comportement est *sensé*. En jouant sur le mot nous concluons qu'elle attribue, qu'elle donne un sens au comportement de l'agent.

Aspect épistémologie : le passage du procès à l'action.
Le besoin d'énergie sous-tend l'activité de l'agent qui est donc finaliste. En conclusion, selon un point de vue épistémologique, les *procès* qu'il effectue sur le monde sont sous-tendus par une finalité indirecte, et peuvent être appelés *actions*.

4) Donner un sens à la vie, donner une vie au sens

4.A) Exemple d'une machine intelligente insensée

La machine de Wilkes [Wil 1956] ci-contre reçoit en entrée sur N bits l'information binaire à traiter, elle fournit en sortie sur un bit le traitement à effectuer, i.e. la commande de l'actionneur à piloter. Pour cela, l'entrée est interprétée comme une adresse qui est fournie à une table, une mémoire de 2^N x 1 bit, laquelle fournit son contenu en sortie.



45 Machine intelligente insensée

Si on retient la définition de l'intelligence comme *l'aptitude à s'adapter à une situation nouvelle*, cette machine est intelligente car, nous l'avons améliorée pour qu'elle adapte son traitement. En effet, à chaque fois que le testeur refuse le fonctionnement courant en protestant sur l'entrée *ajustement du client*, elle génère le remplissage suivant de sa table, et l'utilise pour effectuer son travail. Puisqu'elle sait générer systématiquement tous le remplissages possibles, elle finit par proposer le bon, i.e. par satisfaire le client, et donc par trouver la solution au problème qu'on lui pose.

Cependant, cette machine est insensée, car elle n'a pas accès au sens. Le sens de son fonctionnement provient seulement de son testeur qui est le seul à pouvoir apprécier la justesse de son comportement.

En conclusion, nous sommes bien en présence d'une machine intelligente qui n'a pas de sens.

4.B) La VA pour donner un sens aux simulations

Ainsi cet exemple de la *machine intelligente insensée* nous permet de mieux comprendre comment l'introduction de la VA dans une simulation peut aussi donner un sens à son fonctionnement.

4.C) Des simulations qui n'ont pas de sens, la VA assure un service minimal

Donc, si une simulation n'a pas de sens, i.e. ne possède pas une fonction qui lui permet d'évaluer la justesse de son comportement, il est difficile d'y faire de l'apprentissage.

Ainsi, faire fonctionner un système dans le cadre de la VA, consiste à lui fournir un sens assez simple, mais minimum, qui permet d'y faire de l'apprentissage, sans devoir lui fournir un sens externe.

En conclusion, nous pouvons utiliser la VA, pour fournir à un système un sens assez rudimentaire, mais suffisant. Dans ce cas, il possède un sens minimum, qui nous permet d'y effectuer des apprentissages, sans que nous devions lui fournir un sens, i.e. un ajustement externe.

5) Les conclusions à tirer sur la structure des simulateurs sensés

Au sein d'un simulateur, nous avons besoin de l'énergie pour donner un sens au procès des agents ; ainsi il apparaît comme une action.

Le niveau d'énergie d'un agent permet d'évaluer la justesse de son comportement, i.e. il permet d'évaluer s'il est sensé, s'il possède un sens.

La mesure de l'énergie de l'agent peut aussi servir d'outil pour évaluer la qualité de la réponse du support à ses actions, à son comportement.

5.A) Mesurer le niveau d'énergie des agents, l'outil pour évaluer un simulateur

Dans le chapitre précédent, nous décomposons le simulateur en deux composantes : l'action effectuée par l'agent situé en fonction de l'état du monde, et la réponse instantanée du support à l'action de l'agent.

Les agents, mis en situation dans le monde recherchent leur nourriture sur le support. Quand ils mangent ils emmagasinent de l'énergie qui leur sert à agir. Ainsi, en mesurant cette grandeur, nous disposons de la somme de deux évaluations : l'adaptation du comportement des êtres à leur milieu et la générosité du milieu envers les êtres qui l'habitent.

La mesure du niveau d'énergie peut servir à évaluer la qualité du comportement de l'agent situé dans le monde.

Le critère de l'énergie, i.e. la mesure du niveau de glucide disponible chez chaque agent, peut fournir une évaluation différée de l'efficacité de ses actions, de son comportement, dans le monde. Pour cela, il faut que la réponse du support soit isotrope, et que nous utilisions le lissage statistique : nous faisons des mesures espacées dans le temps et retenons leur valeur moyenne.

La mesure de l'énergie peut servir d'outil pour évaluer la qualité de la réponse du support à l' action de l'agent.

Une méthode simple pour évaluer la qualité du milieu où vit l'agent est de mesurer la réserve d'énergie disponible chez ce dernier. Pour affiner cette mesure nous pouvons la lisser, i.e. l'effectuer plusieurs fois et calculer la moyenne. On reconnaît un milieu accueillant au fait qu'il nourrit de nombreux êtres de plusieurs espèces.

Si la réponse du support n'est pas isotrope, alors on peut tomber sur la notion de niche écologique. Ce phénomène se produit quand la réponse du milieu est spécifique à un lieu donné. Dans ce cas, l'espèce la plus spécialement adaptée à ce biotope s'y développe plus particulièrement.

Cette considération peut aider à évaluer la qualité de la réponse du support. Ainsi, en regardant la mesure de la réserve d'énergie disponible chez chaque agent nous pouvons remarquer qu'elle varie en fonction des lieux. Alors nous conclurons que le milieu de la simulation est anisotrope.

Ch. 13 Les concepts de base de notre travail

1) Introduction : présenter et détailler les concepts de base de notre travail

Au cours du chapitre *l'histoire de l'art*, nous avons évoqué des concepts qui vont orienter notre travail et maintenant nous allons les détailler plus précisément.

2) Le finalisme : un concept nécessaire pour parler d'action sur le monde

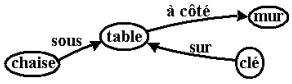
2.A) Définir le concept d'*action* selon le point de vue de la *philosophie de l'action*

Quand un procès est sous-tendu par un but, il devient une action, i.e. qu'on peut le qualifier ainsi.

2.B) Rappels des conventions avancées précédemment :

Nous utilisons une version réduite de la logique pour décrire l'état du monde.

Pour ce faire, à l'instar de la logique, nous représentons les objets du monde au moyen de constantes, et listons leurs relations quand elles sont vraies.

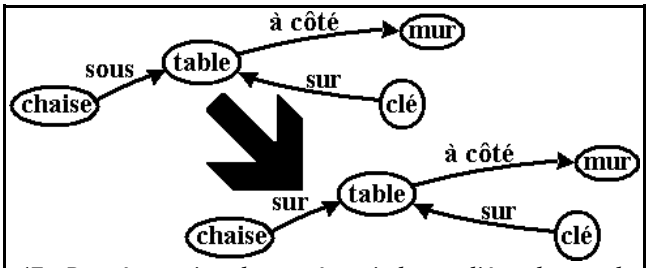


46 - Scène représentée au moyen d'un graphe

Exemple : la scène *S* ci-contre, représentée au moyen d'un graphe, se transcode en logique :
S= sous(Chaise, Table) ∧ àcôté(Table,Mur) ∧ sur (Table,Clé)

La variation de relations pour représenter le procès.

Nous représentons le procès, par un changement d'état du monde, i.e. le changement d'une (ou plusieurs) de ses relations entre ses objets. Donc la représentation d'un procès se fait simplement en changeant une (ou plusieurs) des relations entre les composantes du monde. Par exemple, dans l'exemple ci-contre, la chaise qui était sous la table, passe sur la table.



47 - Représentation du procès qui change l'état du monde

2.C) Selon la philosophie, le *procès* sous-tendu par un but devient une *action*

a) Le point de vue de la philosophie de l'action

Selon le point de vue de la philosophie de l'action, le procès qui transforme l'état du monde doit être sous-tendu par un but pour qu'on puisse le nommer action.

Basic action theory typically describes action as behavior caused by an agent in a particular situation. The agent's desires and beliefs (e.g. my wanting a glass of water and believing the clear liquid in the cup in front of me is water) lead to bodily behavior (e.g. reaching over for the glass). In the simple theory (see Donald Davidson), the desire and belief jointly cause the action. Michael Bratman has raised problems for such a view and argued that we should take the concept of intention as basic and not analyzable into beliefs and desires.

Tableau 2 : Principes de la philosophie de l'action (Wikipédia).

b) Comment composer avec ces contraintes de causalité ?

La réponse de l'IA symbolique à cette objection.

L'IA symbolique répond à ces contraintes au moyen notamment de la planification. Les tactiques ou stratégies générées dans ce cadre sont manifestement finalistes. En conséquence, nous pouvons considérer qu'elles sont composées d'actions.

La réponse de la vie artificielle à cette objection.

Hors du cadre de l'IA symbolique, la réponse à cette objection est fournie la l'opération d'évaluation implémentée dans les simulations de la vie artificielle : un programme simule des êtres en situation de survie dans un cadre où ils cherchent leur nourriture. La justesse de leur comportement, i.e. leur aptitude à trouver de l'énergie est évaluée indirectement et détermine leurs chances de survie. Donc, dans ce cadre, leurs procès sont indirectement sous-tendues par cette contrainte : rechercher sa subsistance ; et, pour cette raison, on peut les appeler *actions*.

3) Nouveau concept de sens : *évaluer le sens d'une fonctionnalité au sein d'un être*

3.A) Introduire le concept : *la sémantique vue comme une fonction de transfert*

La sémantique en logique : une fonction de transfert digitale.

Selon les mathématiques, par définition, la sémantique d'une fonction logique se caractérise par la donnée de sa table de vérité. Ceci correspond à la donnée d'une fonction de transfert digitale i.e. à valeur dans {0, 1}.

La sémantique en analogique : une fonction de transfert analogique.

Ainsi, en transposant cette définition, de la sémantique d'une fonction logique vers le domaine de l'analogique, on conclut que la sémantique d'une fonction analogique correspond à sa fonction de transfert.

À la recherche d'une sémantique qui ait du sens.

Dans les paragraphes qui suivent, nous travaillons à décrire un protocole d'évaluation de cette fonction de transfert en la restreignant à un critère. Cette méthode permet d'évaluer, soit la fonction de transfert restreinte d'un être situé, soit d'une des composantes de son corps (de sa structure).

La conclusion que nous tirerons de cette méthode, c'est que :

- Elle satisfait aux critères de la philosophie de l'action car elle est indirectement sous-tendue par un but.
- Elle correspond au sens commun.

3.B) Appliquer le concept : *la sémantique vue comme une fonction de transfert*

La vie artificielle évalue aussi le synoptique de l'être.

En fait, la vie artificielle est utilisée comme outil normatif : quand elle teste un être en situation dans le monde, elle évalue la justesse du système entier. Et si ce système est modulaire i.e. constitué de couches, de modules empilées selon une structure mixte ET/OU, chacun des modules intervient plus ou moins dans le résultat de ce test.

Évaluer chacun des modules d'un être.

Par exemple, si G_d la fonction globale de l'être est digitale et s'écrit : $G_d = A \wedge B \wedge (C \vee D)$, ou pareillement, si G_a est une fonction analogique et s'écrit $G_a = A*B*(C+D)$, alors, dans les deux cas :

- Les couches, les modules A et B interviennent directement dans le résultat final, si ils sont défaillants alors l'être est défaillant.
- Mais les modules C et D, qui fonctionnent en parallèle, sont plus difficilement évaluables. Si C est défaillant alors on évalue facilement D, mais si elle fonctionne très bien, elle masque totalement D (et réciproquement).

Donc, selon l'architecture, la structure de l'être, ses modules sont plus ou moins évaluables séparément.

Évaluation différentielle d'un être.

Partons d'un être E. En le mettant en situation dans le monde, nous l'évaluons : au bout du temps T la taille de la population initiale P_0 est devenue P_1. En calculant $S = P_1 - P_0$ on obtient une valeur S assimilable au sens de l'être. Par exemple :

- Si S= 10, alors la population de ce groupe d'êtres augmente, et cet être a un sens.
- Si S = -10, alors la population de ce groupe d'êtres diminue, et cet être n'a pas de sens (il va dans le mauvais sens).

Ainsi nous disposons d'une méthode pour évaluer le sens d'un être, i.e. de son adaptation au monde[19].

Évaluation différentielle du module d'un être.

Au sein de l'architecture d'un être, pour évaluer la justesse d'une nouvelle couche, d'un nouveau module, nous pouvons utiliser le critère de la vie artificielle, par exemple voir au bout du temps T quelle taille P_{N1} atteint la population initiale P_{N0} quand elle est munie de ce nouveau module. En calculant $S_N = P_{N1}.- P_{N0}$ nous obtenons une valeur S_N assimilable au sens de l'être. Par exemple :

- Si S_N= 10, alors la population de ce groupe d'êtres nouvellement équipés augmente, et ce nouveau module a un sens.
- Si S_N = -10 alors la population de ce groupe d'êtres nouvellement équipés diminue, et ce nouveau module n'a pas de sens.

Ainsi nous disposons d'une méthode pour évaluer le sens d'un nouveau module d'un être, i.e. de son adaptation au monde.

3.C) Conclusion : une sémantique qui possède du sens commun

À l'issue de l'évaluation de la sémantique d'une fonction ou d'un module, nous constatons une convergence entre le protocole expérimental et le jugement de sens commun. Là où la quantification expérimentale retourne, rend un vecteur orienté dans la bonne direction et possédant un module important, alors nous constatons que le sens commun leur attribut volontiers du sens.

4) Des concepts rattachés à la vie ou l'évolution artificielle

Continuons d'introduire les concepts qui orientent et éclairent notre démarche. En référence à cet état de l'art à propos de cette mouvance de la vie ou de l'évolution artificielle, introduisons trois concepts que nous utiliserons par la suite : émergence, évolution et co-évolution.

Trois étapes d'émergence, d'évolution et de co-évolution.

Nous travaillons sur une progression de simulations d'êtres situés dans le monde. Nous choisissons de la découper en trois étapes : 1, 2 et 3.

À l'étape 1, cette taxinomie commence avec la cellule végétative, qui est remplacée par la bactérie se reproduisant par mitose. Dans cette première étape qui correspond à une émergence, le seul souci des êtres est de se reproduire, car la ressource du monde semble inépuisable.

19 En anglais on dit : *fitness to the world.*

Ensuite, la population prolifère exponentiellement, et durant cette étape de diaspora, elle colonise son milieu au point de le saturer. Finalement, quand elle atteint les confins de son monde, cette émergence est bloquée par manque d'énergie. Alors survient une crise de surpopulation qui induit une transition et marque la fin de l'étape 1.

À l'étape 2, dans ce milieu saturé et en crise, les êtres sont en compétition lourde pour consommer les ressources. Ils sont soumis à une grosse pression de sélection : pour survivre, ils doivent évoluer. Ainsi apparaît le concept d'évolution.

À l'étape 3, d'abord apparaissent des niches écologiques : des agents rationnels cognitifs adaptables sont placés dans un monde anisotrope[20]. Une espèce spécialisée apparaît dans chaque recoin du monde : elle s'est adaptée aux conditions locales. Ensuite, se produit une co-évolution entre le groupe et la niche. Elle peut être observée quand les agents ne survivent que dans certaines zones (évolution surtout du groupe), ou bien quand ils agissent sur leur environnement et modifient leur monde (évolution surtout du monde).

D'abord, une émergence, une évolution de la structure des êtres, ensuite de leurs règles de comportement.

D'abord, principalement dans les étapes 1 et 2, nous observons une émergence, une évolution de la structure des êtres, qui est codée dans l'inné, i.e. dans la génétique. Ensuite, surtout dans l'étape 3, quand apparaît le groupe, nous observons une émergence de la communication et du langage.

- Avec la cognition partagée, la communication permet l'échange d'informations non verbales, à propos du monde.
- Le langage permet toujours aux agents d'échanger des informations décrivant le monde, mais, et c'est nouveau, de transmettre leur culture qui code leurs règles de conduite, leur comportement.

4.A) Étape 1 de l'émergence du groupe

Si le grain ne meurt.

Nous nous situons dans le cadre de la vie artificielle. Dès que le grain qui se reproduit ne meurt pas, nous assistons immédiatement à sa multiplication : la population progresse et se répartit dans le monde. C'est l'étape de la diaspora.

Le parallélisme pour implémenter une simulation informatique de la diaspora.

En conséquence, pour simuler ce phénomène au moyen d'un programme informatique, nous devons utiliser le parallélisme, outil nécessaire pour simuler la vie du groupe.

Transition : la saturation du milieu.

La diaspora produit une vague de peuplement, une onde isotrope qui se propage jusqu'aux confins du monde où elle rebondit. Alors, à ce moment, le domaine est saturé.

4.B) Étape 2 de l'évolution de la structure

a) La pression de sélection accélère l'évolution

Dans ce domaine saturé apparaît une crise de surpopulation : au minimum les êtres sont en compétition pour survivre, et ça peut aller jusqu'à la lutte pour l'existence[21]. Alors, la saturation du domaine induit la transition de la pression de sélection. Pour survivre, les êtres doivent évoluer. Ainsi apparaît le concept d'évolution.

b) Brader à vil coût l'ambition des mathématiques

Avant propos : nous nous plaçons selon un point de vue qualitatif et regardons l'adaptation fonctionnelle des êtres à leur milieu, celui de la vie artificielle. L'émergence de la structure se produit à l'étape 1, mais surtout aux étapes 2 et 3.

20 Un monde anisotrope : un monde dont les caractéristiques varient en fonction du lieu, i.e. en fonction de l'espace.

21 Struggle for the life (selon Charles Darwin).

Tout d'abord, cette démarche d'émergence de la structure est loin de la perfection des mathématiques : nous ne démontrons pas un théorème, nous ne recherchons pas un optimum. Nous mettons la barre bien moins haut.

En fait, la démarche d'émergence ne consiste même pas dans la recherche d'un maximum, même local, elle cherche seulement à améliorer l'efficacité de l'être. Cette démarche est à rapprocher de la montée de gradient en optimisation.

Pourquoi mettre la barre si bas ? Car durant ces trois étapes, les structures émergent du chaos et disposent facilement d'énergie. Elles partent de très bas. Il suffit à l'être de s'améliorer pour en tirer un avantage fonctionnel.

c) La cognition partagée, distribuée

Introduction : notion d'effet d'horizon et de *rationalité limitée* (Herbert Simon).

Au début de cette émergence, les êtres sont immatures : ils sont loin d'être fonctionnellement parfaits. Ils sont plutôt soumis à un effet d'horizon : la perception de leur monde est locale i.e. imparfaite. Alors leur rationalité, qui raisonne sur des bases limitées, est elle aussi, imparfaite. Nous débouchons sur la notion de *rationalité limité* de Herbert Simon [Sim 1947].

C'est pourquoi, dans ces conditions, le groupe d'êtres trouve un intérêt à échanger. Ainsi la démarche de partage des informations i.e. la communication constitue un avantage culturel que nous regroupons dans le concept de cognition distribuée. Nous tombons sur la notion de *cognition partagée*.

Remarque : ne pas confondre cognition partagée et traitement rationnel cognitif.

La cognition partagée, distribuée est très légère et apparaît très tôt dans la phylogenèse. Par exemple, les fourmis, qui échangent des informations au moyen de phéromones utilisées comme marqueurs, la pratiquent déjà. C'est pourquoi nous avons pris soin de la définir afin que le lecteur ne la confondent pas avec le *traitement rationnel cognitif* effectué par l'agent CDI rationnel, qui lui est d'une grande complexité.

d) La cognition partagée (CP) plaide pour le concept de parcours en largeur d'abord

L'efficacité de la cognition partagée, constitue un exemple qui montre l'éclectisme et l'imagination de la nature (ex : la biodiversité). Dans le domaine de la conception de la structure des agents, l'efficacité de ces deux démarches plaide pour le concept de parcours en largeur d'abord.

Dans notre progression de simulations, la cognition partagée/distribué apparaît très tôt car elle procure au groupe qui la met en oeuvre, une efficacité qui la renforce et la confirme.

Selon le point de vue de la conception somatique de l'agent, la CP se construit au moyen d'un actionneur et de deux capteurs. Un senseur local perçoit le contenu de l'environnement. Un senseur et un effecteur globaux implémentent un système émetteur-récepteur de communication globale.

Le système qui implémente la CP est simple, mais il fallait y *penser*. La créativité de cet exemple illustre l'efficacité l'éclectisme de la nature : elle n'approfondit pas sa quête dans une direction, i.e. elle ne prolonge pas une branche d'arbre, mais prospecte plutôt tout azimut, elle choisit et prend à droite et à gauche. Elle est éclectique.

Cette stratégie consiste aussi en une démarche de parcours en largeur d'abord. Nous concluons à une confirmation de l'intérêt de la notion de parcours en largeur d'abord.

e) Émergence vers des attracteurs

Apparition de la communication et de son traitement.

En affirmant que la communication constitue un avantage culturel et que selon le point de vue de l'évolution, elle constitue un attracteur, nous faisons un premier pas en direction de Chomsky, mais à ce stade nous sommes encore loin de sa théorie qui dit que les mécanismes du langage sont initialement câblés dans le cerveau, par la morphogénèse.

La communication se fait au moyen de protocoles, lesquels relèvent de la co-évolution que nous traiterons ci-après.

La culture court-circuite la génétique.

L'évolution culturelle continue l'évolution biologique par d'autres moyens (Karl Popper, 1902-1994). Quand les échanges d'informations entre agents portent sur des règles épigénétiques, la communication devient culturelle. Les échanges culturels entre générations constituent un court-circuit de la génétique. À ce titre ils acquièrent une grande efficacité qui leur confère le statut d'attracteurs.

4.C) Étape 3 : concept de la co-évolution

a) Définition du principe de co-évolution

Définition de la co-évolution basée sur l'étymologie.

Pour définir la notion de co-évolution servons-nous de l'étymologie. Ce terme vient de deux mots latins signifiant *évoluer ensemble*. Nous choisissons de la traiter comme un couple au sein duquel se déroule un protocole : cela consiste en une séquence d'évolutions effectuées en collaboration, i.e. alternativement par l'un puis l'autre. Ce sont des opérations séquentielles où, classiquement en informatique, exécuter les actions dans le bon ordre est important.

b) Trois cas de figure pour l'illustrer le principe de co-évolution

Après avoir défini la co-évolution, nous choisissons de l'illustrer par trois types de couples :

La co-évolution du couple *groupe* ↔ *lieu* débouche sur la notion de niche écologique

Tout d'abord nous regardons l'exemple *groupe* ↔ *niche écologique* : à l'étape 3, dans des simulations très riches, d'agents rationnels dans un monde dont les caractéristiques varient en fonction de l'espace, apparaît d'abord la notion de niche écologique. Ensuite, quand les agents agissent sur leur environnement et modifient cette niche, alors apparaît le concept de couplage, de co-évolution entre le groupe et la niche.

Le couplage *agent* ↔ *lieu* débouche sur des transferts matériels ou culturels

Transition : au sein du couplage *groupe* ↔ *lieu* apparaissent plutôt des séquences d'interaction *agent* ↔ *lieu*. Comme le groupe est faiblement organisé, la probabilité de voir émerger au sein du couplage *groupe* ↔ *lieu*, une séquence d'interaction *groupe* ↔ *lieu*, est très faible. Par contre, Naturellement apparaissent quelques relations *agent* ↔ *lieu* dont chacune peut évoluer vers une interaction séquentielle entre l'agent et son environnement.

Basiquement, ces interactions *agent* ↔ *lieu* fonctionnent ainsi : l'observation du milieu déclenche une action de l'agent qui modifie le milieu. Et le cycle recommence. Le comportement de l'agent peut se décrire ainsi : en interaction avec le milieu, il déroule une séquence d'actions circonstanciées.

Dans le cadre de ce fonctionnement *agent* ↔ *lieu* en interaction avec l'environnement, nous obtenons plusieurs fonctionnement notables : la séquence de fabrication, la transmission culturelle, et le traitement de l'information.

Exemple 1 : le couple *ouvrier* ↔ *séquence d'assemblage.* Finalement nous obtenons un fonctionnement qui ressemble à celui d'une *chaîne d'assemblage* d'un bien de consommation : l'agent suit la chaîne et à chaque poste, en fonction de l'environnement, effectue le traitement nécessaire.

Exemple 2 : le couple *client* ↔ *prestataire* : le client se rend dans le local où travail un prestataire qui lui fournit un service. Nous retrouvons encore ici la notion de *script* et de *scénario* de Schank et Abelson [SA 1977], et de façon plus générale une partie des comportement qui relève de la *pragmatique*.

Exemple 3 : le couple *enfant* ↔ *circuit d'initiation* : la transmission culturelle. L'enfant suit un circuit d'initiation, constitué de lieux dédiés, qui sert à leur éducation. La transmission se fait en marquant le paysage : nous sommes déjà en présence d'une transmission culturelle.

Exemple 4 : le couple *interprète* ↔ *marques à interpréter* : le traitement de l'information. Ce fonctionnement consiste à mettre des marqueurs sur le monde, et à les interpréter afin de décider chacune des opérations à effectuer sur l'information. Ainsi la puissance de traitement des agents réactifs est accrue. En informatique théorique on démontre que si un agent réactif peut placer et

percevoir des marques sur le support, il récupère la puissance de traitement d'une machine universelle de Turing.

La co-évolution du couple *agent↔agent* débouche sur des protocoles de coopération ou d'éducation.

Dès qu'apparaît le groupe, l'agent perçoit les autres agents et rapidement il doit apprendre à déclencher les actions puis le comportement adéquats. Comme les autres font pareil et que tous sont en interaction dans le même environnement, ceci suffit pour commencer à fonctionner selon le registre du protocole.

De même que pour la cognition partagée/distribué/répartie, ici encore, ces comportements, qu'on pourrait penser complexes, apparaissent très tôt et, au départ, ils sont très simples.

Exemple 1 : le couple *agent↔agent* : une coopération de deux agents partenaires dans le cadre de la vie quotidienne.

Exemple 2 : le couple *parent↔enfant* : dans la relation d'éducation *parent↔enfant*, pendant la croissance de l'enfant, certaines interventions des parents ou du milieu viennent fort à propos et déterminent son évolution.

c) Dualité de l'inné et de l'acquis : quel mécanisme a généré ces protocoles ?

Au début, ce protocole est exécuté conjointement par les deux partenaires.

De plus, il faut penser que ce protocole est distribué entre les deux partenaires et que donc, chacun en exécute la moitié : il est exécuté en commun :
- Ils partagent un environnement qu'ils perçoivent en commun.
- Mis à part le différent point de vue qu'ils ont sur le monde, ils analysent les mêmes perceptions pour déclencher leurs actions.
- Leurs actions transforment l'état de l'environnement qu'ils partagent.

Quel mécanisme a généré chacun de ces protocoles ?

À ce stade, nous sommes en présence d'un mécanisme efficace, nous posons la question : qui a généré chacun de ces protocoles ?

Au début, ce protocole est codé génétiquement

Au début la transmission d'information entre générations n'est pas culturelle, elle est seulement génétique. Donc l'information est codée par la morphogénèse. Alors ce protocole est codé de façon déterministe dans une structure implémentée au sein de l'agent, i.e. dans une structure *perception décision action*. Il est instinctif.

Ensuite ce protocole incorpore des transmissions culturelles.

Ce protocole, initialement transmis par la morphogénèse, incorpore ensuite des transmissions culturelles. Il peut aussi être acquis par apprentissage, par apprentissage guidé et par transmission langagière.

C'est donc un comportement qui est codé dans la culture, laquelle court-circuite la morphogénèse.

Ensuite, progressivement, ce protocole incorpore du code relatif à un contexte qui s'enrichit.

Ensuite, dans cette évolution, le contexte spatio-temporel s'enrichit et intervient de plus en plus au cours de la séquence. Progressivement, ce protocole incorpore du code relatif à un contexte qui s'enrichit. Nous allons vers une plus grande diversité. Un deuxième intérêt de ce protocole est qu'il code en fonction du contexte, des traitements très efficaces, par exemple la communication, la cognition partagée.

Prenons l'exemple d'une séance d'initiation : elle se produit souvent dans un lieu consacré, et l'ordre de la cérémonie a certainement de l'importance.

Un troisième intérêt de ce protocole est qu'il code les parcours situés. En apprenant très peu d'instructions instinctives, mais en exécutant ce protocole en situation, l'être effectue un parcours conséquent, car tout est codé dans le contexte, i.e. dans le monde.

Ensuite, progressivement, ce protocole incorpore du code relatif à la chronologie

Pour ce qui est du temporel, nous pouvons parler de la séquence qui intervient au bon moment : pendant la croissance de l'enfant, certaines apprentissages doivent être effectués au

bon moment sinon c'est trop tard. Par exemple, un enfant qui n'est pas soumis à la lumière devient aveugle. Dans le même registre voir les histoires des enfant-loup.

d) La pragmatique

Définition de la pragmatique.

Notre définition de la pragmatique est un peu éloignée de celle que donne la linguistique, car nous préférons la définir en termes d'IA, i.e. de planification dans un monde riche. Quand un groupe social doit générer le plan d'une tâche récurrente dans le contexte d'une grande diversité du mobilier, cette richesse induit un grand éventail de solutions : nous sommes dans le cas d'un problème peu contraint, où les solutions abondent, alors, dans le contexte donné, il convient d'en choisir une, arbitrairement, et de s'y tenir.

La pragmatique d'un groupe donné consiste en un ensemble de solutions comportementales.

En conclusion, une définition simplifiée de la pragmatique d'un groupe consiste en l'ensemble de ses comportements sociaux, sachant que chacun est convenu pour une situation donnée.

De plus, il ressort de cette définition, et c'est là son intérêt, que les protocoles appartiennent à la pragmatique car chacun consiste bien en un comportement social convenu pour une situation donnée.

Au début, la pragmatique est élaborée, codée et transmise génétiquement.

C'est le cas des protocoles, qui relèvent de la pragmatique. Chez les premières créatures, quand la transmission est seulement génétique, ils sont élaborés, codés et transmis génétiquement

Chez les êtres les plus primaires, ces solutions sont élaborées par émergence du patrimoine génétique qui est codé génétiquement. Il est décodé par la morphogénèse qui construit les connexions nerveuses du cerveau. Dans ce cas, la pragmatique est transmise génétiquement, car c'est le seul média possible.

La pragmatique, initialement transmise par la morphogénèse, incorpore ensuite des transmissions culturelles.

Si initialement la pragmatique est codée génétiquement, ensuite elle incorpore une composante élaborée par le système, laquelle est codée et transmise par des marqueurs sur le support.

De par sa nature, la pragmatique qui émerge est transmise au fil des générations. En effet, elle correspond à une convention sociale, élaborée au hasard ou empiriquement, et d'autres (personnes ou groupes) pourraient générer ou choisir des solutions différentes, plus ou moins aussi efficaces.

Dans cette démarche d'évolution, d'émergence, comment se fait le renforcement d'une variation réussie ?

Un des intérêts de la pragmatique est qu'elle code les règles sociales de coopération entre les êtres. Elle codifie la vie de groupe.

Quand le groupe cherche à améliorer un de ses comportements sociaux (et culturels), il effectue des variation autour de son fonctionnement actuel. Alors, s'il trouve une meilleure solution, elle n'est pas forcément optimale, mais si situe plus profond dans le bassin d'attraction d'un attracteur.

Comme l'union fait la force, quand le groupe fonctionne selon ce comportement pragmatique amélioré, il est statistiquement plus efficace. Donc le groupe culturel modifié possède plus de chances d'être sélectionné, ainsi le nouveau couple *groupe-comportement* est renforcé.

5) Du concepts de *symbole* à celui de *sub-symbole*

Décrivons le passage du concept de *symbole* à celui de *sub-symbole* : deux concepts de représentation et traitement cognitifs

Comme leur nom l'indique, à la base des systèmes formels, nous trouvons le terme *forme* qui renvoie à la notion de symbole, c'est pourquoi nous allons préciser cette notion.

En ce qui concerne la représentation et le traitement cognitif, tout d'abord intéressons nous au concept classique de symbole formel, puis nous passerons à celui de sub-symbole plus proche de la représentation neuronale.

5.A) *Signe* et *forme* : deux aspects des symboles

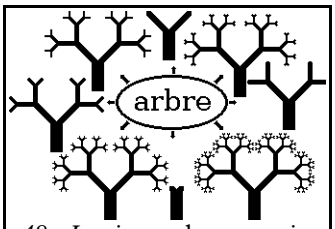
Les symboles présentent deux facettes indissociables : le signe et la forme. Le signe porte la *signification* et la forme (shape en anglais) correspond à sa topologie dans les trois dimensions de l'espace.

a) L'aspect signification d'un symbole

Au sein d'un symbole, le signe a pour rôle de porter le sens, la signification. C'est le contenu intentionnel[22] qu'il représente, la catégorie[23] vers lequel le signe renvoie.

b) L'aspect formel d'un symbole

Au sein d'un symbole, la forme tridimensionnelle sert à son décodage, i.e. à le reconnaître, à le distinguer parmi les autres.



48 - Le signe arbre *renvoie à la catégorie* arbre

Symbole vient de *symbolon.*

C'est le nom grec du morceau de bois que deux amis cassaient au moment de se quitter. Chacun gardait un bout de son côté, et quand ils se retrouvaient, ils symbolisaient leur amitié en joignant les deux parties.

Définition de la notion de symbole (Lexique de philosophie p.230 [Gra 1995]).

Du grec *sumbolon* : signe de reconnaissance entre deux personnes formé par un objet brisé en deux, et dont deux hôtes conservaient chacun une moitié, qu'ils léguaient à leurs enfants ; ceux-ci, en rapprochant les deux moitiés, reconnaissaient l'ancienne hospitalité des parents. D'où : *ce qui représente autre chose en vertu d'une correspondance analogique, manifeste ou cachée.*

Utilisation d'outil de comptabilité symbolique dans la France du siècle dernier[24].

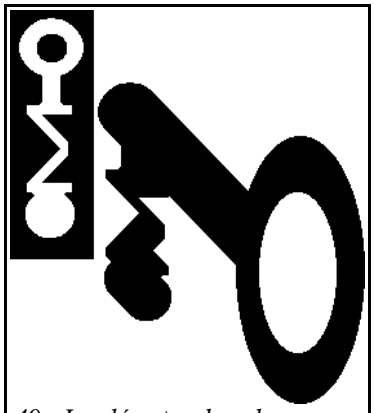
> Au grenier le blé attendait d'aller au moulin. Avant la guerre, nous faisions *l'échange*. Le boulanger recevait la farine et nous donnait le pain en échange. Les boulangers passaient dans les campagnes faire leur livraison plusieurs fois par semaine. Pour noter le nombre de pains reçus en *échange* de la farine, il ne se servait pas d'un crayon et d'un carnet mais d'une *règle*, détail qui vaut d'être raconté : la *règle* ou *coche*, était une baguette de bois d'environ cinquante centimètres de long. Fendue en deux, la moitié était accrochée dans le camion du boulanger, l'autre moitié chez le client. A chaque pain que nous prenions, il joignait les deux baguettes face à face, et faisait une entaille. Chacun reprenait sa moitié. Ainsi, pas d'erreur ou de tricherie possible.

22 Au sens linguistique.
23 La classe d'objets équivalents.
24 D'après les mémoires de Thérèse Lucas (La Payse). Communication personnelle.

5.B) Construction des symboles

Aspect formel des symboles.

En général dans les mathématiques, et en particulier dans un ordinateur, un symbole est comme la clé d'une serrure, il ne prend de sens que par sa forme. Donc, à la base de ce décodage, nous trouvons une opération de comparaison par rapport à deux seuils analogiques (le maximal et le minimal).



49 - La clé entre dans la serrure

En biologie, le principe de décodage d'une hormone est aussi formel. En fonction de sa géométrie elle vient s'encastrer dans un récepteur, comme une clé dans une serrure. Mais il faut faire attention, si cette étape est symbolique, tout le reste de la chaîne de traitement est analogique et donc sub-symbolique. En effet, quand le symbole est décodé, quand le niveau d'hormone dans le milieu dépasse un certain seuil, il commute le fonctionnement de son vecteur (de l'être qui le porte).

Aspect agrégatif des symboles.

Quand nous essayons d'ouvrir une porte au moyen d'une clé, chacune de ses dents doit être à la bonne taille. Il suffit qu'une seule comparaison basique échoue pour condamner l'opération globale.

En informatique, en base 10, quand le symbole *12* est comparé à *12*, nous comparons les 1 entre eux, et nous faisons de même pour les 2. Il suffit qu'une seule comparaison basique échoue pour conclure à la différence des symboles.

Ainsi un symbole est composé de plusieurs sous-symboles élémentaires, et la loi qui préside à cet assemblage est la fonction logique ET, notée $\wedge$: il faut que la comparaison 1-1 fonctionne, et que la comparaison 2-2 fonctionne, etc.

5.C) Définition de la notion de symbole par le dictionnaire (Lalande)

Maintenant, au travers des exemples précédents, nous comprenons mieux la définition suivante :

Ce qui représente autre chose en vertu d'une correspondance analogique.

6) *Symbole de l'informatique formelle* vs *sub-symbole du connexionisme*

6.A) *Symbole* vs *sub-symbole* à propos de *codage*, de *propagation* et *traitement*

a) Codage comparé *symbole* vs *sub-symbole*

Dans l'ordinateur de von Neumann, le symbole est codé formellement. Il l'est par sa forme, un peu comme la clé qui entre dans la serrure. Nous devrions plutôt dire par ses formes, car la forme du penne est codée de façon multiple : elle est codée par chacune des formes de ses multiples cannelures.

Le codage du sub-symbole est analogique et topologique. D'abord il est codé par une unique grandeur numérique ; ensuite par une approche topologique. En effet son codage est ainsi car il est propagé de façon connexe, tout au long d'un fil qui lui est dédié.

b) Propagation comparée des symboles :

La mobilité des symboles en informatique formelle.

Dans un ordinateur, nous pouvons faire passer plusieurs symboles sur le même bus, i.e. dans le même tuyau. Par exemple, un symbole qui est dans l'accumulateur peut être présenté au bus de données, puis mémorisé dans la mémoire vive. En effet, un symbole y est codé en parallèle sur plusieurs bits qui cheminent séparément, sur chacun des digits du bus. Ainsi, à l'autre bout du tuyau, nous obtenons chacune de ses composantes : nous disposons bien du même symbole. Ainsi, il s'est dupliqué, et une de ses instances a traversé le bus et est allé s'inscrire dans la RAM. Ce phénomène s'appelle *la mobilité des symboles*.

En toute rigueur nous devrions plutôt dire : *mobilité des instances de symbole*. Nous pouvons faire passer les instances de plusieurs symboles différents dans le même tuyau. En effet, prenons le cas d'un registre connecté par un but à un accumulateur. Si la mémoire fournit sur sa sortie la séquence (A, B, C) le bus est capable de transporter l'une après l'autre ces trois données. Ainsi, à l'autre bout du bus, nous obtenons la même information : nous avons fait passer les symboles A, B et dans le même bus.

Remarque : cette caractéristique de mobilité n'est pas réservée aux symboles digitaux de l'ordinateur, les mots du langage peuvent aussi se propager de bouche à oreilles. Il faut cependant noter que cette mobilité des symboles constitue le marqueur d'un traitement intentionnel[25] de l'information.

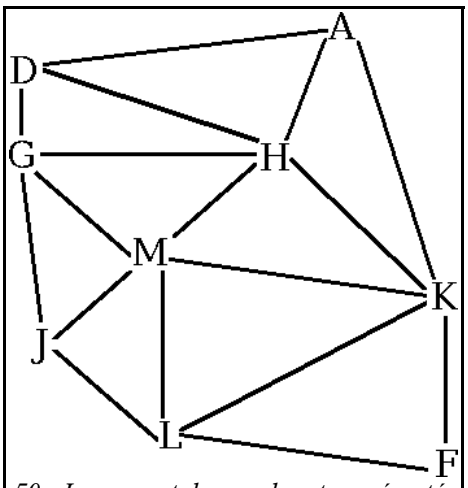
La propagation des symboles en neuronal : chaque circuit câblé est dédié à un symbole.

Dans la représentation neuronale il est impossible de faire la même chose, car chaque circuit est dédié à un symbole. Il est câblé au décodeur qui l'analyse. De plus le symbole possède une dimension topologique, car il est présent tout au long de ce fil : il se propage le long de cette circuiterie.

c) *Symbolique* vs *neuronal* : représentation des lieux, et traitement des déplacements

Représentation symbolique d'un réseau de lieux.

Dans un premier temps, nous présentons une représentation de la connaissance sous forme d'un réseau de segments (voir l'exemple ci-contre). Puisqu'en partant d'un lieu, nous pouvons accéder à un ou plusieurs lieux d'arrivée, elle peut donc se coder dans une liste d'associations L qui soit de cette forme : L = (segment1 segment2 ... segmentN) ; avec chaque segment qui se décompose ainsi : (LieuDeDépart LieuDArrivée).



50 - Le support du monde est représenté par un réseau de lieux.

Finalement, nous obtenons L=
((A D) (A H) (A K)
(D A) (D G) (D H)
(F K) (F L)
(G D) (G H) (G J) (G M)
(H A) (H D) (H G) (H K) (H M)
(J G) (J L) (J M)
(K A) (K F) (K H) (K L) (K M)
(L F) (L J) (L K) (L M)
(M G) (M H) (M J) (M K) (M L))

Représentation neuronale d'un réseau de lieux.

Maintenant nous présentons une représentation sub-symbolique, plus proche des réseaux de neurones. Soit N un nœud du réseau, et pN un pointeur sur ce nœud N. En utilisant ces conventions, nous pouvons représenter le réseau sous cette forme :

$\mathcal{R}$= {A, D, F, G, H, J, K, L, M}	Le réseau $\mathcal{R}$ est constitué d'un ensemble de nœuds
A= {pD, pH, pK}	Chacun de ces nœuds est constitué d'un ensemble
D= {pA, pG, pH}	de pointeurs sur un autre nœud.

25 Intentionnel au sens de l'informatique, notion que nous détaillerons quand nous traiterons les agents. Un agent est intentionnel s'il gère (positionne ou efface) des états internes. Ces dernier peuvent être codés chimiquement (peur, faim ...) ou par des états mentaux (amour de quelqu'un, peur d'une situation ...).

D= {pA, pG, pH}
F= {pK, pL}
G= {pD, pH, pJ, pM}
H={pA, pD, pG, pK, pM}
J= {pG, pL, pM}
K= {pA, pF, pH, pL, pM}
L= {pF, pJ, pK, pM}
M={pG, pH, pJ, pK, pL}

Le sub-symbolique pour raisonner en termes de changement de lieu.

Avec ces conventions, nous montrerons comment simuler le déplacements d'un agent, sa quête de nourriture, puis même comment lui permettre de planifier ses déplacement, et enfin comment un système peut raisonner sur ses connaissances.

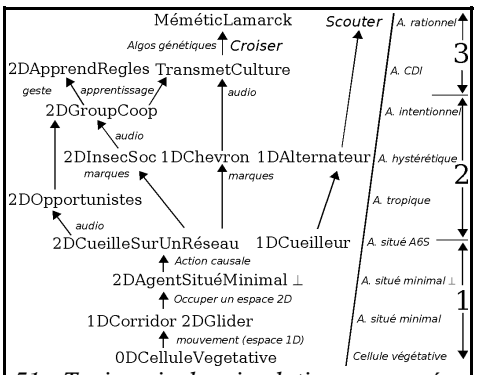
Quelques références bibliographiques étayant ces concepts et méthodes[RK 1986] [RK 1987].

De plus, il apparaît que nous pouvons aller plus loin dans cette direction : en utilisant cette méthode nous pouvons représenter une situation où une machine M connaît un fait seulement en termes de corrélation entre l'environnement et son état E_M. Cette représentation évite de coder le savoir de M au moyen de la représentation classique sous forme de liste de faits. Cependant, en utilisant cette représentation nous obtenons un système comparable au système logique modal S5 (KDT45) [Fer 1995] p 267.

Ch. 14 Méthodes utilisées afin d'atteindre nos buts

1) Présenter les méthodes retenues

Dans ce chapitre nous présentons les méthodes que nous retenons pour construire une plate-forme d'agents situés simulant des phénomènes cognitifs, de communications et quelques fois langagiers.



51 - Taxinomie des simulations proposées

2) Rappel rapide de notre problématique

2.A) Fournir une plate-forme

Notre but est de construire une plate-forme destinée à des chercheurs qui voudraient modéliser certains phénomènes cognitifs ou langagiers. Dans les chapitres précédents, nous avons mis en avance quelques concepts directeurs. Maintenant, à la lumière de ces résultats, nous regardons plus précisément comment nous procédons ?

2.B) Fournir une classification des simulations qui tournent sur la plate-forme

a) Fournir à l'utilisateur une classification de simulations

Nous fournissons une classification qui se représente à mi-chemin entre ces deux organisations :

- Une progression en couche concentriques.
- Une taxinomie : une représentation arborescente.

b) Les deux orientations de cette classification de simulations :

Étape 1 : Reconstruire l'évolution de l'automate fini (AF) jusqu'à l'agent situé.

Obtenir une organisation en couches concentriques. Initialement notre but était de guider une émergence de la structure des agents, depuis l'AF jusqu'à l'agent situé. En fait, nous sommes partis du séquenceur, i.e. encore plus bas.

Étape 2 : Simuler, modéliser des phénomènes cognitifs, de communication, langagiers.

Ensuite, munis de cet agent situé, nous simulons différents agents ou groupe d'agents. Nous commençons par modéliser des phénomènes cognitifs, puis de communication, pour finalement aborder des phénomènes langagiers simples.

3) Le concept de *zone privilégiée* où de *masse critique* invite à la simplicité

Dans l'analyse de notre travail, le concept de *zone privilégiée* et la notion de *masse critique* nous incite à concevoir des simulation très simples.

En effet, nous pensons qu'il existe une zone privilégié, coincée entre la barrière basse de l'insignifiance, et la barrière haute de la complexité. À l'intérieur de ce domaine, nous trouvons des simulations de la vie artificielle, qui sont à la fois suffisamment simples pour échapper au mur de la combinatoire, et assez complexes pour atteindre une masse critique présentant une épaisseur ontologique.

3.A) Définitions

La zone privilégiée.
C'est la zone qui englobe toutes les simulations non triviales, qui évitent les problèmes la combinatoire.

Le poids d'une simulation.
C'est la mesure de sa complexité informatique. Elle doit rester au-dessous d'un seuil, dit mur de la combinatoire, sinon sa complexité dépasse la puissance de traitement de l'ordinateur : dans ce cas, il semble ne plus avancer avant même d'avoir fournit les premiers résultats, il plante, il explose sous la combinatoire.

La masse critique d'une simulation.
Cette notion relève de considérations holistiques. En effet, une simulation atteint une masse critique quand elle est construite au moyen de tous les composants nécessaires à son fonctionnement, sinon, elle est en panne, elle ne fonctionne pas.

Ainsi nous obtenons donc la définition de la zone privilégiée.
C'est la zone où se trouvent plusieurs simulations optimalement simples, où chacune est suffisamment grande pour posséder une masse critique significative et suffisamment simple pour se tenir en-dessous du mur de la combinatoire.

3.B) Illustrer cette notion de zone privilégiée en termes de surface

Tout au long de la classification des simulations d'IA, nous souhaitons focaliser sur une zone privilégiée située entre deux extrêmes, i.e. après les réalisations triviales et naïves, et avant le mur de la combinatoire.

Avant cette zone privilégiée apparaissent les réalisations triviales et naïves.
Par exemple, au début de notre progression nous plaçons la cellule végétative, qui est non causale et solipsiste.

Cette zone privilégiée est située après les réalisations triviales et naïves.
Chaque simulation de cette zone privilégiée est suffisamment complexe pour être réaliste, et ne pas apparaître comme naïve ou triviale. Elle doit passer le seuil critique et présenter de l'intérêt.

Par exemple la simulation de la cueillette sur une grille est dépouillée : elle présente un agent qui cueille sa nourriture sur un support. Elle est simple, cependant elle est pleine de sens, spectaculaire et déjà causale.

Chaque simulation de cette zone est suffisamment simple pour éviter le mur de la combinatoire.
Dès que nous, concepteurs, compliquons un peu les structures modélisées, la combinatoire nous explose au visage. C'est pourquoi, dans le *design* de chacune de nos réalisations, nous devons être éclectiques, i.e. faire le ménage pour ne pas alourdir la simulation et la concevoir suffisamment simple pour qu'elle s'anime sans butter contre le mur de la combinatoire.

Par la suite elle grossit et s'enlise.
Par la suite, quand une recherche monte en complexité, elle grossit et elle s'enlise. Alors pour éviter cela, chaque chercheur l'éclate en de multiples sous réalisations, et focalise sur un thème donné. Ce faisant, cette recherche devient modulaire, dispersée et complexe. Ainsi, pour le profane qui en contemple seulement une partie, s'il ne possède pas le point de vue global de son histoire, alors elle perd son sens.

3.C) KISS[26] : plus ses simulations sont simples, denses, plus cette zone est large

Cette zone privilégiée, à l'existence si discrète, est toute petite. Cependant, elle est d'autant plus large que ses simulations, ces modèles sont nerveux, ramassés et dépouillés.

26 KISS : Keep It Simple and Stupid (voir page suivante).

Pour la faire apparaître, pour la développer il importe de ne pas avoir d'objectif industriel, mais plutôt d'imaginer des exemples éclectiques, d'écrire un code très dépouillé et d'utiliser un monde simple, i.e. un support et du mobilier digital et régulier.

> The models simulate the simultaneous operations and interactions of multiple agents, in an attempt to re-create and predict the appearance of complex phenomena. The process is one of emergence from the lower (micro) level of systems to a higher (macro) level. As such, a key notion is that simple behavioral rules generate complex behavior. This principle, known as K.I.S.S. ("Keep it simple and stupid", a backronym first introduced by Robert Axelrod [AHD 1981][Axe 1976][Axe 1984][Axe 1997] is extensively adopted in the modeling community.

Tableau 3: Les système multi-agents : http://en.wikipedia.org/wiki/Agent-based_model.

3.D) Conclusion : il existe une zone de fonctionnement privilégiée

Selon les considérations classiques des SMA, le programme monolithique classique *code+données*, éclate en un essaim d'agents élémentaires, qui par effet holistique (de groupe) s'avère capable d'un comportement conséquent.

Ainsi nous constatons que les SMA possèdent un positionnement original, dans une zone de fonctionnement privilégiée où ils peuvent simuler des phénomènes cognitifs et langagiers consistant. Nous montrons ceci au travers des exemples, que nous sommes capables d'exhiber.

4) Les méthodes que nous avons retenues, au vu de l'état de l'art

4.A) Préciser notre positionnement par rapport à la vie artificielle (VA)

Nous nous démarquons de la synthèse montante (bottom-up) de la VA.

La VA s'attache à reproduire des créatures vivante selon une synthèse montante. Ceci correspond à une démarche montante (bottom up). Elle part des briques initiales et les assemble. À cause de cette approche, elle se retrouve rapidement confrontée à une combinatoire lourde.

Un exemple de la difficulté de la VA.

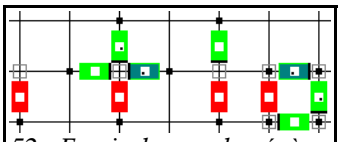
Regardons par exemple le travail de T. Taylor [Tay 1995], il s'occupe seulement d'émergence de la cellule ou de la vie : *From Artificial Evolution to Artificial Life*. Il butte inexorablement sur une combinatoire lourde, et en plus sa recherche échoue car il tombe sur des problèmes de duplication anarchique de cellules cancéreuses.

Éviter de se noyer dans la combinatoire de l'émergence et l'évolution.

Pour situer notre recherche par rapport à l'émergence et l'évolution, disons que ce n'est pas notre thème de recherche. Nous voulons éviter de nous noyer dans la lourde combinatoire de l' émergence et l'évolution, car c'est un thème difficile.

Notre expérience de la difficulté de la VA : Attention Danger.

Nous aussi avons travaillé sur la morphogénèse d'agents, nous avons essayé de générer une structure porteuse des capteurs et d'actionneurs qui seraient interconnectés par des liaisons électriques. Attention Danger ! Le problème est très raide, nous nous y enlisons.



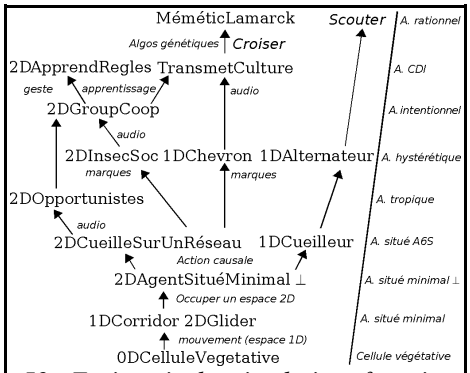
52 - Essais de morphogénèse

5) Les méthodes qui ont guidé la construction de notre taxinomie

5.A) Le choix d'un parcours en largeur d'abord

Parcours en largeur d'abord contre synthèse montante (bottom-up).

À proprement parler, notre thème de recherche n'est pas l'émergence et l'évolution. En effet, si l'IA induit à priori une approche descendante (top down), la VA induit plutôt une approche montante. Finalement, nous choisissons d'utiliser une démarche de parcours intermédiaire : la construction en largeur d'abord.



53 - Taxinomie des simulations fournies

D'un coté nous avons l'approche descendante de l' IA qui induit plutôt une démarche *top-down*. De l'autre, nous avons l'approche montante de la VA qui induit plutôt une démarche *bottom-up*. Mais, à proprement parler, notre thème de recherche n'est pas la VA, mais il consiste en une *émergence guidée*, une *évolution aidée*.

Alors ce positionnement de notre recherche fait que, finalement, nous choisissons d'utiliser une démarche de parcours intermédiaire : la construction en largeur d'abord. Ainsi nous générons une structure mixte, située à mi-chemin entre les deux extrêmes. La première partie, qui part de la cellule végétative, est organisée selon des couches en pelures d'oignons. Puis, à partir de l'agent situé, la seconde partie devient arborescente et se ramifie en 4 branches.

Nous construisons notre taxinomie selon une stratégie en largeur d'abord

Nous construisons notre ensemble de simulations comme les mathématiciens construisent un système : en mathématique chaque introduction d'axiome, permet de partir dans une direction et de déduire de nouveaux théorèmes. De même, nous fabriquons notre monde (mobilier+support) en introduisant progressivement les composants (capteurs, actionneurs et règles).

Cependant, nous évitons d'approfondir d'abord une branche, i.e. de construire toujours dans la même directions. Nous choisissons plutôt d'avancer dans toutes les directions, selon une stratégie en largeur d'abord : de façon éclectique, nous introduisons les notions les moins profondes.

5.B) D'un côté nous guidons l'émergence d'une progression (*en largeur d'abord*)

Notre méthode de classification consiste en une émergence assistée, guidée par l'utilisateur : afin d'éviter les problèmes de combinatoire, nous ne faisons pas de l'émergence à proprement parler. En fait nous balisons la trajectoire *AF→agent situé* comme pour aider un système émergeant à le parcourir. Nous nous attachons à introduire progressivement dans le modèle, les différents composant d'un système *support+mobilier* : le domaine, les grandeurs physiques, les capteurs, les actionneurs, les connexions neuronales...

5.C) De l'autre côté, la bionique guide notre classification de simulations

L'émergence de la vie naturelle guide notre travail de classification : dans une démarche bionique, nous analysons la nature, et ce que nous y observons nous aide à choisir l'ordre de cette séquence de construction.

Voyons maintenant cet aspect en détail.

a) Recenser les simulations susceptibles de figurer au sein de la classification

Quel sujet ? : initialement le thème de notre thèse est formulé ainsi :

Interprète pour modéliser le fonctionnement cognitif d'agents coopératifs et reproduire des phénomènes linguistiques

En fait, nous recherchons plutôt des simulations modélisant trois types de phénomènes : d'abord cognitifs, puis de communication, et enfin langagiers.

Ici, notre but est éclectique : reproduire *des* phénomènes.

D'un côté, comme le dit l'article indéfini *des* utilisé dans la formulation ci-dessus, notre démarche ne se veut pas exhaustive : l'idée qui oriente notre sélection est de chercher *des* candidats qui rentrent le mieux dans le cadre. En effet, puisque, cette thèse ne s'effectue pas dans le cadre d'un contrat industriel, nous ne partons pas à priori avec le désir de modéliser un phénomène particulier.

Il s'agit donc d'une méthode opportuniste.

En fait, notre démarche est plutôt opportuniste. Ceci veut dire qu'à l'instar des bioniciens, nous prospectons les cas, les occurrences de simulations connues. Nous cherchons un nouvel exemple de structure d'agent ou d'organisation de groupe : une modélisation qui présenterait un registre de fonctionnement non encore répertorié dans notre classification (un capteur ou un actionneur de plus, une nouvelle règle comportementale...).

C'est donc l'utilisateur qui construit ou trouve un nouvel exemple original à intégrer au sein de la classification. En quelque sorte, il guide, aide cette évolution, cette émergence de la structure des agents classifiés dans la taxinomie.

b) Épurer la simulation candidate pour en extraire la structure profonde

Quand nous disposons d'un bon candidat, nous travaillons à l'épurer, afin de mettre en évidence son architecture. La structure d'un agent ou groupe d'agents est holistique : il faut conserver les composants indispensables, et ôter tout ce qui est inutile

Pour simplifier la modélisation à l'extrême, nous enlevons de la structure les fonctionnalités qui nous paraissent inutiles. Bien sûr ici, le jeu consiste à ne pas jeter le bébé avec l'eau du bain, i.e. à ne pas ôter une fonctionnalité essentielle de la simulation.

Notre but est d'arriver à un synoptique épurée et enrichit : au sein de la structure candidate, nous cherchons à reconnaître une fonctionnalité nouvelle qui vient s'ajouter à la structure classique d'un agent parent, déjà membre de la taxinomie.

c) Recenser les traits caractéristiques de la structure de la modélisation

Quand nous avons trouvé puis épurée une simulation/modélisation candidate à la classification dans la taxinomie, nous recensons les traits caractéristiques de son profil (domaine, grandeurs physiques, capteurs, actionneur, connexions neuronales).

Pour effectuer ce difficile travail de classification, nous utilisons une méthode sans ambiguïté : nous classifions une simulation dans la taxinomie selon le critère objectif de sa <u>structure</u>.

Au départ, la structure des premières simulations de la taxinomie est basée sur les maths : nous utilisons les systèmes formels pour faire fonctionner un séquenceur, puis un AF. Ensuite, avec les agents réactifs, CDI, puis rationnels, nous passons à des architectures informatiques.

Finalement, nous disposons d'un bon candidat, muni d'une structure bien mise en évidence.

d) Utiliser ces traits caractéristiques pour classer la simulation dans notre taxinomie

Quand nous disposons d'un candidat dont nous connaissons la structure, ces informations nous servent à le classer au sein de la progression de simulations.

Dans cette taxinomie, nous classons des simulations. Les premières modélisent seulement des agents. Par la suite, nous classifions aussi les groupes d'agents, i.e. leur organisation sociale. Par exemple nous séparons les communications au moyen des critères *volontaire* ou *involontaire ;* ou encore selon le média *: par marqueurs* ou bien *audio*.

e) Analyse du résultat, i.e. de la qualité de la taxinomie

Notre taxinomie de simulations est aussi une progression selon un critère d'évolution des espèces : chaque simulation classifiée doit se différencier de ses voisines par la fonctionnalité nouvelle qu'elle apporte et présente.

Une fois la nouvelle simulation classifiée au sein de la taxinomie, nous vérifions le résultat final. Nous évaluons si ce nouveau candidat contribue à la qualité de la classification : une bonne progression ne doit pas présenter de *gap* i.e. l'espacement entre ses simulations doit être régulier.

6) Modéliser le procès cognitif de l'agent : structure d'arbre ou de réseau ?

6.A) Intro : la fonction de traitement cognitif de l'agent est difficile à modéliser

Au préalable, faisons un petit rappel : en matière de représentation, une organisation arborescente est très pratique car elle constitue une structure parcourable. Par contre, si une ou plusieurs de ses feuilles en viennent à se toucher, elle devient un graphe, qui s'avère plus compliqué à parcourir.

Selon une démarche fonctionnaliste, nous cherchons à décrire, modéliser la fonction de traitement de l'information effectuée au sein de l'agent. Vu les états internes et perceptions de ce dernier, son rôle consiste à décider les actions à effectuer. La décrire est compliqué car, quand l'être devient complexe, la modélisation de son fonctionnement prend une structure de graphe, i.e. de réseau.

À l'évidence, nous devons simplifier le problème. Notre but est de le séparer en sous-fonctions autonomes, mais est-il séparable ? Comme tout est imbriqué, quand nous enlevons un morceau à droite, nous remarquons qu'il servait à gauche.

En conclusion, nous essayons de reprendre le problème au départ, et dans le bon sens.

6.B) Quelle est l'intérêt d'une représentation arborescente des connaissances ?

Lemme : introduire quelques jalons.

Pour la clarté de l'exposé, prenons un peu de recul et posons quelques jalons sur la RC des connaissances en général. Ensuite nous retournerons à notre sujet : représenter la structure interne d'un agent effectuant un traitement cognitif.

Choisissons de simplifier en traitant le problème comme s'il présentait une structure d'arbre.

D'abord, afin de simplifier le problème, commençons par imaginer que la structure du programme cognitif de l'agent soit réduit à un arbre. Nous souhaitons le parcourir systématiquement, afin de le coder et de le transformer en programme ou jeu de règles. Mais alors, quelle stratégie de parcours allons-nous mettre en oeuvre ?

Étude 1 du parcours d'un arbre : selon une stratégie en profondeur d'abord.

Si nous choisissons un parcours en profondeur d'abord, nous optons pour une démarche analytique, qui expanse chaque fonction parcourue en sous-fonction. Mais si l'arbre à traiter est trop grand, dans cette démarche de déconstruction, nous rentrons trop dans les futilités, et comme le diable est dans les détails, nous finissons par nous y noyer.

Finalement, à trop garder le nez sur le guidon, nous ne savons plus où nous en sommes.

Étude 2 du parcours d'un arbre : selon une stratégie en largeur d'abord.

Si nous choisissons un parcours en largeur d'abord, nous optons pour une démarche synthétique, qui, en présence d'une fonction, évite de rentrer dans les détails, et parcourt seulement le premier niveau de ses sous-fonctions.

Ainsi, nous conservons un point de vue global, mais nous ne faisons que survoler superficiellement le problème.

6.C) Épistémologie de l'approche informatique : du synthétique à l'analytique

Introduction.

Maintenant, regardons un peu l'évolution de l'IA au fil des années. Pour étayer notre position, citons l'extrait suivant [CBJ 2010], dont nous paraphraserons ensuite le contenu :

Mais il nous semble qu'il sera difficile de conserver à l'IA un rôle moteur dans une informatique considérée comme science des modèles et systèmes calculables :
- l'informatique se technologise de plus en plus et n'a pas vraiment l'ambition de se faire science,
- les spécialistes de l'IA semblent manquer de projets comme en posaient les pères fondateurs de la discipline,
- la fréquentation d'autres disciplines demande une culture à la fois très vaste et très profonde, ce qui quelquefois décourage les jeunes chercheurs.

4 Conclusion de Panorama actuel de l'IA *(Alain Cardon, J-Paul Baquiast, Christophe Jacquemin)*

a) Les premières réalisations des pères fondateurs de l'IA étaient plutôt synthétiques

Au début de l'IA, les réalisations marquantes des pères fondateurs étaient simples, générales et plutôt synthétiques. Les chercheurs, analysaient, concevaient et réalisaient in extenso leur produit, sans assembler de modules préfabriqués.

b) Montée en puissance du traitement symbolique : s'enfoncer dans l'arborescence

Ensuite, en prenant soin de ne pas mettre de côté le connexionisme, nous constatons cependant principalement une montée en régime du symbolique. Avec des ordinateurs qui montent en puissance, les langages symboliques de plus en plus macroscopiques permettent d'implémenter plus de réalisations conséquentes.

En conclusion, l'informatique et l'IA partent dans toutes les directions.

c) Aujourd'hui l'IA ne croit plus à son baptême et est devenue *Informatique Avancée*

L'informatique et l'IA se technologisent de plus en plus ; elles n'ont pas vraiment l'ambition de se faire science et tournent à l'ingénierie.

Nous voyons les travaux actuels comme atomisés en une constellation de produits dédiés et spécialisés. Si nous les représentons au sein d'une taxinomie des réalisations de l'IA, ils se répartissent en position de feuilles, au bout des branches de la classification.

Le nez sur le guidon, cette recherche devient analytique, elle déconstruit les modèles, sans être capable de tirer des passerelles entre les disciplines. Les spécialistes de l'IA semblent manquer de projets synthétiques comme en réalisaient autrefois les pères fondateurs de la discipline.

6.D) Comment traiter de façon arborescente le graphe d'un procès cognitif ?

a) Introduction

Imaginons que la représentation du traitement de la cognition chez un agent soit complexe au point d'être représentée par un graphe. Ainsi nous sommes confrontés à la difficulté de l'analyser et le décrire. Alors regardons dans la littérature afin de connaître les méthodes généralement utilisées en informatique.

Finalement, à l'issue de notre recherche, nous retiendrons une méthode pour parcourir de façon arborescente le graphe de cette structure. Évidemment, cette démarche paradoxale ne peut s'employer qu'au prix d'un compromis, i.e. en acceptant d'effectuer une représentation incomplète : nous délaissons les arbres éloignés, pour seulement traiter les arc locaux comme les premières branches d'un arbre. Et ainsi nous récupérons ainsi les facilités d'affichage des arbres.

Voyons le résultat obtenu en fonction de la stratégie de parcours utilisée, i.e. selon que nous privilégions largeur ou profondeur.

b) La démarche analytique de déconstruction ne fonctionne pas

Quelque soit la stratégie de parcours utilisée, nous concluons que la démarche analytique de déconstruction ne fonctionne pas

Introduction : d'abord nous essayons la démarche analytique.

Nous commençons par essayer d'abord la démarche analytique : c'est une démarche de déconstruction, elle consiste en un parcours top-down, i.e. en profondeur d'abord.

Si nous ne coupons pas à partir d'une certaine profondeur alors ça ne marche pas.

Nous retombons sur le parcours classique en profondeur d'abord mais là nous parcourons tout le réseau : c'est une démarche trop lourde que nous ne pouvons pas envisager.

Si nous coupons à partir d'une certaine profondeur *n* alors ça ne marche guère mieux.

C'est la décomposition du tout vers ses parties, mais nous arrêtons à une distance *n*, nous pouvons atteindre le sous-graphe de rayon *n*, i.e. constitué de tous les points situés à une distance du point d'entrée qui est inférieure à *n*. Mais, quand *n* est grand, nous risquons de repasser sur nos traces, car nous parcourons un graphe et pas un arbre.

Conclusion :

Quelque soit le cas, la démarche analytique de déconstruction ne fonctionne pas

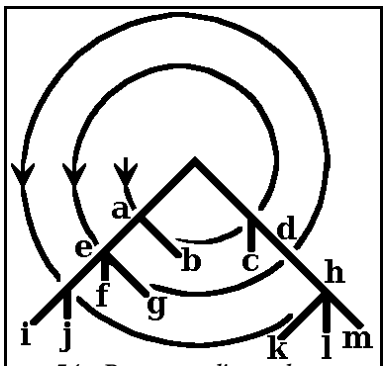
c) La solution la stratégie de parcours en largeur d'abord

Parcours en largeur d'abord : si nous ne coupons pas, alors ça ne marche pas.

Si nous parcourons le réseau sans couper dans les branches, alors nous retombons sur le parcours classique en largeur d'abord, mais là encore nous parcourons tout le réseau : c'est une démarche trop lourde que nous ne pouvons pas envisager.

Si nous coupons au-delà d'une profondeur *n* : approche synthétique des philosophes.

Si nous parcourons le réseau en largeur d'abord, en coupant au-delà d'une profondeur limite, nous obtenons une approche globale synthétique qui ressemble à celle des philosophes



54 - Parcours d'un arbre, en largeur d'abord

Dans ce cas, les branches sont tractables parce que nous coupons à une profondeur limite, alors nous résumons, synthétisons le domaine. Nous ne faisons que de le survoler, mais nous réussissons à parcourir le sous-graphe de rayon *n* jusqu'au bout. Ainsi la démarche obtenue correspond à plusieurs résultats, citons par exemple :

Résultat 1 : d'abord signalons le regard synthétique, l'approche épistémologique des philosophes qui peuvent ainsi appréhender globalement un domaine complexe en le survolant.

Résultat 2 : ensuite nous reconnaissons surtout la façon dont Herbrand construit son univers éponyme. Par exemple avec les constantes A et B, les fonctions f et g, il assemble son univers U = {A, B, f(A), f(B), g(A), g(B),f(f(A)), f(f(B)), f(g(A)), f(g(B)) ...}. Il s'agit là, typiquement, d'un exemple de parcours en largeur d'abord.

Résultat 3 : enfin nous retrouvons une démarche équivalente, mais vue sur un autre angle. Il s'agit de la synthèse d'un arbre, par emboîtement de sous-partie, selon une approche descendante.

Si nous partons assez près de la base pour essayer de reconstruire : approche locale synthétique des pionniers.

Si nous nous plaçons dans une démarche de construction d'un modèle et que nous parcourons le réseau en largeur d'abord, mais en commençant assez près de la base, i.e. de la racine. Concrètement, selon la complexité du modèle, cela consiste :

- Soit à construire au moyen de briques macroscopiques, un modèle qui ratisse large.
- Soit à construire au moyen de briques élémentaires, un modèle assez basique mais qui est autonome.

Dans les deux cas, la démarche de parcours est tractable, les branches sont *soulevables* :

- Soit car nous utilisons des modules macroscopiques.
- Soit parce que nous sommes assez proches des feuilles et que nous y arrivons rapidement : les branches qui y mènent, sont très courtes).

La démarche que nous décrivons ici se concrétise dans l'histoire de l'informatique ou de l'IA. En effet, les premières réalisations y sont intéressantes, car elles sont effectuées sur un sous-

graphe peu profond. À ce prix les pionniers synthétisent une réalisation autonome et entière : elle est *localement globale*

D'abord, la réalisation est locale car elle est effectuée localement sur un sous-graphe : chaque chercheur se focalise sur un sous-problème qui est dédié et local.

Ensuite, la réalisation est globale car le fait que le sous-graphe soit peu profond entraîne qu'il soit parcourable : le problème traité est entièrement soulevable, appréhendable.

Conclusion : la solution du parcours en largeur d'abord, sur un faible rayon

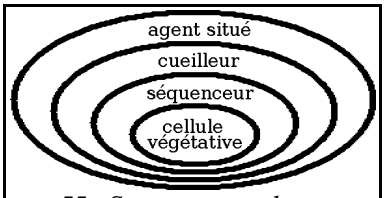
Donc nous avons trouvé cette solution de la stratégie en largeur d'abord : elle parcourt toutes les branches parce que chacune est tractable, car elle est soit modulaire, soit basique.

En conclusion, nous retenons cette solution pour la prolonger en direction de l'évolution.

d) Une construction en couches concentriques, inspirée par l'évolution

Description de la méthode :

- Départ depuis une fourche qui constitue la base, la racine initiale : la cellule végétative.
- Au début, chaque embranchement ou fourche se réduit à un segment de droite.
- Stratégie en largeur d'abord : nous parcourons tous les segments de la fourche.



55 - Structure en pelures d'oignon

Analyse du résultat :

Toute les branches sont parcourables car elles sont des segments de droite, donc cette méthode évite de couper car la coupure y est naturelle, elle provient de la donnée du problème : l'être simulé est autonome par nature.

À la limite, quand l'arbre à parcourir se réduit à une branche principale, il ne présente qu'une branche à chaque nœud, dans ce cas, nous obtenons une structure en couches concentriques, i.e. selon une évolution en pelure d'oignons.

Nous retombons sur le concept d'évolution, puis sur celui de bionique.

Pour construire la structure d'un être effectuant un traitement cognitif, nous partons depuis le point de départ constitué par la cellule végétative, et nous la construisons en couches concentriques, jusqu'à la structure d'agent.

Notre démarche est directement inspirée par la bionique : dans la mesure du possible, nous suivons et copions les inventions de la nature.

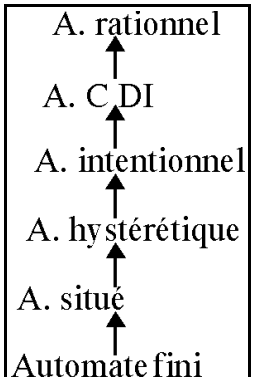
Ch. 15 Simplifier l'automate fini (AF) pour obtenir un séquenceur

Dans notre état de l'art, nous avons montré comment, sur la scène informatique, sont apparus successivement : les langages de haut niveau, les langages structurés, les objets, les agents, puis les agents situés (AS).

1) Pour formaliser l'agent situé, baser notre construction sur l'AF

À ce stade, notre but est de formaliser le domaine des AS. Pour ce faire, nous basons notre travail sur l'automate à états finis (AF). Nous l'utilisons comme brique de base de notre construction : il constitue une pierre angulaire intéressante, car il est à la fois utilisé en informatique théorique, et bien formalisé sur le plan mathématique.

En partant de cette base, nous construisons pas à pas toute une progression jusqu'à l'agent situé. Par la suite nous continuons cette construction, jusqu'à l'agent intentionnel, l'agent CDI et l'agent rationnel.



56 - Construction depuis l'AF

2) Rappels à propos de l'AF (automate à états finis)

Ici notre propos n'est pas de faire un cours sur l'AF, donc nous effectuons rapidement quelques rappels.

2.A) Description mathématique de l'AF

Un automate fini déterministe A est un quintuplet $\{Q, \Sigma, \delta, q_0, F\}$:

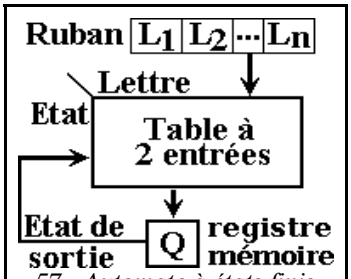
1) Q est un ensemble fini d'états.
2) Σ est un alphabet (le ruban) d'entrée.
3) $\delta : (Q \times \Sigma) \rightarrow Q$, la fonction de transition donnée par la table à deux entrées Q et Σ .
4) q_0 est l'état de départ du système.
5) F, inclus dans l'ensemble Q, est l'ensemble des états finals.

2.B) Implémentation de l'AF

Ci contre le schéma de l'implémentation d'un AF :

Un registre tampon mémorise la sortie de la table à deux entrées, et la fournit sur sa sortie. Cette information est ramenée sur l'entrée *état* de cette table.

À chaque impulsion de chargement qu'il reçoit, ce registre mémoire enregistre la sortie de la table et la fournit sur sa propre sortie. Cette donnée détermine *Q* l'état courant du système, qui remonte immédiatement pour adresser la table.



57 - Automate à états finis

Au même moment, cette impulsion est fournie au ruban de symboles en entrée, ce qui le fait avancer d'un cran : ainsi nous préparons la prochaine transition d'état du système.

2.C) Ainsi la lecture du ruban ouvre l'automate sur l'extérieur

Selon un point de vue épistémologique, le ruban placé en entrée sert de capteur de l'environnement. Il ouvre l'automate sur l'extérieur. Alors ce dernier n'est plus solipsiste, il devient sensible au monde : maintenant il perçoit l'extérieur et il en tient compte.

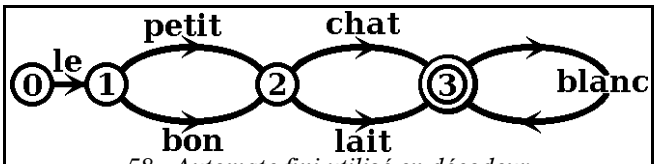
Ainsi, confronté à un environnement diversifié, il peut se trouver placé devant une alternative, pour laquelle il doit prendre une décision : il doit choisir sa destination entre deux états possibles.

La figure ci-dessous illustre ceci au moyen d'un graphe de transition.

2.D) La structure du graphe de transition

Voici un exemple qui illustre le cas d'un AF ouvert sur l'extérieur.

Dans le graphe de transition ci-contre, en deux endroits, aux états 1 et 2, le graphe de cet automate présente une structure en patte d'oie, en delta : selon la lettre lue sur le ruban, il choisit soit un chemin, soit l'autre.



58 - Automate fini utilisé en décodeur

3) Présentation de notre feuille de route pour les chapitres à venir

Voici la feuille de route que nous allons suite au fil des chapitres à venir. Dans le tableau ci-dessous, elle est présentée en totalité, mais en fait, elle est seulement donnée pour information, car nous venons seulement de traiter la colonne 1.

type catégorie	***1) Sys.For. type 3***	***2) Sys.For. type 4***	***3) Machine séquenceuse***	***4) Cellule végétative***	***5) agent situé mini***	***6) Pluralité***
Nom commun	Automate fini	Séquenceur	Machine séquenceuse	Cellule végétative	agent situé minimal	Agent situé (AS)
Exemples :			jaquemart trotteuse	clignotant à deux états	cueilleur:thon requin	2DcueilleSurUnRéseau
Nature du cadre	mathématiques abstraites	mathématiques simplifié	physique, concret, moteur énergie ; solipsiste	vie artificielle solipsiste monde 0D : 1 seul lieu	agent sur un support constant	as en// simplifie support as non déterministe
quelle transition pour arriver ici?	départ depuis ici : l'automate fini	simplification de l'automate fini	*abstrait→concret, réel* se matérialise, s'incarne	introduit la vie, solipsiste autonome interne	situé sur un support, déplacement	mettre des agents actifs ou passifs en parallèle
Nature du procès effectué	règle de réécriture circuit de logique séquentielle	règle de réécriture circuit de logique séquentielle	moniste : mécanisme dualiste : prog. informatique ou système formel	solipsiste = f(interne) perçoit-décide-agit états, horloge internes	décision_cnst action = f (constante) états horloge extern	percept-décision-action = f(environnement) états & horloge externes

Nous travaillons à formaliser une partie du domaine des agents situés :

Colonne 1 : Nous sommes partis de la base mathématico-informatique de l'automate fini

À ce stade de notre présentation, nous venons de traiter la colonne 1. Nous l'avons surtout abordé dans le cadre abstrait des mathématiques. Nous avons montré que le traitement qu'effectue l'automate fini (AF) peut être décrit au moyen des règles d'un système formel, ou par la table d'une mémoire au sein d'un circuit de logique

Colonne 2 : Dans le paragraphe suivant, nous simplifions l'AF pour obtenir un séquenceur

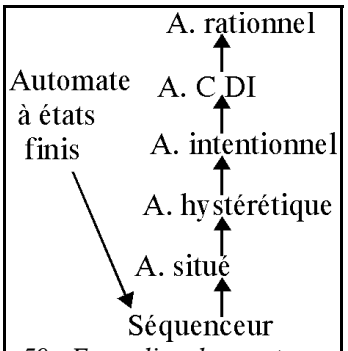
Pour ce faire, nous resterons dans le même cadre des mathématiques et implémenterons le procès de la même façon.

4) La restriction de l'AF, à un séquenceur

Dans la partie pratique de ce travail, nous exhibons une progression de simulations, qui part de la cellule végétative (CV) et va jusqu'à présenter des modèles simplifiés de traitements cognitifs.

Mais, de suite, au début de cette progression, cette cellule induit un problème, son comportement est si simple, si épuré, que le modèle de l'AF s'avère trop fort pour la décrire.

Aussi, pour construire notre progression, nous ne partons pas de l'automate fini, mais nous prenons même une base de départ plus simple, le séquenceur, car il utilise une table à seulement une entrée, alors que celle de l'AF en requiert deux.



59 - Formaliser la construction depuis le séquenceur

En prenant le séquenceur comme base initiale, nous construisons pas à pas toute une progression jusqu'à l'agent situé. Par la suite nous continuons, jusqu'à l'agent intentionnel, l'agent CDI, jusqu'à l'agent rationnel.

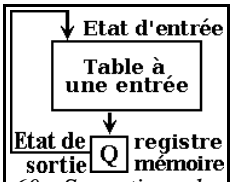
Ceci nous amène à présenter rapidement le séquenceur, comme restriction de l'AF. Pour cela nous prenons un AF et lui ôtons son ruban externe.

4.A) Description mathématique du séquenceur

Un séquenceur A est un quadruplet {Q, δ, q_0, F} :
1) Q est un ensemble fini d'états.
2) δ : Q→ Q, la fonction de transition donnée par la table à une entrée Q.
3) q_0 est l'état de départ du système.
4) F, inclus dans l'ensemble Q, est l'ensemble des états finals.

4.B) Ci-contre nous présentons le schéma de l'implémentation d'un séquenceur

Un registre tampon enregistre la sortie de la table à une entrée, puis il ramène cette information sur son entrée. À chaque impulsion de chargement qu'il reçoit, ce registre mémorise la sortie de la table et la fournit sur sa propre sortie. Cette donnée détermine *Q* l'état courant du système. Immédiatement, cet état remonte pour adresser l'entrée de la table.



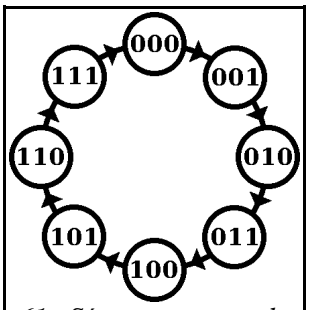
60 - Synoptique du séquenceur

4.C) Le séquenceur est fermé au monde : il est solipsiste

Selon un point de vue épistémologique, la disparition du ruban en entrée ferme l'automate au monde extérieur. Alors il devient solipsiste, il n'est plus ouvert au monde. Ce fait se traduit de trois façons :

Conséquence 1 : le séquenceur ne possède plus qu'une table à une seule entrée
La table de l'AF qui possédait deux entrées, perd celle correspondant au ruban, qui l'ouvrait au monde extérieur. Ainsi, elle ne possède plus que celle correspondant à son état interne : le séquenceur est seulement tourné vers lui même, il est solipsiste.

Conséquence 2 : pas d'alternative dans son graphe de transition
Ceci se traduit dans l'architecture de son graphe de transition : il ne peut plus se trouver face à une alternative, i.e. une structure en patte d'oie, en delta de fleuve, comme dans le graphe de l'automate monté en décodeur de la page précédente. Illustrons ceci au moyen d'un graphe de transition :



61 - Séquenceur en cycle

Conséquence 3 : La structure du graphe de transition des séquenceurs
La route d'un séquenceur est toute tracée, il file droit devant lui. Parmi ses structures de graphes significatives, on distingue particulièrement ces deux là : dans les rares mondes infinis où nous travaillons parfois, le graphe file droit vers l'infini ; et dans les petits mondes finis où nous travaillons préférentiellement, un cycle tourne indéfiniment (illustration ci-contre).

5) Transition : vers la matérialisation

Dans le chapitre suivant, présenté en colonne 3 de la feuille de route, nous décrivons la matérialisation, et constatons que cette action de plonger le séquenceur dans le monde réel, l'incarne en processeur (celui qui effectue un procès, un changement d'état du monde).

Ch. 16 Vie artificielle, la cellule végétative exécute une séquence digitale

(Introduire l'inconscient somatique et une vie solipsiste dans une boîte noire)

1) Nous introduisons une vie artificielle (VA) dans notre modèle(colonne 4)

Une cellule végétative qui, en vivant, exécute une séquence digitale

Dans notre progression de simulations, nous procédons à un lourd changement de registre. À l'étape précédente, nous disposions d'une machine séquenceuse qui était programmée pour exécuter une séquence digitale. Maintenant, notre simulation est construite directement autour d'une cellule végétative, qui vit de façon autonome. En vivant, elle exécute une séquence digitale.

La cellule végétative présente la forme d'une boîte noire autonome et solipsiste

Au sein de notre simulation, nous modélisons la cellule végétative (CV), sous la forme d'une boîte noire. Elle est solipsiste et autonome :

- Elle est solipsiste car elle est fermée au monde extérieur.
- Elle est autonome car elle mène sa propre vie : elle se base sur des états locaux, internes afin de gouverner ses propres actions.

En son sein, la CV est programmée pour exécuter une séquence digitale.

La simulation est de type : OD, c'est à dire sans dimension : elle se produit dans un lieu unique, et n'implique pas de changement de lieu, si bien qu'on peut éviter de mentionner le lieu.

Dans cette approche 0D, à un seul lieu, à une case, nous obtenons une simulation sans mouvement. Selon cette approche très matérialiste, un mécanisme déroule un programme : la cellule grandit et des capteurs, palpeurs détectent des étapes et numérisent le processus. En sortie de cette simulation, le phénomène digital consiste en une grandeur physique digitalisée, par exemple en une taille qui varie en escalier, i.e. par bonds.

Selon la vie artificielle : $CV_{(+MS)} \rightarrow CV_{(+MS)}$

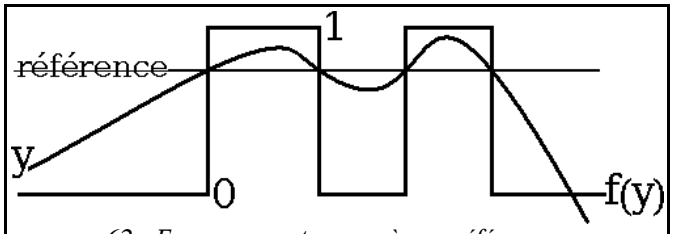
Puisqu'en son sein la CV est programmée pour exécuter une séquence digitale, nous disposons d'une cellule de la forme $CV_{(+MS)}$, i.e. qui porte en son sein un séquenceur.

Dans cette modélisation, nous simulons la vie artificielle, alors la cellule est capable de se reproduire identique à elle-même. Dans ces conditions, la cellule-fille naît avec la même structure $CV_{(+MS)}$ que celle de sa mère. Ainsi le cycle se reproduit continûment...

type catégorie	***1) Sys.For. type 3***	***2) Sys.For. type 4***	***3) Machine séquenceuse***	***4) Cellule végétative***	***5) agent situé mini***	***6) Pluralité***
Nom commun	Automate fini	Séquenceur	Machine séquenceuse	Cellule végétative	agent situé minimal	Agent situé (AS)
Exemples :			jaquemart trotteuse	clignotant à deux états	cueilleur:thon requin	2DcueilleSurUnRéseau
Nature du cadre	mathématiques abstraites	mathématiques simplifié	physique, concret, moteur énergie ; solipsiste	vie artificielle solipsiste monde 0D : 1 seul lieu	agent sur un support constant	as en// simplifie support as non déterministe
quelle transition pour arriver ici?	départ depuis ici : l'automate fini	simplification de l'automate fini	*abstrait → concret, réel* se matérialise, s'incarne	introduit la vie, solipsiste autonome interne	situé sur un support, déplacement	mettre des agents actifs ou passifs en parallèle
Nature du procès effectué	règle de réécriture circuit de logique séquentielle	règle de réécriture circuit de logique séquentielle	moniste : mécanisme dualiste : prog. informatique ou système formel	solipsiste = f(interne) perçoit-décide-agit états, horloge internes	décision_cnst action = f (constante) états horloge extern	percept-décision-action = f(environnement) états & horloge externes

Traitement de l'information.

Durant chaque cycle de vie de la cellule, ces états et paramètres internes sont traités, hors contexte, au sein de cet être solipsiste[27]. Ceci induit chez lui une prise de décision autonome.



62 - En comparant y avec à une référence, nous obtenons f(y) : une courbe digitale

Dans la CV ces grandeurs analogiques sont comparées à des seuils de référence, et les résultats de ces comparaisons fournissent des états internes digitaux. Par exemple, le sentiment de faim apparaît quand le niveau d'énergie de la cellule arrive dans la zone d'hypoglycémie. De même, il disparaît quant ce niveau dépasse le seuil de satiété.

Externalisation.

Dans la cellule, le déclenchement des actions est rythmé par ces différentes synchronisations, qui activent ses actionneurs. Finalement, la cellule digitalise son phénomène physique interne, ainsi elle peut présenter à l'observateur extérieur une séquence numérique. Par exemple, la taille de la cellule peut prendre les états 00, 01, 10 et 11, où l'état 00 correspond à une graine et 11 à une cellule qui va se reproduire.

En conclusion, quelle est la relation de la cellule avec l'extérieur ?

Ainsi les actions de l'être sont internes et causales ; par contre lui-même est solipsiste : il est seulement tourné sur sa vie intérieure, et donc demeure fermé au monde externe.

Ainsi l'observateur extérieur peut penser que le fonctionnement macroscopique du système est régi par les règles d'un système formel, mais il ignore comment il est implémenté. Ainsi l' implémentation de ce traitement peut être *moniste matérialiste* et utiliser un mécanisme totalement matériel. De même, il peut être dualiste *logiciel-matériel*, lorsque l'exécution d'un système informatique pilote les actions des effecteurs.

1.A) Analysons l'architecture de la $CV_{(+MS)}$ qui exécute une séquence digitale

Rappels.

Nous avons effectué la transition *colonne_2→colonne_3*, qui va de séquenceur à machine séquenceuse, en traduisant la forme abstraite du séquenceur vers la forme concrète de la MS, qui est une machine physique.

À ce stade *(colonne_3)*, nous disposons d'un être matériel, il effectue des procès sur le monde, en consommant de l'énergie, mais il est solipsiste, il n'agit pas causalement en fonction de paramètres extérieurs.

Ajout d'états somatiques internes à la machine séquenceuse (MS).

L'introduction de la cellule $CV_{(+MS)}$, entraîne un changement important : En colonne 4, le moment est venu d'implémenter au sein de l'être des états somatiques internes qui produisent des pulsions.

Au sein de la cellule végétative, nous trouvons des états internes somatiques correspondant aux grandeurs physiologiques de l'être. Dans la CV ces grandeurs analogiques sont comparées à des seuils de référence, et ils fournissent des états internes digitaux.

Au début nous travaillons particulièrement sur le *niveau d'énergie.*

27 Solipsisme : du latin *solus* : seul et *ipse* soi-même. Théorie philosophique de l'être seul dans sa tour d'ivoire, face à lui-même: l'esprit est la seule chose qui existe réellement. Solipsiste (adjectif) : se dit de celui qui est fermé au monde, coupé de celui-ci.

Nous disposons seulement de la glycémie, i.e. du niveau interne de glucose dans le sang. La cellule assimile une énergie diffuse en provenance du milieu où elle est en immersion, elle peut ainsi vivre et se reproduire.

Ainsi, le sentiment de faim apparaît quand le niveau d'énergie de la cellule arrive dans la zone d'hypoglycémie. De même, il disparaît quant le niveau d'énergie dépasse le seuil de satiété.

Par la suite viendront d'autres grandeurs (*faim, soif, chaud, froid...*).

Ajout d'actionneurs immobiles à la machine séquenceuse (MS)

Dans la boîte noire, nous introduisons des actionneurs immobiles. Grâce à eux, la cellule peut assimiler l'énergie diffuse présente autour d'elle, mourir d'inanition si elle ne trouve pas de nourriture et se reproduire quand elle est bien portante.

Ensuite, dans les chapitres suivants, nous passons à un fonctionnement en relation avec l'extérieur (colonne 5).

1.B) Traitement des états somatiques, au sein de la $CV_{(+MS)}$

Maintenant chaque cellule dispose d'états internes somatiques et les gère de façon causale et solipsiste : elle effectue un traitement causal i.e. en appliquant des règles formelles, mais il est solipsiste, i.e. il est effectué sans que la cellule perçoive le monde externe : le système de règles utilisé code des fonctionnements de cellule végétative.

La $CV_{(+MS)}$, cette cellule qui porte une machine séquenceuse, fonctionne de façon autonome

La $CV_{(+MS)}$, correspond à une boîte noire où se déroule toute une vie intérieure : ses états somatiques internes évoluent de façon autonome, et ainsi génèrent son cycle de vie. Quand dans les simulations suivantes de la progression, chaque cellule possède des états internes plus nombreux, ils lui fournissent une base de temps variée et personnelle.

Dans le traitement de l'information de la cellule les interruptions de survie sont prioritaire

Puisqu'ils les fonctionnements somatiques de la $CV_{(+MS)}$ sont autonomes et primordiaux, pour des raisons vitales, alors ils sont prioritaires. Ainsi ils interviennent avant toutes les autres règles du système (mécanique ou informatique) qui gère le comportement de la simulation globale, i.e. du séquenceur.

Quand, dans cette cellule, nous modélisons ces interventions prioritaires, qui peuvent se manifester intempestivement, en fait nous simulons l'inconscient somatique de chaque être et ses pulsions internes. Elles interviennent, au niveau des couches les plus profondes, sur le comportement global de son système.

Le traitement de l'information, est synchronisé, rythmé par les pulsations des états internes

Durant chaque cycle de vie de la cellule, ces états et paramètres internes sont traités, hors contexte, au sein de cet être solipsiste[28]. Ceci induit chez lui une prise de décision autonome.

Ainsi, dans la cellule, le déclenchement des actions est synchronisé par ces différents rythmes, qui activent ses actionneurs.

Dans la CV le traitement de l'info est digital, les pulsations des états internes sont redressées

Au début de la phylogenèse le traitement des données à l'intérieur de la cellule était sans doute analogique. Mais afin qu'il soit résolu et énergique, il doit être abrupte, i.e. logique, numérique.

Alors, afin de traiter une information numérique, la cellule, qui reçoit en entrée des phénomènes physiques analogiques, les digitalise au moyen de comparateurs.

Finalement, à cause de cette digitalisation, l'observateur extérieur constate que la cellule déroule une séquence numérique.

En conclusion, quelle est la relation de la cellule avec l'extérieur ?

28 Solipsisme : du latin *solus* : seul et *ipse* soi-même. Théorie philosophique de l'être seul face à lui-même dans sa tour d'ivoire : l'esprit est la seule chose qui existe réellement. Solipsiste : fermé au monde

Rappel : l'être est solipsiste. Les actions de l'être sont causales ; par contre lui-même est solipsiste : il est seulement tourné sur sa vie intérieure, et donc demeure fermé au monde externe.

Aspect implémentation informatique

Ainsi l'observateur externe, qui regarde la cellule évoluer, pense que son fonctionnement macroscopique est régi par les règles d'un système formel, mais il ignore comment il est implémenté. Dans les premières simulations de la classification, le traitement effectué par la cellule est *moniste matérialiste* et utilise un mécanisme totalement matériel. Dans les simulations suivantes, plus sophistiquées, l'implémentation du traitement est dualiste *logiciel-matériel.* Alors l'exécution d'un système informatique (programme ou système formel) pilote les actions des effecteurs de la cellule.

Au niveau de l'implémentation informatique apparaît une autre contrainte. Même si les états internes somatiques de la CV relèvent du monde physique, pour implémenter cette simulation sur un ordinateur nous fonctionnons dans un monde digital, synchronisé par une base de temps numérique, alors ces interventions somatiques (manger mourir enfanter) sont prises en compte seulement une fois dans chaque intervalle de temps élémentaire. Comme elles sont prioritaires, elle le sont en son début.

1.C) Quelles sont les conséquences de cette introduction ?

En introduisant la glycémie dans la modélisation de l'être, nous l'amenons à gérer cet état somatique interne, et, comme il est en situation de survie dans le monde, s'il est bien conçu, ceci induit chez lui deux comportements : la cueillette et la reproduction.

a) Une glycémie faible stresse l'être et déclenche un comportement de cueillette

Maintenant la modélisation de l'être est plus fine. Nous avons introduit chez lui une variable analogique : le niveau de glycémie. L'être la gère de façon réflexe inconsciente, et quand son amplitude devient faible, ce manque entraîne chez lui un stress qui déclenche un comportement de cueillette.

b) Introduction d'un comportement causal et local

Au sein de la boîte noire, ce niveau d'énergie est testé et interprété causalement. En conséquence nous concluons que : d'abord nous venons d'introduire dans la cellule un comportement causal et local. Ensuite :

c) Le niveau de glycémie permet d'évaluer l'adaptation de l'être

Nous nous sommes inspirés de la VA et maintenant disposons d'un indicateur du niveau de glycémie. Cet emprunt, nous fournit une mesure pour évaluer l'adaptation du comportement de l'être à son milieu. Certes cette évaluation est indirecte mais elle est infiniment précieuse car elle s'effectue facilement. Pour la mettre en oeuvre il est inutile d'utiliser des outils de traitement formel de l'information, il suffit de sommer une grandeur analogique. Muni de cet outil, nous l'utilisons dans deux directions : la sanction et l'encouragement.

Une évaluation dans le sens d'une sanction.

Après un lissage statistique, un état d'hypoglycémie peut indiquer une mauvaise adaptation de l'être à son milieu, ou un milieu peu susceptible d'héberger des êtres. En tous cas la sanction vient immédiatement : l'être perd la vie et disparaît.

Une évaluation dans le sens d'une récompense.

Si l'être est bien adapté à son milieu, ou bien si le milieu est riche, alors il y prospère. Ainsi, à peu de frais, le programmeur dispose d'une évaluation de la relation du couple *être-milieu*, et quand la mesure est positive ceci permet aux êtres de se reproduire. Ensuite, dans les chapitres suivants, nous verrons que, par ce biais, une sélection naturelle devient possible. Ainsi nous quitterons la démarche de VA pour aller vers une évolution artificielle.

d) Une conception, un *design* intelligent

Lemme : notion d'action

Dès qu'on dispose d'un monde, même très simple i.e. un objet sur un support de cases, le fait de modifier la relation *objet-support*, induit un changement, une transition d'état du monde, que nous appelons procès.

En toute rigueur, selon un point de vue épistémologique, ce procès ne devient une action que s'il a été effectué dans une démarche téléologique, finaliste i.e. orientée vers un but.

Ces êtres proviennent d'une conception, d'un design intelligent

Dans le cadre de l'incarnation du séquenceur dans le concret (colonne 3 du tableau), nous obtenons la MS qui change l'ordre du monde en effectuant des procès. Comme son nom *machine séquenceuse* l'indique, nous sommes en présence d'une machine qui déroule mécaniquement une séquence pré-programmée : chez elle nous ne voyons pas de démarche finaliste.

Dans ce chapitre (colonne 4 du tableau), nous introduisons les états internes au sein des êtres, et particulièrement le niveau de glycémie. Nous ne prétendons pas que ces êtres soient directement finalistes, mais nous argons que leur conception l'est : ils sont conçus pour collecter de l'énergie, pour grandir, afin de toujours pouvoir se reproduire.

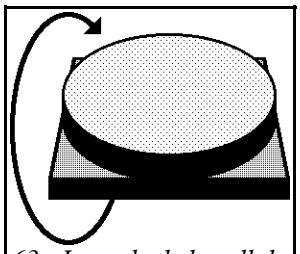
De par leur conception, la montée de leur niveau d'énergie est analysé par des capteurs et quand le niveau requis est atteint, cela peut déclencher des actions : grandir et se reproduire.

Leurs procès sont indirectement causals : ces êtres exécutent des actions

Ainsi nous avançons que ces êtres sont indirectement causals car leur conception est assurément causale. À ce titre, il ne reste plus d'objection au fait considérer que leurs procès qui concourent à cette collecte sont indirectement causals, et que, dans ce cadre, ces êtres exécutent des actions.

2) Quelques illustrations bioniques de l'introduction de l'énergie

2.A) La cellule végétative (CV)



63 - Le cycle de la cellule végétative

a) Description, présentation générale

La CV mime le cycle de la cellule végétative de la vie courante : à chaque cycle elle se reproduit identique à elle même.

b) L'extérieur de la CV :

De l'extérieur, la CV est une machine :

- Son support se limite à une unique case.
- Elle reste sur place donc son état demeure le même : *ÉtatActuel-(machine -lieu→ case)*.
- Les processus somatiques internes sont responsables de sa reproduction, qui se passe sur l'unique case du support.
- Au niveau macroscopique extérieur, selon la dimension *lieu*, elle reste sur place. Selon le paramètre *taille*, à chaque cycle elle présente la même taille (mais à chaque fois il y a reproduction cachée au sein de la boîte noire).

c) L'intérieur de la machine : comment elle est implémentée

La relation à l'énergie : comment elle est assimilée

La machine est implémentée par des mécanismes biologiques. Elle puise son énergie de façon diffuse et l'assimile de façon chimique, sans qu'il y ait mouvement, un peu comme les racines d'un arbre puisent l'eau du sol, ou encore comme les branchies des poissons assimilent l'oxygène, ou enfin comme la digestion.

L'implémentation du mécanisme de reproduction

La reproduction est effectuée par un mécanisme implémenté au sein de la boîte noire de la CV. Il scrute son état interne, i.e. son niveau de glycémie, et quand ses réserves de nourriture dépassent un seuil de consigne, il se reproduit.

Macroscopiquement son fonctionnement est bien modélisé par un système formel appliquant une unique règle : C→C.

Remarques sur le lien entre énergie et reproduction

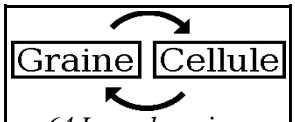
En un cycle unique, la CV assimile l'énergie externe pour grandir depuis la graine jusqu'à une cellule viable, produit une graine, et disparaît. Ceci amène deux remarques successives à propos du lien entre énergie et reproduction au sein de la CV :

1 - Dans une première approche macroscopique, l'observateur peut être amené à considérer que la reproduction de la cellule est inconditionnelle, ce qui se traduit bien dans la règle C→C.

2 - Dans une seconde approche microscopique, i.e. interne à la machine, l'observateur peut être amené à considérer que la reproduction de la cellule est conditionnée par l'arrivée du niveau de la réserve d'énergie au seuil de reproduction.

Ce second point de vue permet de souligner que le niveau d'énergie est la condition interne qui déclenche une règle somatique de reproduction. Ce point de vue permet de modéliser au sein de l'agent, les états internes somatiques que nous retrouverons dans les agents pulsionnels, car ils orientent son activité alors qu'ils consistent en des mécanismes inconscients : ils ne sont pas positionnés de façon volontaire par la cellule, en appliquant le jeu de règle(s) de réécriture du SF central de la cellule, mais ils sont géré par un système de règles somatique indépendant.

3) Le cycle graine-cellule (GC)



64 Le cycle graine-cellule

3.A) Description, présentation générale

La GC mime de façon plus fine le cycle de la cellule végétative. Elle distingue deux étapes : la graine et la cellule. Ainsi le cycle s'ouvre : la graine grandit et devient une cellule qui produit une graine et disparaît.

3.B) L'extérieur de la GC :

De l'extérieur, la GC est une machine :
- Son support se limite à une unique case.
- Elle reste sur place, mais change de nature. Son état oscille entre deux relations : *Graine-lieu→**C**ase* et *Cellule-lieu→**C**ase*.
- Elle se reproduit sur la même et unique case du support : elle ne bouge pas.
- Au niveau macroscopique extérieur, elle reste sur place, mais passe par deux tailles : G et C.

3.C) L'intérieur de la machine : comment elle est implémentée

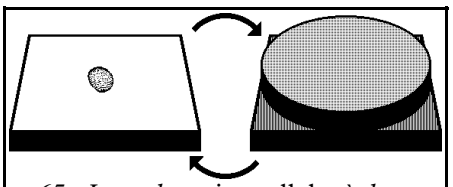
Remarque sur le lien entre énergie et reproduction

Le cycle initial de la CV éclate en deux étapes. Dans la première, la graine assimile l'énergie externe pour grandir jusqu'à donner une cellule viable, qui dans la seconde étape, produit une graine, et disparaît.

La relation à l'énergie : comment elle est assimilée

Un peu comme pour la CV, un mécanisme implémenté au sein de la boîte noire de la machine gère son état interne, i.e. son niveau de glycémie, et quand cette réserve de nourriture dépasse un seuil de consigne, il marque la fin de la croissance de la cellule.

Macroscopiquement ce fonctionnement est bien modélisé par un système formel appliquant la première règle du SF *G→C,* : la graine assimile l'énergie et grossit au point de devenir une cellule.

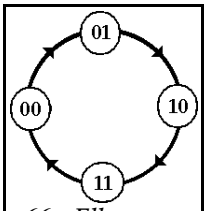


65 - Le cycle graine-cellule, *à deux temps*

L'implémentation du mécanisme de reproduction

Quand le mécanisme de gestion de l'énergie détecte la fin de la croissance de la cellule, il déclenche la phase de reproduction : la cellule produit une graine et disparaît. Macroscopiquement son fonctionnement est bien modélisé par un système formel appliquant la seconde règle du SF $C \rightarrow G$: la cellule disparaît et est remplacée par la graine.

4) La machine séquenceuse pulsionnelle (MSP)



66 - Elle passe par 4 tailles différentes

4.A) Description, présentation générale

La MSP mime de façon plus fine le cycle de la cellule végétative.

Comme sa précédente, la MS, elle présente quatre états : dans les trois premiers la graine grandit et devient une cellule qui, dans le dernier produit une graine et disparaît.

Mais elle reçoit un corps, une boîte noire, qui lui fournit des états somatiques, et pilote les transformations. À l'étape 11, cette dernière détecte le seuil de reproduction et enclenche deux processus : elle produit une graine (une cellule très petite), puis elle disparaît.

4.B) L'extérieur de la MSP :

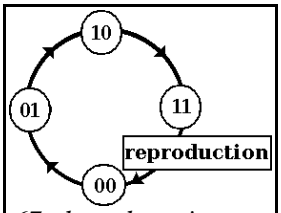
De l'extérieur, la MSP est une machine :

- Son support comporte quatre cases.
- À chaque instant élémentaire, elle change de lieu : elle passe à la case suivante. Son état est donc de la forme :

machine-taille $\rightarrow$ *taille*$_i$; avec i $\in$ [0, 3].

- Au sein de la machine, est implémenté le SF qui comporte quatre règles : 00 $\rightarrow$ 01, 01 $\rightarrow$ 10, 10 $\rightarrow$ 11, 11 $\rightarrow$ 00.
- Durant les trois premières transitions, l'observateur extérieur voit la cellule grandir. Durant la dernière, il la voit donner naissance à une nouvelle petite et disparaître.

4.C) L'intérieur de la machine :



67 - le cycle croissance reproduction de la machine séquenceuse

La relation à l'énergie : comment elle est assimilée.

L'assimilation de l'énergie au sein de la MSP est implémentée par des mécanismes biologiques. Elle puise son énergie qui est distribuée de façon diffuse dans le monde et l'assimile de selon un processus chimique, un peu comme les branchies des poissons assimilent l'oxygène. Comme certains poissons (requin, thon...).

Comme pour la CV, la MS assimile l'énergie et grossit au point de devenir une cellule, mais cette croissance dure 3 cycles.

L'implémentation du mécanisme de reproduction.

La reproduction est effectuée par un mécanisme implémentée au sein de la boîte noire de la machine. Elle est pilotée par la dernière règle du SF *11 $\rightarrow$ 00* : la cellule produit une graine et disparaît.

Remarque sur le lien entre énergie et reproduction.

Le cycle initial de la CV éclate en 4 étapes. Dans les trois premières, la graine assimile l'énergie externe pour grandir jusqu'à une cellule viable, qui, dans la dernière étape, produit une graine, et disparaît.

4.D) Conclusion : quel est l'intérêt de la MSP

Dans notre démarche d'incarnation des SF dans le monde réel, la MSP exécute un comportement causal, mais solipsiste.

La MSP est causale.

La cellule fonctionne selon la chaîne *perception décisions action*. En effet, dans ce modèle nous observons déjà la scrutation des capteurs afin de décider l'activation des effecteurs : ses capteurs scrutent son niveau de glycémie pour décider les étapes de sa croissance et de sa reproduction.

La MSP est solipsiste.

Elle est solipsiste car les seules perceptions qu'elle traite en entrée proviennent de ses états internes somatiques.

Perspectives.

Nous devons attendre qu'elle sorte de sa tour d'ivoire et s'ouvre au monde.

5) Transition :

D'abord complexifier le monde modélisé : introduction des lieux.

Dans le chapitre suivant nous introduirons une modélisation du mouvement. Pour obtenir des simulations plus réalistes, nous utiliserons des agents : le SF initial est remplacée par un agent situé qui avance sur le support de cases.

Ensuite progresser vers l'action de l'agent situé dans le monde : introduire l'action causale.

Ceci permet d'introduire dans le modèle la scrutation des capteurs pour décider l'activation des effecteurs, ainsi que la réaction du monde à ses actions.

Ch. 17 Matérialiser le séquenceur en une machine séquenceuse[29]

1) Introduction

Nous travaillons à formaliser une partie du domaine des agents situés.

type catégorie	*1) Sys.For. type 3*	*2) Sys.For. type 4*	*3) Machine séquenceuse*	*4) Cellule végétative*	*5) agent situé mini*	*6) Pluralité*
Nom commun	Automate fini	Séquenceur	Machine séquenceuse	Cellule végétative	agent situé minimal	Agent situé (AS)
Exemples :			jaquemart trotteuse	clignotant à deux états	cueilleur:thon requin	2DcueilleSurUnRéseau
Nature du cadre	mathématiques abstraites	mathématiques simplifié	physique, concret, moteur énergie ; solipsiste	vie artificielle solipsiste monde 0D : 1 seul lieu	agent sur un support constant	as en// simplifie support as non déterministe
quelle transition pour arriver ici?	départ depuis ici : l'automate fini	simplification de l'automate fini	*abstrait→concret, réel* se matérialise, s'incarne	introduit la vie, solipsiste autonome interne	situé sur un support, déplacement	mettre des agents actifs ou passifs en parallèle
Nature du procès effectué	règle de réécriture circuit de logique séquentielle	règle de réécriture circuit de logique séquentielle	moniste : mécanisme dualiste : prog. informatique ou système formel	solipsiste = f(interne) perçoit-décide-agit états, horloge internes	décision_cnst action = f (constante) états horloge extern	percept-décision-action = f(environnement) états & horloge externes

Colonne 1 et 2 : Simplifier la base math-info de l'AF pour obtenir un séquenceur.

Partis de la base mathématico-informatique de l'automate fini (AF) nous l'avons simplifiée afin d'obtenir un séquenceur solipsiste.

Colonne 3 : Nous matérialisons le séquenceur abstraite en un mécanisme physique concret.

Mais le séquenceur consiste en un être immatériel et abstrait. Maintenant, le moment est venu de matérialiser ses symboles en les habillant de matière. Ainsi nous pourrons traduire sa forme mathématique abstraite, afin d'obtenir une forme de surface concrète.

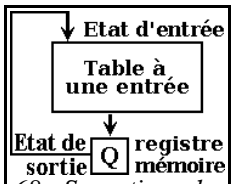
Dans un premier temps, avec ce chapitre, nous introduisons seulement le modèle d'une machine et de ses procès sur le monde.

Colonne 4, 5 et 6.

Puis, dans les chapitres suivants, nous introduirons un agent et le mettons en situation dans le monde afin qu'il y évolue librement.

2) Transformer le séquenceur en machine séquenceuse, en le matérialisant

La transition, qui transforme le séquenceur en machine séquenceuse(MS), se fait en traduisant sa forme profonde immatérielle vers une forme de surface concrète, où il est matérialisé : où ses symboles sont transformés en matière.



68 - Synoptique du séquenceur

2.A) Nous partons du séquenceur

Un séquenceur A est un quadruplet $\{Q, \delta, q_0, F\}$:

1) Q est un ensemble fini d'états. $Q=\{q_0, q_1... q_n\}$ où chacun des états q_i est un symbole formel
2) $\delta : Q \rightarrow Q$, la fonction de transition donnée par la table à une entrée, adressée par q, $q \in Q$.
3) q_0 est l'état de départ du système.
4) F, inclus dans l'ensemble Q, est l'ensemble des états finals.

29 Tout d'abord, nous convenons d'abréger *machine séquenceuse* en *MS*, parfois nous dirons aussi simplement *machine*.

2.B) La transformation ϕ : A$-\phi\rightarrow$ MS matérialise le séquenceur A en une MS

Introduction : matérialiser le séquenceur en un mécanisme d'horloge digitale

Dans ce paragraphe nous traitons l'aspect mathématique formel de la transformation ϕ**.** Cette même transformation est illustrée ensuite au moyen d'un exemple dans un paragraphe suivant, à la fin du chapitre.

Au séquenceur A, par l'opération de matérialisation ϕ : A$-\phi\rightarrow$ MS, nous faisons correspondre MS : MS = ϕ(A)

Partant de ce séquenceur A, au moyen de l'opération de matérialisation ϕ nous l'implémentons sans changer sa structure. Nous la traduisons en une structure matérielle concrète, et plus précisément dans un domaine spatio-temporel afin d'obtenir une machine séquenceuse (un mécanisme d'horloge qui compte en digital).

À l'arrivée, nous obtenons une aiguille, i.e. un pion, en relation de lieu avec une des cases d'un support.

séquenceur	$-\phi\rightarrow$	**machine séquenceuse**
Le domaine	$-\phi\rightarrow$	Le support
Chaque symbole du domaine	$-\phi\rightarrow$	Une des cases du support
Le lien *ÉtatActuel-symbole*	$-\phi\rightarrow$	Le lien *ÉtatActuel-lieu(aiguille,*$case_{symbole}$*)*
Le système formel	$-\phi\rightarrow$	La machine séquenceuse

À partir du séquenceur A, représenté par un quadruplet A= {Q, δ, q_0, F}
nous fabriquons la MS = ϕ(A), un quadruplet, MS= {S, R, ϕ(δ), lieu(aiguille,$case_0$)}

Fabrication/matérialisation du support S, un ensemble fini de lieux/cases

À partir de Q = {q_0, q_1, ... q_N}, un ensemble d'états, nous fabriquons, matérialisons le support *S = {*$case_0$*,* $case_1$*, ...* $case_N$*}*.

Chaque symbole q_i se matérialise en la case $case_i$ correspondante : $q_i-\phi\rightarrow case_i$, avec $case_i$ = $\phi(q_i)$

Ainsi transformé, le domaine Q se matérialise en un support de cases S.

Fabriquer/matérialiser R, l'ensemble de tous les états possibles de la machine séquenceuse

Dans un SF, l'état du système est donné par le lien *ÉtatActuel-symbole*, où *symbole* ∈ Q.

Dans la MS, la matérialisation de l'état du système correspond à la position de l'aiguille sur une des cases du support.

R = {r_0, r_1, ... r_N}, est un ensemble fini de relations r_i, d'états de la MS, i.e. où r_i = lieu(*aiguille*, $case_i$),

Fabrication/matérialisation de la transition d'état en un moteur

La fabrication/matérialisation de la transition d'état en un moteur consiste en un mécanisme qui produit un changement d'état

δ : Q→ Q, |Dans le SF, la fonction de transition est donnée
| par la table ou le jeu de règles.

δ_i : $q_i \rightarrow q_{i+1}$ |Chacune des règles δ_i produit la transition du
| symbole q_i au symbole q_{i+1}.

$\phi(\delta_i)$: $\phi(A= q_i)\rightarrow\phi(A= q_{i+1})$ |Dans le monde réel, la transition d'état est effectuée par
| un moteur, i.e. un mécanisme qui meut le pion/aiguille

$\phi(\delta_i)$: lieu(*aiguille,* $case_i$)→lieu(*aiguille, c*$case_{i+1}$)
|Elle correspond à la matérialisations de la
| transition produite par δ_i dans le SF.

Fabrication/matérialisation de l'état de départ du système

A= q_0 |Sachant l'ancien état de départ dans le SF

(A=q_0)$-\phi\rightarrow$lieu(aiguille**,**$case_0$) |Le nouvel état de départ dans MS est calculé
| en matérialisant l'ancien

ϕ(A=q_0) = lieu(aiguille**,**$case_0$) |Ainsi on obtient un résultat disant que l'aiguille
| est sur la $case_0$

2.C) La transformation de chaque composante de la transition *séquenceur→MS*

Illustrons la transformation de chaque composante de la transition séquenceur→MS

a) Quels sont les composantes de cette transition ?

Afin d'être plus clair, tout au long de ce paragraphe, prenons un exemple. Supposons que nous disposons initialement d'un séquenceur A = {Q, δ, q_0, F} et que nous le matérialisons en une MS= {S, R, $\phi(\delta)$, lieu(aiguille,$case_0$)}

Partant de ce séquenceur, nous le transformons en machine séquenceuse. Maintenant regardons, composante après composante, la transformation qu'elle subit, lors de la transition séquenceur$-\phi\rightarrow$machine.

b) Fabrication/matérialisation du support S, un ensemble fini de lieux/cases

Le domaine Q= *{0, 1, 2, 3, 4, 5, 6, 7}* est un ensemble fini de 8 symboles/états numériques ;

Chaque symbole du domaine devient une case

Chaque symbole *i* du système *Q*, $i \in [0, 7]$ se matérialise en la $case_i$ qui, quand c'est le moment, supporte le pion, i.e. l'aiguille digitale..

La matérialisation se décrit ainsi : $\forall i \in [0, 7]$; $i -\phi\rightarrow case_i$; avec $case_i = \phi(i)$.

Le domaine devient le support

À partir du domaine $Q = \{0, 1, 2, 3, 4, 5, 6, 7\}$ nous fabriquons, matérialisons le support S.

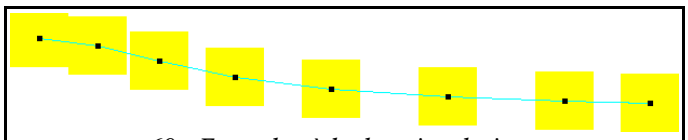
$S = \{case_0, case_1, case_2, case_3, case_4, case_5, case_6, case_7\}$ que nous abrégeons en : $S = \{c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7\}$.

Ainsi transformé, le domaine Q devient le support de cases S : $Q -\phi\rightarrow S$.

Le support initial devient un vrai couloir de cases

Nous venons d'énumérer les composants du support S, mais maintenant allons plus loin et décrivons les relations entre ses composants.

Dans le monde, l'aiguille est posée sur une des cases, afin qu'elle puisse ensuite aller de case en case, nous devons mettre les cases du support $S = (c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7)$, en relation.



69 - Exemple où le domaine devient un support de cases chaînées

D'abord, nous devons écrire que chacune est en relation de proximité avec sa suivante, alors nous spécifions : la relation *EstConnectéÀ*$(c_i, c_{i+1},)$;

De plus, pour obtenir un couloir de cases consécutives, nous mettons en relation la dernière avec la première. Ainsi nous obtenons un couloir rebouclé sur lui-même comme la piste d'un manège.

Finalement, ∀ i, i ∈ [0, 7] ; nous déterminons le couple de cases $(c_i, c_{i+1},)$ et pour chacun de ces couples, créons, modulo 8, la relation *EstConnectéÀ*$(c_i, c_{i+1},)$;

c) Fabriquer/matérialiser R, l'ensemble de tous les états possibles de la MS

Le lien *ÉtatActuel-i* devient le lien *ÉtatActuel-lieu(aiguille,*$case_i$*)*

Nous sommes dans une interprétation en situation. Chaque symbole *i* du système *Q*, devient une $case_i$ qui, quand c'est nécessaire, supporte le pion, i.e. l'aiguille digitale. Ainsi le lien *ÉtatActuel-i* devient le lien *ÉtatActuel-lieu(aiguille,*$case_i$*)*.

Ceci s'écrit :

- Dans le SF de départ, l'état du système est donné par le lien *ÉtatActuel-i*, où le symbole i∈Q
- Dans la MS d'arrivé, la matérialisation de l'état du système correspond à la position de l'aiguille sur une des $cases_i$ du support S.

R, l'ensemble de tous les états possibles de la machine séquenceuse

L'ensemble de tous les états possibles de la MS, est un ensemble fini R = {r_0, r_1, ... r_7}, de 7 relations r_i, i.e. de 7 états de la MS, où r_i = lieu(*aiguille*, *case*$_i$) ;

d) Description de la transition d'état

Dans ce SF, l'état du système est donné par le lien *ÉtatActuel-i*, où $i \in Q = [0, 7]$.

La fonction de transition δ dans le SF.

Dans le SF chacune des règles δ_i produit la transition du symbole i au symbole i+1.

- Le jeu de règles (la table δ : Q→ Q,) peut être défini en compréhension : δ_i : i→i+1.
- Elle peut être défini en extension δ = {0→1, 1→2, 2→3, 3→4, 4→5, 5→6, 6→7, 7→0}.

La fonction de transition $\phi(\delta)$ dans le monde réel correspond à un procès physique.

$\phi(\delta_i)$: ϕ(A= q_i) → ϕ(A= q_{i+1}) $\phi(\delta_i)$: lieu(*aiguille, i*)→lieu(*aiguille, i+1*)	- Dans le monde physique, la matérialisation de l'état du système r_i correspond à la position de l'aiguille sur une des cases du support. r_i = lieu(*aiguille*, *i*). - Dans le monde réel, la matérialisation de la transition d'état produite par δ_i dans le SF correspond à changement d'état du système, à un procès physique.

e) Fabrication/matérialisation de l'état de départ du système

A= 0	Sachant l'ancien état de départ dans le SF est 0, le nouvel état
(A= 0)−ϕ→lieu(aiguille,0)	de départ dans MS est calculé en matérialisant l'ancien,
ϕ(A=0) = lieu(aiguille,0)	Ainsi on obtient un résultat disant que l'aiguille est sur 0.

3) Transition d'état du séquenceur → procès de la machine dans le monde

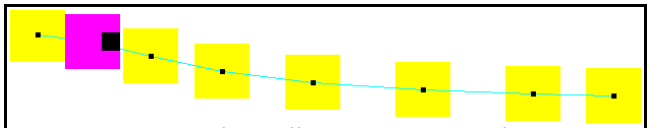
(La transition d'état du séquenceur devient le procès de la machine dans le monde)

3.A) Le système formel (SF) devient une machine séquenceuse (MS)

Le SF avant la projection/matérialisation.

Initialement le SF consiste en : un état, un jeu de règles et un mécanisme d'inférence.

- L'état du SF consiste en une relation ÉtatActuel-i

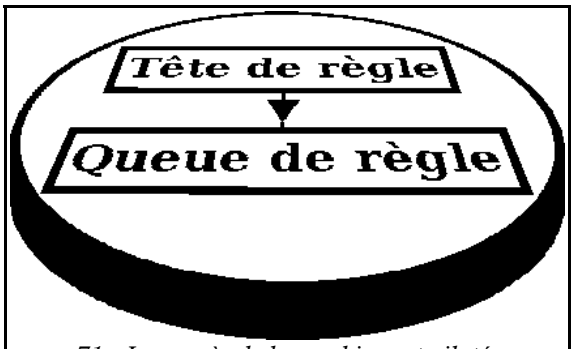


70 - En cyan, l'aiguille sur son support de cases

- Le jeu de règles (la table de transition δ : Q→ Q,) est défini en extension :
  0→1, 1→2, 2→3, 3→4, 4→5, 5→6, 6→7, 7→0
- Le mécanisme d'inférence applique une des règles à l'état actuel du monde pour obtenir l'état suivant de ce monde.

Après la projection/matérialisation, nous obtenons la machine.

Maintenant, dans le monde d'arrivée, l'aiguille figurée par un pion, devient un être autonome car il transporte en son sein, son système de décision : l'implémentation d'un SF. Convenons que ce système *tête de règle→queue de règle* peut y être implémenté sous forme physique, électrique, neurale...



71 - Le procès de la machine est piloté par un système formel

Résultat.

La machine séquenceuse matérialise le temps qui s'écoule. Nous pourrions dire qu'elle ressemble à un métronome, à une horloge, à une pendule. Mais dans ce cas, si l'aiguille se déplace lentement et sans à coup, elle illustre plutôt un encodage analogique de l'information. Or nous nous situons bien dans une approche logique.

Alors, afin d'obtenir un encodage digital/numérique de notre modèle, disons que la MS ressemble plutôt à la trotteuse d'une montre, ou à un jaquemart, qui indiquent les intervalles de temps en changeant de lieu, par à coups, à chaque instant élémentaire.

3.B) La transition d'état du séquenceur

Dans le séquenceur initial, à chaque application de règle, le système change d'état : Le lien *ÉtatActuel-symbole* reçoit une nouvelle valeur, un autre symbole du domaine. Ce changement est appelé une transition d'état.

3.C) Comment est effectué le procès physique qui change la position du pion ?

Après avoir plongé le séquenceur dans la réalité concrète, sur le support, à chaque intervalle élémentaire de temps, un procès physique change l'état du monde physique, il change la relation du pion/aiguille avec le support de lieux, i.e. le lien *ÉtatActuel-lieu(aiguille,case).*

Le procès physique qui change la position du pion/aiguille sur le support

Comment est effectué le procès physique qui change la position du pion ou de l'aiguille digitale sur le support ?

Selon une approche moniste matérialiste[30], il est effectué par un mécanisme physique qui met en mouvement, accélère et freine le pion, i.e. l'aiguille digitale. Dans ce cas, le traitement du temps est effectué directement par le mouvement d'un mécanisme physique qui meut le pion.

Selon une approche dualiste[31] *logiciel-matériel*, deux niveaux interviennent. Au niveau matériel, le procès est effectué par un moteur qui met en mouvement, accélère et freine le pion, i.e. l'aiguille digitale. Au niveau *information*, ce moteur est commandé par un procès de données effectué par l'exécution d'un programme ou d'un système formel.

4) Conclusions

La machine est solipsiste.

La machine effectue un procès sans percevoir le monde. Elle passe à l'état suivant simplement en le trouvant à partir de l'état actuel. Elle est solipsiste.

La machine n'est pas causale.

Puisqu'elle est solipsiste, elle fermée au monde, elle calcule son prochain état interne, mais ne prend pas de décision causale, i.e. en fonction de la situation perçue : elle n'est pas causale.

Dans le cadre de la machine séquenceuse :

Le monde M est un quadruplet, M= {S, Q, δ, q_0} :

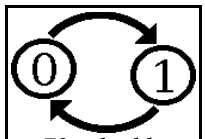
1) S le support est un ensemble fini de cases, chaque case c correspond à un symb : c= $case_{symbole}$
2) Q est un ensemble fini des états du monde. (de la forme *lieu(aiguille, $case_{symbole}$)*)
3) δ : Q→ Q, la fonction de transition est donnée par la table ou le jeu de règles
4) q_0 est l'état de départ du système.

30 En philosophie, le moniste est un matérialiste qui dit que *tout est matière et rien n'existe en dehors d'elle*.

31 En philosophie classique le dualisme *âme-esprit* distinguait l'esprit du corps. En informatique, on distingue programme et matériel, i.e. le traitement de l'information et son implémentation, ce qui constitue un autre type de dualisme.

5) Quelques illustrations de l'application des machines séquenceuses

5.A) Le clignotant double



72 - double clignotant

Description, présentation générale.

Le clignotant double est une machine à deux états, il matérialise visuellement le temps qui s'écoule en allumant alternativement une de ses lampes (approche dualiste information-matériel), ou plutôt en déplaçant un pion d'une de ses cases à l'autre (approche moniste matérialiste). Il s'apparente à une boîte à rythme paramétrée sur un rythme à deux temps, ou tout simplement à un métronome qui serait digital, i.e. qui changerait d'état par à coups.

L'extérieur du clignotant double :

Nous sommes dans une interprétation en situation. Chaque symbole *i* du système *D*, devient une $case_i$ qui, quand c'est nécessaire, supporte le pion, l'aiguille digitale.

Ainsi, de façon générale, le lien *ÉtatActuel-symbole* devient le lien *ÉtatActuel-(agent-lieu→*$case_{symbole}$*)*.

Ici, en l'occurrence, le domaine initial comporte deux symbole : D = {0,1}, on en déduit :

- le lien ÉtatActuel-0 devient le lien ÉtatActuel-(agent-lieu→$case_0$)
- le lien ÉtatActuel-1 devient le lien ÉtatActuel-(agent-lieu→$case_1$)

Plus concrètement, on en déduit que, vu de l'extérieur, le clignotant double est une machine séquenceuse :

-Chaque symbole du domaine S devient une case, son support comporte 2 cases. S={$case_0$, $case_1$,}

-À chaque instant élémentaire, il change de case : il passe à l'autre.

-Au sein de l'aiguille, est implémenté le SF qui comporte deux règles : 0→1, 1→0 ; ou plutôt, au sein du pion est un petit moteur qui lui permet de sauter d'une case à l'autre.

Comment est implémentée l'intérieur du clignotant ?

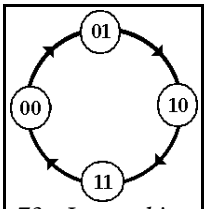
Selon une approche moniste matérialiste à l'intérieur du clignotant, est implémenté un mécanisme physique qui, à chaque intervalle élémentaire de temps, déplace d'un coup sec, le pion/aiguille.

Selon une approche dualiste (information-matériel) à l'intérieur du clignotant, est implémenté le système formel qui applique une règle à chaque intervalle élémentaire de temps.

Conclusion : quel est l'intérêt du clignotant double.

Pour la première fois, nous pouvons exhiber un procès élémentaire qui soit matérialisé dans le monde concret.

5.B) La machine séquenceuse (MS)



73 - La machine séquenceuse

Description, présentation générale.

La machine séquenceuse déroule une séquence dans le monde réel. Elle ressemble à un métronome, au jaquemart d'une horloge, à la trotteuse d'un montre, ou à une boîte à rythme. Elle matérialise le temps qui s'écoule en le scandant.

L'extérieur de la machine séquenceuse :

De l'extérieur, la machine séquenceuse est une mécanisme :

- Son support comporte quatre cases.
- À chaque instant élémentaire, elle change lieu : elle passe à la case suivante. Son état est donc de la forme : *lieu(aiguille, $case_i$)*; avec i ∈ [0, 3].
- Au sein de l'aiguille, est matérialisé le mécanisme qui implémenté le SF. Il comporte quatre règles : 00→01, 01→10, 10→11, 11→00.

L'intérieur de l'aiguille :

Par construction, à l'intérieur de la machine séquenceuse, nous trouvons le système formel qui applique une règle à chaque intervalle élémentaire de temps.

Bilan : quel est l'intérêt de la machine séquenceuse.

Voici enfin une implémentation concrète du séquenceur, cet outil central de notre construction.

6) Conclusion

Mécanisme de projection *forme de surface→forme profonde.*

Voici un mécanisme de morphogénèse *abstrait→concret* qui projette la forme profonde du SF vers sa forme de surface : la simulation profonde se fait avec les règles du SF, ensuite, le passage *forme initiale profonde→forme de surface* plonge cette forme profonde dans le monde réel. Ainsi nous obtenons en surface la machine réelle incarnée dans le monde quotidien.

À propos du déplacement.

Dans la MS, au niveau profond, le passage de case en case est encore implémenté au moyen d'un SF. Cette modélisation demeure abstraite, mais elle a l'avantage d'être simple, et nous permet ensuite d'obtenir une forme de surface en incarnant le SF dans le monde réel.

Conclusion à propos de ce chapitre.

Grâce à sa simplicité, cette simulation, basée sur les règles d'un SF, modélise le déplacement du pion, de l'aiguille digitale sur le support. Dans les chapitres suivants nous introduisons les agents et leurs actions que nous présentons comme des procès finalistes (sous-tendus par un but).

7) Perspective, col 4 : mettre en place un fonctionnement autonome interne

7.A) Le passage *colonne 2→colonne 3* constitue donc une étape intermédiaire

Donc cette simulation basée sur les machines, la transformation *forme abstraite→forme concrète de surface,* obtenue après matérialisation, constitue une étape intermédiaire entre les SF bien mathématiques (colonne 2), et les simulations de qualité où les agents situés autonomes planifient leur action (colonne 4, 5...).

type catégorie	***1) Sys.For. type 3***	***2) Sys.For. type 4***	***3) Machine séquenceuse***	***4) Cellule végétative***	***5) agent situé mini***	***6) Pluralité***
Nom commun	Automate fini	Séquenceur	Machine séquenceuse	Cellule végétative	agent situé minimal	Agent situé (AS)
Exemples :			jaquemart trotteuse	clignotant à deux états	cueilleur:thon requin	2DcueilleSurUnRéseau
Nature du cadre	mathématiques abstraites	mathématiques simplifié	physique, concret, moteur énergie ; solipsiste	vie artificielle solipsiste monde 0D : 1 seul lieu	agent sur un support constant	as en// simplifie support as non déterministe
quelle transition pour arriver ici?	départ depuis ici : l'automate fini	simplification de l'automate fini	*abstrait→concret, réel* se matérialise, s'incarne	introduit la vie, solipsiste autonome interne	situé sur un support, déplacement	mettre des agents actifs ou passifs en parallèle
Nature du procès effectué	règle de réécriture circuit de logique séquentielle	règle de réécriture circuit de logique séquentielle	moniste : mécanisme dualiste : prog. informatique ou système formel	solipsiste = f(interne) perçoit-décide-agit états, horloge internes	décision_cnst action = f (constante) états horloge extern	percept-décision-action = f(environnement) états & horloge externes

7.B) *Colonne 3→colonne 4* introduire la vie artificielle et la reproduction

D'abord, dans nos simulations, nous devons introduire la vie artificielle pour l'être soit doté d'un corps somatique et qu'il puisse se reproduire comme une cellule végétative.

Ensuite nous pourrons marcher en direction de simulations plus réalistes. Dans les simulations plus fines que nous modélisons ensuite, nous utilisons des agents, qui introduisent capteurs et effecteurs, ainsi que la réaction du monde à leurs actions.

Ch. 18 L'agent situé minimal (ASM), son actionneur et son action

1) Introduction

type catégorie	*1) Sys.For. type 3*	*2) Sys.For. type 4*	*3) Machine séquenceuse*	*4) Cellule végétative*	*5) agent situé mini*	*6) Pluralité*
Nom commun	Automate fini	Séquenceur	Machine séquenceuse	Cellule végétative	agent situé minimal	Agent situé (AS)
Exemples :			jaquemart trotteuse	clignotant à deux états	cueilleur:thon requin	2DcueilleSurUnRéseau
Nature du cadre	mathématiques abstraites	mathématiques simplifié	physique, concret, moteur énergie ; solipsiste	vie artificielle solipsiste monde 0D : 1 seul lieu	agent sur un support constant	as en// simplifie support as non déterministe
quelle transition pour arriver ici?	départ depuis ici : l'automate fini	simplification de l'automate fini	*abstrait → concret, réel* se matérialise, s'incarne	introduit la vie, solipsiste autonome interne	situé sur un support, déplacement	mettre des agents actifs ou passifs en parallèle
Nature du procès effectué	règle de réécriture circuit de logique séquentielle	règle de réécriture circuit de logique séquentielle	moniste : mécanisme dualiste : prog. informatique ou système formel	solipsiste = f(interne) perçoit-décide-agit états, horloge internes	décision_cnst action = f (constante) états horloge extern	percept-décision-action = f(environnement) états & horloge externes

Nous travaillons à formaliser une partie du domaine des agents situés :
- Col 1 : Nous partons de la base mathématico-informatique de l'automate fini (AF).
- Col 2 : Nous la simplifions pour obtenir un séquenceur solipsiste.
- Col 3 : La matérialisation transforme sa forme abstraite en un mécanisme physique concret.
- Col 4 : Nous digitalisons le fonctionnement autonome solipsiste de la cellule analogique.
- Col 5 : Nous mettons l'agent en situation dans le monde afin qu'il y évolue causalement.
- Col 6 : Mettons plusieurs agents en parallèle. Nous obtenons un groupe organisé en essaim.

2) Col 5 introduit le pilotage de l'agent autonome qui remplace le SF initial

Dans ce chapitre nous changeons l'âme du système. Nous délaissons son ancien animateur, la cellule solipsiste autonome, et le remplaçons par l'agent situé. Maintenant ce dernier reçoit le rythme initial de l'horloge centrale, et à chaque intervalle de temps élémentaire, exécute les actions qui vont animer le système.

Dans le tableau de la page 1, en colonne 1, 2 et 3 nous utilisions un SF pour animer la simulation. Déjà en colonne 4, nous le remplacions parfois par la vie interne du corps et son rythme autonome pour prendre le relais, et animer quelques instant le système.

Maintenant, dans ce chapitre correspondant à la colonne 5, nous délaissons totalement le SF pour le remplacer par l'ASM. Ce dernier, au moyen de son fonctionnement *décisionConstante-action calcule* ses interventions et anime la simulation. Cependant, il continue d'être perturbé par ses propres pulsions somatiques, introduites colonne 4.

Cet abandon du SF entraîne un changement important dans l'animation du système. Il marque la fin de la machine passive synchronisée par une base de temps extérieure, pour passer au rythme autonome de l'agent situé, même si parfois, il doit composer avec ses éventuelles pulsions internes somatiques. De plus nous gardons à l'esprit que l'agent lui même est aussi synchronisé par une base de temps externe.

3) Transition vers l'action de l'agent situé dans le monde

3.A) Introduire un mouvement piloté par un étage de décision

Maintenant, le moment est venu d'introduire dans le modèle un mouvement autonome des êtres : sur le support de cases du monde, la cellule devient un agent situé qui effectue une action à chaque intervalle élémentaire de temps.

Dans ce cadre, l'ASM perçoit ses états somatiques internes, et son environnement qui cependant s'avère constant car l'agent est seul sur le support de cases. Nous considérons qu'il décide de façon autonome ses actions et active en conséquence ses effecteurs (locaux ou globaux).

Le fonctionnement causal de sa boîte noire somatique est simple et ses quelques règles sont faciles à énoncer. Par contre, la décision de ses actions d'agent situé est réduite à la portion congrue : l'agent avance perpétuellement.

L'agent situé minimal agit sur le monde extérieur. Dans sa structure, nous introduisons un actionneur pilotés par son étage de décision. Il exécute des actions régulières sur le monde extérieur.

3.B) Le but des chapitres suivants est d'introduire progressivement l'agent situé

- Dans ce chapitre, nous présentons d'abord les actionneurs. Au travers des exemples de l'assimilateur (thon et requin) nous allons toucher du doigt leur fonctionnement générique : quelque soit la case sur laquelle ils sont situés, ils effectuent le même traitement.

- Dans le prochain chapitre nous montrerons comment ce fonctionnement débouche sur la formulation générale de l'action, qui consiste en un procès volontaire et générique.

- Ensuite, dans les chapitres suivant, nous introduirons les capteurs et la causalité.

3.C) L'agent situé décrit comme l'agent A situé sur un support S

Voici une description de l'agent situé formulée au moyen des mathématiques, nous expliquerons et justifierons ces notation dans ce chapitre et le suivant.

L'agent A et le support S permettent de construire le monde M.

Le monde M est un quadruplet $\{S, A, Q, q_0\}$: avec $A = (\varphi, \psi)$

S : Le support de cases, il est défini en extension

φ : Le physique, le soma de l'agent réactif (qui peut mourir de faim).

ψ : La psyché, ses règles. Soit $q_i, q_j \in Q$, $\psi : q_i \rightarrow act°_i$ et $(q_i, act°_i) \rightarrow q_j$, avec $q_j = \text{RéponseDeM}(q_i, act°_i)$;

Q : L'état de M, les relations entre ses composantes : *sur(A, Case$_i$) ;* Les *précède(Cases$_i$,Cases$_{i+1}$);*

q_0 : L'état du monde au départ : $q_0 \in Q$; q_0 = *sur (A, Case$_i$)*.

3.D) L'agent A6S : Agent Somatique Situé en Survie Sur un Support Simple

a) L'agent situé A6S

En informatique, on parle de l'agent situé, nous reprenons ce terme, mais nous précisons ce dont nous parlons. Pour nous l'agent situé est l'A6S : l'Agent Somatique Situé en Survie Sur un Support Simple

b) Agent situé sur un support simple

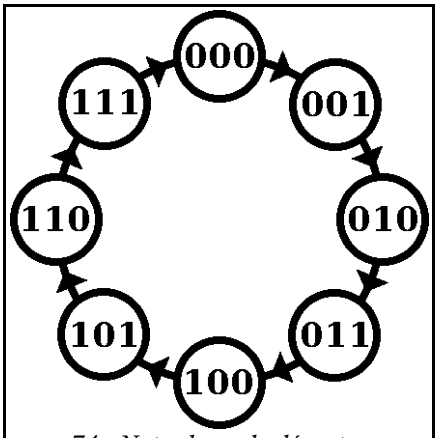
Les notions d'agent et d'action.
Elles sont précisées dans la suite de ce chapitre.

Nous introduisons un support :
Le monde est constitué de l'agent figuré par un pion ; et de son support, un chemin de cases qui se reboucle sur lui-même ; ex : le cycle du manège.

Le support est simple dans le sens où il est décrit en compréhension, comme un ensemble de lieux/cases mise en relation : *précède(Cases$_i$,Cases$_{i+1}$);*

La relation *sur* utilisée pour décrire la situation.
L'agent est situé en relation avec une des cases du support. Il est posé sur elle : *sur(A, Case$_i$).*

La relation du pion à son support est simple dans le sens où il ne peut pas y avoir plusieurs pions sur une case et un pion ne peut pas recouvrir plusieurs cases.



74 - Notre base de départ : le cycle du manège

La relation *sur* introduit la notion de procès.
L'agent est seul sur le support, il fait varier sa relation au support. À ce stade, cette opération ne constitue pas encore une action mais s'appelle plutôt un procès. On la décrit seulement comme un changement de la position du pion sur le support.

c) Agent somatique en survie

Nous introduisons φ le corps somatique.
Dans notre modèle, comme pour la cellule végétative, nous introduisons φ : le physique, le corps somatique de l'agent réactif.

Nous introduisons l'énergie.
L'agent situé est en relation avec une des cases de son support. Il fait varier cette relation au moyen du procès. Dans notre modèle physique, comme pour la cellule végétative, nous introduisons l'énergie : l'agent consomme de l'énergie pour faire varier sa position sur le support.

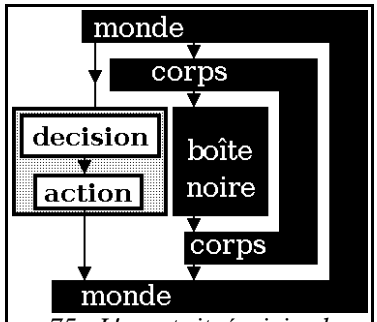
Dans les modélisations de l'agent situé, l'énergie est distribuée de façon diffuse. Nous convenons que l'agent l'assimile proportionnellement à son déplacement : à chaque étape il absorbe un peu de l'énergie du lieu. Il se déplace à la manière de certains poissons (requins, thons, etc.) : pour assimiler l'oxygène dissout dans l'eau, il doit toujours être en mouvement.

Le système végétatif.
L'agent est muni d'une boîte noire où est caché le système végétatif qui gère particulièrement le niveau de la réserve d'énergie. Quand le niveau d'énergie dépasse le seuil de reproduction, alors il se reproduit ; quand il passe dessous le seuil de survie, alors il disparaît.

Ce fonctionnement peut être illustré par deux exemples :

1 - Le cycle off-on : $0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 1$... du clignotant digital : un pion se déplace entre deux lieux appelés case$_0$ et case$_1$.

2 - Le cycle classique de l'agent qui tourne dans un manège octogonal : $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 0$. Quand l'agent arrive à la case$_7$ terminale, le circuit se reboucle et recommence en case$_0$, ce qui correspond à une graine qui recommence le cycle et grandit progressivement.



75 - L'agent situé minimal

Cette boîte noire perçoit l'état du corps et elle agit sur lui. Sur le synoptique de l'ASM cette relation est représentée par une flèche entrant et une flèche sortant de cette boîte.

3.E) Généralisation : le fonctionnement causal *décision action* d'un agent situé

a) L'intervalle élémentaire de temps de l'agent situé

Le système formel (SF), qui pilote l'agent est rythmé par un intervalle élémentaire de temps relatif à l'application d'une règle de réécriture : à chaque instant élémentaire le SF applique une règle et son état change. Cette pulsation du SF est transmise à l'agent situé qui, lui aussi, vit à ce même rythme.

b) Tout d'abord faisons un petit rappel

Partis avec la règle causale de l'AF : |$\delta : (Q \times \Sigma) \rightarrow Q$, fonction de transition donnée par la table de l'AFD.

Nous la simplifions avec le séquenceur solipsiste : |$\delta : Q \rightarrow Q$, la fonction de transition donnée par la table à 1 entrée.

Puis vient ψ : La psyché de l'agent situé : |soient $q_i, q_j \in Q$, $\psi : q_i \rightarrow act°_i$

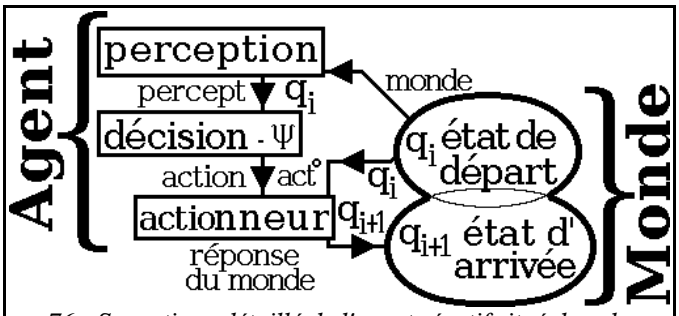
Et enfin on obtient: q_j, l'état d'arrivée du monde |avec la fonction $(q_i, act°_i) \rightarrow q_j$, qui donne la réponse de M à l'action.

c) Le fonctionnement *décision action* de l'agent situé

Le fonctionnement de l'agent réactif redevient causal.

La psyché constitue la fonction de décision $\psi : q_i \rightarrow act°_i$. À chaque état q du monde elle associe l'action à effectuer dans la situation donnée.

Avec l'ASM cette fonction se réduit à une règle inconditionnelle : toujours activer l'actionneur.



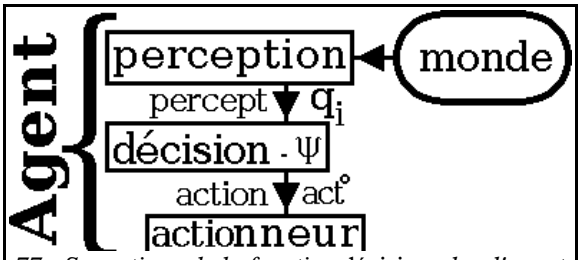
76 - Synoptique détaillé de l'agent réactif situé dans le monde

$\psi : q_i \rightarrow act°_i$ la partie *décision* de l'agent.

Maintenant détaillons $\psi : q_i \rightarrow act°_i$ le fonctionnement de la partie *décision* de l'agent.

Si on la considère comme un jeu de règles, on énonce que chaque règle de réécriture associe à l'état du monde q_i une action $act°_i$.

Si on la considère comme une fonction on la dessine comme ci-contre, sous la forme d'une fonction à une entrée qui, de la fonction de perception, reçoit le percept q_i et retourne $act°_i$ la commande de l'actionneur à activer.

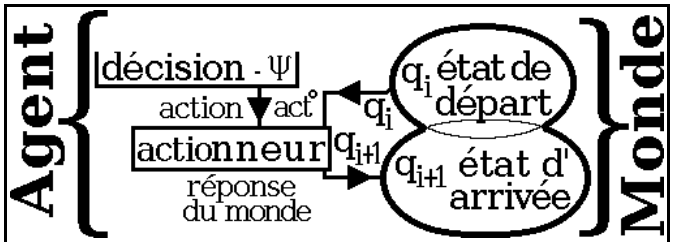


77 - Synoptique de la fonction décision *chez l'agent*

$(q_i, act°_i) \rightarrow q_j$, réponse du monde M à l'action effectuée.

Maintenant détaillons la réponse du monde à l'action de l'effecteur activé :

Ensuite, pour l'agent situé, comme pour l'agent réactif, cet état du monde q_i, est transformé par l'activation de l'actionneur sélectionné au moyen de



78 - Synoptique de la fonction action *chez l'agent*

l'application de la commande $act°_i$ et nous obtenons q_j, (noté q_{i+1}, sur le dessin[32]) nouvel état du monde M à cette action.

Pour l'agent situé, qui est seul sur le support, l'état du monde est de la forme : *sur(A, case$_i$)*. Dans le cadre de l'ASM, après l'action nous obtenons cet état du monde : *sur(A, case$_j$)*.

3.F) Application : implémentation concrète de l'architecture interne de l'ASM

Maintenant nous décrivons plus concrètement l'architecture interne de l'ASM car il constitue un exemple très simple d'agent situé.

3.G) Application : les fonctions de perception et de décision chez l'ASM

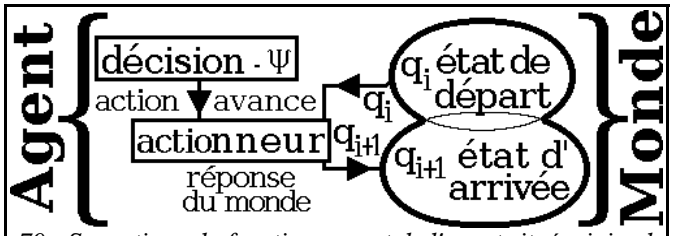
Nous plaçons l'ASM au début de notre progression de simulations. À chaque intervalle élémentaire de temps, il passe à la case suivante. Il tourne dans un manège dont la structure est très régulière. Précisons maintenant comment il active son actionneur.

a) La fonction de perception : pas de capteur/senseur

Décrivons petit à petit l'architecture de notre agent situé minimaliste. Nous venons de l'équiper d'un actionneur, puisqu'il est seul sur ce chemin de lieux dont l'ordonnancement est régulier, il n'a pas besoin d'observer le monde afin d'éviter les obstacles ou de rechercher sa nourriture. En fait, il n'a pas besoin de faculté de perception. Donc nous choisissons de ne pas le doter de capteur/senseur pour scruter le monde extérieur afin de décider son action. Ainsi il est aveugle.

b) La fonction de décision : un traitement très limité

À ce stade de la construction de notre agent, nous utilisons un centre de décision limité. En fait dire cela constitue un euphémisme.



79 - Synoptique du fonctionnement de l'agent situé minimal

Puisque notre agent ne possède pas de perception, la seule décision qu'il puisse prendre est d'activer en permanence son actionneur de mouvement : l'agent avance toujours : son effecteur est actionné, activé perpétuellement.

En conclusion, à chaque cycle, l'action d'avancer est exécutée inconditionnellement.

3.H) Application - L'actionneur est activé pour effectuer un procès générique

a) Rappel : le procès change l'ordonnancement du monde

Ici le procès change la relation *être ↔ support.*

Dès qu'on dispose d'un monde, même très simple i.e. un objet sur un support de cases, le simple fait de modifier cette relation *être ↔ support*, induit un changement, une transition d'état du monde, que nous appelons *procès*.

Implémentation d'un actionneur au sein du pion.

Pour construire cet agent situé nous y introduisons un actionneur qui induit un effet sur l'extérieur, il a pour rôle d'effectuer le procès, i.e. de changer la relation du pion au monde (à son support) : il effectue l'action :

Sur (Agent, Case$_n$) → Sur (Agent, Case$_{n+1}$) (Dans quelques lignes nous expliciterons cette notation)

Concrètement, nous savons que d'un côté, un mécanicien installe un actionneur en le fixant sur l'agent. De l'autre, il le pose sur le sol, au contact de la case. Ainsi quand l'effecteur est activé, puisque la case est immobile et l'agent libre, alors ce dernier bouge.

32 Ici l'état q_j du texte est noté q_{i+1} sur le dessin. Le lecteur peut consulter à la page suivante, les conventions de *notation pour les mondes réguliers*.

b) L'actionneur, une interface pilotée par l'agent qui effectue un procès sur le monde

L'actionneur, l'effecteur est un système physique qui change l'ordonnancement du monde quand il est activé par la commande $act°_i$. Par nature, il est piloté par une commande pour réaliser un procès sur le monde physique.

- D'abord, c'est un interprète car il reçoit de l'agent une commande qu'il interprète et traduit en action sur le monde.
- Ensuite, c'est une interface car il est positionné entre l'agent et le monde physique.
- Finalement, il agit concrètement sur le monde : il en change l'ordre, l'agencement.

Remarque à propos du traitement des erreurs

L'actionneur n'est pas infaillible, ce n'est pas parce que nous lui passons une commande que l' action correspondante sera exécutée. Le déroulement par l'effecteur d'une action pilotée par le module *décision* peut être entravé par différents facteurs extérieurs et intempestifs.

Puisque nous sommes inscrits dans un monde concret, parfois les actions peuvent être entravées, nous devons donc prendre en compte la réponse du monde M. Mais ici, l'ASM est seul sur une piste régulière, donc la réponse q_j du monde à l'action d'*avancer* est toujours positive.

Notation pour les mondes réguliers :

Dans le cas d'un monde très régulier comme ceux qui nous utilisons au début de la progression des simulations, nous pouvons remplacer la transition $q_i \rightarrow q_j$ par la forme q_i, q_{i+1}, ce qui sous-entend que topologiquement, les deux lieux du support sont successifs.

Par exemple, c'est cas pour l'ASM, et c'est ce qui explique notre notation :
Sur (Agent, $Case_n$*)* $\rightarrow$ *Sur (Agent,* $Case_{n+1}$*).* Après la $Case_n$, nous faisons apparaître la $Case_{n+1}$

c) Dans les mondes réguliers, l'actionneur effectue un procès générique

L'actionneur est activé par une commande qui lui dit d'agir. Bien souvent, dans les simulations simplifiées que nous étudierons, elle n'est pas paramétrée, ceci signifie qu'elle dit à l'actionneur : *fait ce que tu as à faire*. Ainsi, à environnement identique, la simulation effectue le même procès.

En effet, dans la plupart de nos modélisations, le monde décrit est très régulier, alors souvent le procès effectué par l'actionneur est invariant par translation et par rotation. Par exemple, quand on dit à un agent d'avancer, il avance quelque soit son lieu de départ. Quand on lui dit de tourner à gauche, il le fait, quelque soit son cap initial.

Prospective : note à propos de la suite de l'évolution.

Par la suite, quand l'agent se complexifie, il peut posséder plusieurs actionneurs, mais, comme à chaque instant élémentaire il effectue une unique action, la fonction ψ active un unique actionneur.

3.I) Le premier actionneur à paraître est celui de mouvement :

Alors que la nature fait preuve d'une grande imagination[33], notre progression de simulations est si contrainte que, paradoxalement, les actions y apparaissent de façon déterministe : il est inéluctable que, selon la construction que nous avons choisie, le mouvement apparaisse en premier.

Même s'il existe aussi des mondes 2D, naturellement nous commençons à travailler dans un monde 1D. Dans ce monde la relation unaire la plus simple qui puisse exister est celle de voisinage. Chaque case du support possède seulement deux voisins : un devant et un derrière. Nous obtenons une sorte de <u>chemin de lieux</u> ou de <u>couloir de cases</u>.

Donc, dans ce monde 1D, il n'existe pas de direction mais seulement deux sens. Pour l'instant, nous choisissons d'évacuer les changements de sens que nous traiterons par la suite dans le cadre de l'alternateur. Donc il reste à traiter les changements de relation *sur : sur(Être,* $Case_i$*).*

Nous devons démontrer que dans ce cadre, le seul procès possible pour l'agent consiste à avancer sur la case devant lui. Dans notre travail de simulation, nous ne pouvons pas nous

33 Ici nous évoquons le thème, très à la mode actuellement, de biodiversité.

éloigner de la réalité physique. Nous travaillons sur des objets pesants, dont nous traitons les déplacements le long d'un couloir. Donc, pour des raisons d'inertie des masses, ils ne peuvent pas sauter une case. Ainsi, dans notre modélisation, qui se veut réaliste, la téléportation n'existe pas.

En conclusion : soit un support 1D, prenons le cas d'un être sur une de ses cases. Naturellement, elle est seulement en relation avec deux autres cases, mais le sens de l'agent étant fixé, nous ne considérons uniquement la case qui est devant lui. Puisque pour des raisons d'inertie, il doit obligatoirement passer par cette case, il effectue donc cette action.

Le fait de terminer une relation *sur* avec une case où il est, pour en commencer une avec la case suivante, consiste à passer sur la case suivante (devant lui), et s'appelle *avancer*. Donc, dans cette condition, le seul procès susceptible d'être effectué, est celui d'*avancer*.

Finalement, ici, nous venons d'introduire la notion de mouvement au moyen d'un actionneur.

4) L'*assimilateur* : i.e. une introduction concrète d'actionneurs et d'action

Maintenant introduisons dans notre progression des exemples concrets de simulations illustrant les situations où nous sommes arrivées. Nous décrivons un être assimilateur, qui assimile l'énergie présente dans le milieu, sans faire de mouvement, i.e. au moyen d'un processus chimique. Plus précisément, nous focaliserons sur deux exemples d'assimilateur : le thon et le requin.

Tout d'abord, avec un support 1D en boucle, nous modélisons très bien le thon qui est pris dans un filet. Pour continuer de *respirer*, de *ventiler* ses branchies il doit avancer en permanence, et puisqu'il est prisonnier d'un filet, il tourne en rond.

Ensuite, Nous pouvons encore prendre l'exemple du requin. D'un côté, pour les mêmes raisons de ventilation, il doit se déplacer en permanence, mais comme il est libre, il parcourt un domaine infini.

4.A) Rappel sur l'action

a) La causalité pour passer du procès à l'action

En toute rigueur, selon un point de vue épistémologique, un procès ne devient une action que s'il a été effectué dans une démarche téléologique, finaliste i.e. orientée vers un but.

b) L'assimilateur : une conception intelligente

En reprenant l'argumentation développée dans le chapitre précédant, nous dirons que les êtres modélisés dans nos simulations proviennent d'une conception intelligente. Nous ne prétendons pas qu'il soient finalistes, mais nous argons que leur conception l'est : ils sont conçus pour assimiler de l'énergie, et implicitement ce design est étudié afin qu'ils puissent toujours vivre (et dans les simulations plus élaborées se reproduire).

Leurs procès sont indirectement finalistes : ces êtres exécutent des actions.

Ainsi nous avançons que ces êtres sont indirectement finalistes car leur conception l'est assurément. À ce titre, il ne reste plus d'objection à considérer leurs procès comme indirectement finalistes : ces êtres exécutent des actions.

c) Une action indirectement finaliste et faiblement causale

Ici nous obtenons une action qui est déjà causale : l'actionneur est relié à la fonction perception de l'être, il est donc activé par une commande. Mais, en toute rigueur nous devons dire qu'elle est faiblement causale, car cette commande de l'actionneur est perpétuellement activée, elle consiste en une fonction constante qui n'est pas modulée par la perception du monde.

Remarque par rapport aux actions causales des agents cueilleurs.

Deux chapitres après celui-ci, nous introduisons le parallélisme et décrivons les cueilleurs. Ce sont des agents causaux : quand ils trouvent de la nourriture, il font le geste de la cueillir. Avec

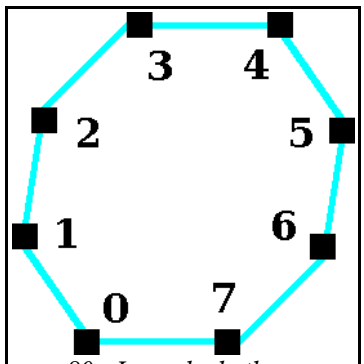
eux, nous obtenons une action causale (il décident de l'action en appliquant une règle), mais qui demeure encore indirectement finaliste (il n'ont pas élaborée la règle qui pilote leur comportement : elle est fournie par leur concepteur).

Quel est le coût à payer pour modéliser des actions finalistes ? D'un côté, les systèmes de la vie artificielle sont bon marché car ils sont faciles à implémenter, mais ils sont seulement indirectement finalistes. De l'autre, les puristes doivent payer très cher pour élaborer des systèmes réellement finalistes, car, en toute rigueur, il doivent au minimum simuler des agents planifieurs.

4.B) Un exemple simple d'assimilateur : le thon (prisonnier dans un filet)

Présentons le cycle du thon.

Il avance en permanence, et comme il se déplace en suivant le bord d'un filet rond, il tourne en boucle. Il ne grandit pas, ce n'est pas le but, mais en se déplaçant, il assimile l'énergie nécessaire à son métabolisme : il survit.



80 - Le cycle du thon dans le filet

Comment est-il modélisé ?

Dans notre modélisation, nous lui attribuons un domaine D = {0, 1, 2, 3, 4, 5, 6, 7}. Les cases sont liées séquentiellement et la dernière se reboucle sur la première. Le thon se déplace le long de ce parcours discrétisé de huit étapes. À chaque intervalle élémentaire de temps, il change de case.

Décrivons le SF qui le pilote.

Listons les huit règles du SF qui le pilotent. Elles sont encore peu nombreuses : c'est encore possible de les définir en extension :

$0 \rightarrow 1$, $1 \rightarrow 2$, $2 \rightarrow 3$, $3 \rightarrow 4$, $4 \rightarrow 5$, $5 \rightarrow 6$, $6 \rightarrow 7$, $7 \rightarrow 0$.

Remarque et conclusion :

En regardant ces règles, nous entrevoyons que l'action est régulière. Sa généralisation sera effectuée au chapitre suivant.

4.C) Le trajet du requin s'en va à l'infini

a) Présentons le trajet du requin

Il avance en permanence, et comme il se déplace en errant, il va au hasard, ne tourne pas en boucle et parcourt un trajet qui s'en va à l'infini.

En se déplaçant, il assimile de l'énergie. Elle ne lui sert pas à grandir, ce n'est pas le but, il en consomme la quantité nécessaire à son métabolisme.

b) Comment modéliser ce comportement ?

Modélisation mathématique.

Grâce à la puissance de la notation mathématique, nous pouvons utiliser une description en extension. Dans notre modélisation, nous lui attribuons un domaine infini D = {$case_i$ 0, $case_i$, ... $case_{i\ldots}$}, avec $i \in [0, 1 \ldots \infty[$.

Les cases sont liées séquentiellement et il n'y a pas de dernière car le parcours part à l'infini. Il se déplace le long de ce parcours discrétisé, ce qui fait qu'à chaque intervalle élémentaire de temps, il change de case.

Il suffit de définir son trajet sur la forme d'une liste : ($case_0$, $case_1$, ... $case_{i\ldots}$).

Modélisation informatique.

Dans une modélisation informatique, il est difficile, voire impossible de rendre la notion d'infini. Cependant nous avons choisi de créer un très long cycle de cases non alignées pour figurer la longueur et la diversité du parcours.

Les cases sont liées séquentiellement et la dernière se reboucle sur la première. Le requin se déplace le long de ce parcours discrétisé d'un grand nombre N d'étapes. À chaque intervalle élémentaire de temps, il change de case.

Il suffit de définir son trajet sous la forme d'une liste : ($case_i$, $case_i$, $case_k$, ... $case_{N-1}$, $case_i$).

81 - Parcours selon un cycle long

c) Décrivons le SF qui le pilote

Nous pourrions encore lister en extensions les N règles du SF qui pilotent la modélisation informatique, mais symboliquement nous choisissons de ne pas le faire.

En effet, en regardant ces règles, nous entrevoyons que l'action est régulière. Nous traiterons ce problème au chapitre suivant en utilisant une notation avec des variables pour effectuer une généralisation, mais, au point où nous en sommes, ce n'est pas le sujet.

5) Une double transition

Transition vers la causalité.

Dans les chapitres suivants, pour obtenir des actions causales chez les agents, nous introduirons dans le monde des objets que les agents devront attraper ou éviter. La chose peut se modéliser en introduisant le parallélisme dans les simulations.

Transition vers l'actionneur et vers l'action définie en compréhension.

Nous avons présenté le cycle du requin qui nous pose problème avec son cycle trop long. Nous avons énoncé que nous le traiterons plus rigoureusement la prochaine fois.

Ch. 19 Représenter formellement l'agent situé (A6S)

1) Tableau synthétique pour présenter notre démarche

type catégorie	*1) Sys.For. type 3*	*2) Sys.For. type 4*	*3) Machine séquenceuse*	*4) Cellule végétative*	*5) agent situé mini*	*6) Pluralité*
Nom commun	Automate fini	Séquenceur	Machine séquenceuse	Cellule végétative	agent situé minimal	Agent situé (AS)
Exemples :			jaquemart trotteuse	clignotant à deux états	cueilleur:thon requin	2DcueilleSurUnRéseau
Nature du cadre	mathématiques abstraites	mathématiques simplifié	physique, concret, moteur énergie ; solipsiste	vie artificielle solipsiste monde 0D : 1 seul lieu	agent sur un support constant	as en// simplifie support as non déterministe
quelle transition pour arriver ici?	départ depuis ici : l'automate fini	simplification de l'automate fini	*abstrait→concret, réel* se matérialise, s'incarne	introduit la vie, solipsiste autonome interne	situé sur un support, déplacement	mettre des agents actifs ou passifs en parallèle
Nature du procès effectué	règle de réécriture circuit de logique séquentielle	règle de réécriture circuit de logique séquentielle	moniste : mécanisme dualiste : prog. informatique ou système formel	solipsiste = f(interne) perçoit-décide-agit états, horloge internes	décision_cnst action = f (constante) états horloge extern	percept-décision-action = f(environnement) états & horloge externes

2) Nous travaillons à formaliser une partie du domaine des agents situés :

- Col 1 : Nous partons de la base mathématico-informatique de l'automate fini (AF).
- Col 2 : Nous la simplifions pour obtenir un séquenceur solipsiste.
- Col 3 : La matérialisation transforme sa forme abstraite en un mécanisme physique concret.
- Col 4 : Nous digitalisons le fonctionnement autonome solipsiste de la cellule analogique.
- Col 5 : Nous mettons l'agent en situation dans le monde afin qu'il y évolue causalement.
- Col 6 : Mettons plusieurs agents en parallèle. Nous obtenons un groupe organisé en essaim.

3) Colonne 5 : Description formelle de l'Agent Situé (A6S)

3.A) Rappel sur l'agent situé (A6S)

Dans le chapitre précédent, nous avons défini l'agent situé comme l'A6S : Agent Somatique Situé Sur un Support Simple.

3.B) Rappel sur un exemple d'agent situé : l'agent situé minimal (ASM)

L'ASM : un agent situé très simple.

Avec l'agent situé minimal (ASM – ex : le thon qui tourne dans le cycle du filet), nous venons de décrire un premier agent situé. Il est très simple :

- Son support est un monde 1D : il s'agit d'un cycle unidirectionnel.
- L'agent ne possède pas de capteur.
- L'agent ne possède qu'un unique actionneur qui lui sert pour avancer.
- Son centre de décision est très simple : il avance en permanence.

Sur son support, notre agent situé est seul.

Nous avons réussi à exhiber un agent presque complet (à part le fait qu'il ne possède pas de capteur), mais il demeure seul dans le monde. Non seulement, il n'est pas en présence d'autres agents, mais en plus, sur le support, on ne trouve pas de case contenant un obstacle ou un morceau de nourriture.

4) Continuer de formaliser le monde de l'agent situé sur son support

Dans le chapitre précédent nous venons de poser les premières bases d'un monde qui, dans la suite de notre progression de simulations, va se complexifier. Aussi, pour décrire ce monde qui évolue, nous devons continuer de formaliser sa représentation.

4.A) Notre but : continuer de formaliser ce monde régulier

Ce travail semble possible car ce monde est régulier. Ainsi, sa description formelle peut être simplifiée, car cet agent est seul, sur un support régulier, muni d'un comportement régulier.

En informatique, nous parlons beaucoup d'agent situé. Nous allons même formaliser cette notion, alors, nous devons d'abord le définir.

4.B) Rappel : la description formelle de l'agent situé

L'agent A et le support S permettent de construire le monde M.

Le monde M est un quadruplet {S, A, Q, q_0} : avec A = (φ, ψ)
S : Le support de cases, il est défini en extension
φ : Le physique, le soma de l'agent réactif (qui peut mourir de faim).
ψ : La psyché, ses règles. Soit $q_i, q_j \in Q$, $\psi : q_i \rightarrow act°_i$ et $(q_i, act°_i) \rightarrow q_j$, avec q_j = RéponseDeM(q_i, $act°_i$);
Q : L'état de M, les relations entre ses composantes : *sur(A, $Case_i$) ;* $\forall i \in [0, 7]$ *précède($Cases_i$,$Cases_{i+1}$);*
q_0 : L'état du monde au départ : $q_0 \in Q$; q_0 = *sur (A, $Case_i$).*

4.C) Rappel : le monde M est un quadruplet {S, A, Q, q_0} :

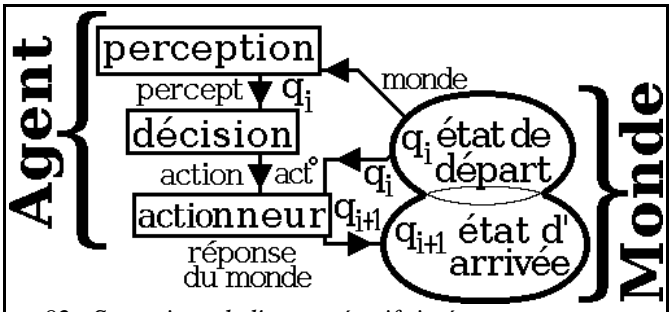
a) Le domaine S est le support de cases

S le domaine est un ensemble fini de cases. Il est déterministe, i.e. totalement connu : il est décrit en extension.

b) L'agent A = (φ, ψ)

φ le soma, état physique de l'agent.

C'est l'agent réactif correspondant. De plus, si nous sommes dans le cadre d'un agent en survie, si son état interne devient déficient alors l'agent meurt.



82 - Synoptique de l'agent réactif situé sur son support

ψ : la psyché de l'agent. C'est l'ensemble de ses règles comportementales.

$\psi : Q \rightarrow Q$, la fonction de transition donnée par la psyché ψ.

Soient $q_i, q_j \in Q$; ψ est de la forme $\psi : q_i \rightarrow q_j$;

et plus précisément ψ est de la forme $\psi : q_i \rightarrow act°_i$; puis $(act°_i, q_i) \rightarrow q_j$; Avec :

- $q_i \rightarrow act°_i$; |à l'état du monde perçu la décision de l'agent associe une action
- $(act°_i, q_i) \rightarrow q_j$; |Cette action change l'état du monde
- q_j = RéponseDeM(q_i, $act°_i$); |L'état du monde obtenu est la réponse du monde M à l'action.

c) L'état du monde Q

C'est l'ensemble des relations entre les composantes du monde. Au début, il est défini en extension, puis dans ce chapitre nos montrons comment nous passons à une définition en compréhension.

- La relation $A \leftrightarrow Q$: la position de l'agent sur le support : sur*(A, $case_i$) ;*
- Les relation $Q \leftrightarrow Q$; i.e. les relations entre les cases : chaque case est rattachée à la suivante (modulo 8).

d) Q_0 : l'état du monde au départ

$q_i \in Q$; q_i = *sur (A, case$_i$)...* Dans l'état du monde, l'agent est sur la case$_i$.

Ch. 20 Formaliser le monde de l'agent situé : *un agent sur un support*

Après avoir décrit le monde M comme un quadruplet {S, A, Q, q_0} où A = (φ, ψ) maintenant nous focalisons sur la description de q_t l'état de M, $q_t \in Q$, l'ensemble des états de M. Quand c'est possible, nous passons d'une représentation en extension vers une représentation en compréhension.

1) Représenter q_t l'état du monde M au temps t

Dans ce chapitre, pour décrire l'agent A, nous montrons comment et pourquoi passer vers une représentation en compréhension.

a) Pour décrire q_t l'état de M, nous devrons décrire deux choses

Les relations entre A et la $Case_i$.
Ainsi, nous explicitons le passage vers une représentation en compréhension : A est sur la case *: sur(A, $Case_i$) ;*

Les relations entre les composantes du support.
De même, nous explicitons le passage vers une représentation du support qui décrit les cases chaînées l'un à l'autre : $\forall i \in [0, 7]$ précède($Cases_i$,$Cases_{i+1}$);

2) Comment réduire la description du monde à celle d'un agent sur sa case

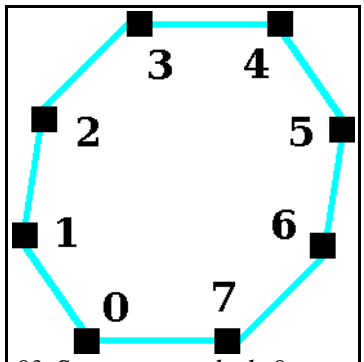
Dans le SF initial, l'état du système consistait en un symbole *q* mémorisé dans un registre (une mémoire). Maintenant que nous sommes dans le cadre de l'agent situé, cette représentation change. Cette mémoire continue d'enregistrer l'état du monde, mais ce dernier présente une structure plus complexe.

Normalement nous devrions représenter le monde comme un pion/agent sur une piste circulaire de cases. i.e. la relation de l'agent sur la $case_i$ où il est, celle de l'agent avec les autres cases, et les relations entre chaque couple de cases. Mais en fait nous choisissons de simplifier et finalement la représentation. Parmi ces relations, nous focalisons seulement sur celle entre A et la $Case_i$. Aussi nous décrivons seulement l'état d'un agent sous la forme d'un pion posé sur une des cases de la piste circulaire de cases : *sur(A, $Case_i$) ;*

3) Représenter les relations entre les cases du support

Après avoir représenté le pion sur une case du support, maintenant nous décrivons l'état de ce dernier sous la forme d'une piste circulaire de cases.

Puisque nous représentons un état, nous utilisons une représentation statique : d'abord sous une forme extensive, car elle est ainsi initialement; puis sous une forme compréhensive, car c'est plus pratique et la régularité du support nous le permet.



83 Support en cycle de 8 cases placées dans un monde 2D

3.A) Plongeon dans la réalité physique

Dans un premier temps nous utilisons un repère extérieur, le monde 2D externe, et nous y posons notre piste de cases. Par la suite nous réussirons à éviter cet ajout, et nous l'évacuerons.

Pour simplifier l'exposé, nous supposons que dans la réalité d'un monde physique 2D, le support de 8 cases connectées en cycle, se présente comme illustré dans la figure de droite.

3.B) Représentation statique du système : méthode extensive

p\n°	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
x	57	9	24	99	208	256	241	166
y	256	174	84	10	9	92	182	256

D'un côté, on pourrait parfaitement représenter représenter extensivement l'état du système. Pour ce faire, il faudrait fournir toutes les données brutes à propos des composants du monde (l'agent et son support).

Par exemple, on pourrait penser à présenter le fichier informatique binaire du dessin du monde, mais cette représentation en extension du système est aride. En effet, l'utilisateur croule sous les données brutes difficilement exploitables car peu significatives.

Alors on pourrait encore penser présenter le tableau ci-dessus, qui donne les position en x et y des cases sur le support externe 2D. Mais cette représentation en extension demeure peu parlante.

Finalement, l'usage veut que, pour communiquer, on cesse de représenter les données extensivement, et qu'on privilégie une représentation en compréhension, i.e. analytique.

3.C) Représentation statique du système *agent-support* : méthode compréhensive

Pour décrire l'état du cycle de cases, nous avons listé les relations de lieu de ses composantes avec le monde 2D externe (leurs position x et y). Mais cette méthode présente l'inconvénient d'introduire un référent externe au système. Pour l'évacuer, nous choisissons alors d'utiliser une démarche interne au monde, nous décrivons les relations entre ses composantes : le pion et les cases.

Pour décrire l'état du monde à un auditeur, au lieu de lister des n données brutes correspondant à la position de chaque objet dans le monde, le locuteur analyse la scène observée et cherche à fournir un résultat qui soit interne au monde et plus exploitable. Pour cela, il part du produit cartésien des relations entre tous les composants, et il communique seulement celles qui apparaissent vraies (et significatives).

	Agent	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
Agent		Sur							
C_0	Sous		Pred.						Succ.
C_1		Succ.		Pred.					
C_2			Succ.		Pred.				
C_3				Succ.		Pred.			
C_4					Succ.		Pred.		
C_5						Succ.		Pred.	
C_6							Succ.		Pred.
C_7		Pred.						Succ.	

84 - Produit cartésien des relations entre les objets du monde

Cette méthode présente deux avantages : Elle évacue le référent 2D externe et elle condense l'information en la synthétisant.

Par exemple, voyons ce que cette méthode peut donner quand on l'utilise pour représenter le pion en relation avec son support en cycle de 8 cases (ci-dessus à droite).

Initialisons l'état du système

Au départ le pion est sur *Case0*, la première case.

Première démarche d'analyse : scène statique.

Dans une premier temps, nous décrivons seulement une scène statique : nous ne représentons pas le mouvement.

Le tableau ci-contre correspond au produit cartésiens des relations entre les composants du monde, il se décompose en trois parties :

Sa diagonale correspond aux relations réfléchies R(x, x) qui ne sont donc pas l'objet de notre étude, c'est pourquoi nous neutralisons ses cases en les dessinant en gris.

Maintenant analysons les relations *agent-support* (ligne jaune et colonne magenta). Au préalable, nous avons délibérément dimensionné l'agent à la taille du support ; de plus nous sommes dans un domaine discret, alors l'agent ne peut pas être à cheval sur plusieurs cases, et réciproquement une case ne peut pas contenir plusieurs agents. Ainsi la relation *agent-case* se réduit à une case unique : **elle est univoque**. Nous analysons seulement la relation *sur (A, $Case_0$)* et/ou sa relation réciproque *sous($Case_0$, Agent)*. Puisque le pion est posé sur une seule case, selon cette représentation analytique, la quantité d'information est nettement moindre : elle se réduit une relation *sur* et à sa réciproque *sous*.

Ensuite, regardons les relations au sein du support (le cycle de cases), ce sont des rapports de proximité entre les lieux (les deux triangles rectangles colorés en cyan). Au préalable, nous avons délibérément choisi une organisation ordonnée des cases. Nous les orientons dans le sens croissant et ne gardons que les relations les plus significatives : il ne reste donc que celles de *prédécesseur* et *successeur*. Cela suffit pour obtenir une information pertinente.

Mais, nous constatons que le tableau obtenu est creux. Alors, pour obtenir une information synthétique, nous la condensons en fournissant à l'utilisateur seulement la liste des relations vérifiées et significatives. Finalement l'information pertinente se réduit à cette liste : *C_0 précède C_1, C_1 précède C_2,* et de façon générale *C_i précède C_{i+1}.*
Note 1 : de façon parallèle, on pourrait travailler avec la relation réciproque *succède*.
Note 2 : en notation préfixée on obtient la forme générale : $\forall i \in [0, 7]$ *précède(C_i, C_{i+1}).;*

Conclusion : attention, nous obtenons une représentation pratique mais imparfaite

À ce stade, finalement, nous décrivons Q l'état du monde, en compréhension, i.e. au moyen d'une liste de relations entre ses composants (entre son support et son mobilier) : $Q = sur(A, Case_i) \wedge \forall i \in [0, 7], précède(Cases_i, Cases_{i+1})$.

Assurément, cette représentation est plus pratique, mais il faut garder à l'esprit que nous venons de faire entrer le loup dans la bergerie : nous venons de quitter une représentation lourde, mais parfaite et complexe ; pour une qui est légère mais imparfaite parce que simplifiée et donc incomplète.

Selon un point de vue philosophique, nous touchons du doigt, nous disséquons un phénomène clé : celui de l'abandon du monde parfait mais irréel des mathématiques, pour le passage vers celui réel mais imparfait du quotidien.

4) Représentation séquentielle du procès de l'agent situé

Nous venons de simplifier la description du support S en utilisant une méthode en compréhension. Maintenant ce nouveau résultat nous sert à simplifier la représentation du procès effectué par l'agent situé.

4.A) Introduction générale, globale : le procès consiste en une transition d'état

Étape 2 dans la démarche d'analyse, une scène dynamique pour représenter le mouvement

Ensuite, dans un second temps, quand, à chaque impulsion de chargement du registre, le système change d'état en passant de q_i à q_j (avec q_i, $q_j \in Q$), nous passons à une seconde démarche et l'analysons alors dynamiquement. Afin de fournir à l'utilisateur, une représentation en compréhension de son état, nous ne pointons que les faits significatifs, i.e. nous ne signalons que les relations changées au fil du temps. Évidemment le support, le cycle de case, est statique, rien n'y bouge ; donc il nous reste seulement à analyser les relations agent ↔ support.

Ainsi, au départ de la simulation, lors de la première impulsion, dans la description du monde, la relation *sur (Agent, $Case_0$)* disparaît et devient *sur (Agent, $Case_1$)*. Finalement nous généralisons le processus et décrivons ainsi le système : à l'étape n+1, *sur (Agent, $Case_n$)* disparaît et devient *sur (Agent, $Case_{n+1}$)*, ce qui se verbalise : *l'agent quitte la $case_n$ pour aller sur la $case_{n+1}$.*

Conclusion, le procès : un changement de relation entre les composants du monde

Dans le cadre des SF l'état du système se réduisait à un symbole et les règles le réécrivaient pour changer cet état. Nous avons nommé *procès* cette opération.

Par contre, maintenant, dans le cadre des agents situés, l'état du système se décrit par la liste des relations entre les objets du monde. Alors, quand ce système change d'état, à chaque cycle, le procès y change quelques relations.

Ici, le procès est décrit synthétiquement comme un changement de position du pion

Prenons l'exemple de notre agent situé construit avec un séquenceur en cycle sur 8 cases ; l'état du système s'y réduit à la relation *sur(Agent, Case)*. Ensuite, dans le cadre de la simulation, ce système change d'état, et pour décrire cette scène de façon synthétique, nous focalisons sur cette relation et verbalisons : à chaque cycle, le procès change la relation pion/support, il déplace l'agent sur son support, i.e. il passe le pion à la case suivante.

5) Outils pour décrire le fonctionnement axiomatique du procès résultant

5.A) Maintenant décrivons le procès de l'agent situé sur un support

À ce stade nous savons que, dans le cadre des agents situés, nous représentons le procès subi par le monde, au moyen d' une variation des relations au sein de ses composants. Maintenant nous allons rechercher les outils et les moyens de décrire plus précisément le procès qui intervient particulièrement dans notre exemple : l'agent dans un cycle de lieux, i.e. le pion sur un manège de cases.

Dans ce cadre, la combinatoire demeure raisonnable car nous sommes sur un support à une dimension, mais dans les simulations suivantes, où le monde passe à deux dimensions, elle devient rédhibitoire. C'est pourquoi, rapidement nous cherchons à simplifier sa description.

5.B) Sur plusieurs aspects, le monde de l'agent situé est très régulier

La régularité du domaine induit une régularité des relations.

Dans les simulations de la progression construites autour de l'agent situé, nous introduisons des supports réguliers. Dans les mondes à une dimension nous utilisons le couloir ou le chemin de lieux (rebouclé ou non). Dans les mondes à deux dimensions nous utilisons la grille ou des graphes réguliers. Chez certains de ces support 2D, chaque lieu possède 4 voisins ; chez d'autres, chaque lieu en possède 6. Par exemple, au sein de l'ensemble de nos simulations, chez certaines, nous plaçons l'agent sur la piste d'un manège, i.e. un cycle de 8 cases rebouclées ; chez d'autres sur un graphe en structure de nid d'abeilles. Ainsi, dans tous ces exemples, nous obtenons des relations très régulières entre les lieux.

L'agent situé entretient une relation binaire avec son support.

De plus, nous prescrivons que la relation entretenue entre l'agent/pion et son support soit, elle aussi, très régulière. Nous imposons qu'elle soit digitale : le pion est sur le lieuA ou le lieuB, i.e. il est forcément sur une case mais pas à cheval sur les deux.

L'agent et les cases du support présentent une taille standardisée.

En fait, nous choisissons un cadre encore plus contraint. Nous aurions pu prendre un pion fabriqué en assemblant plusieurs pions unitaires. Mais nous prenons à dessein, un pion juste à la bonne taille : une case ne peut pas contenir plusieurs pions, et réciproquement, un pion ne peut couvrir plusieurs cases.

Les cases du support et leur contenu sont standardisés et donc indifférentiables.

Déjà à ce stade nous commençons à deviner l'évolution des simulations de la suite de notre progression :

Les objets du monde eux-mêmes sont très réguliers, au point qu'ils se classent dans très peu de catégories. Au sein de la même classe ils sont d'aspect identique et produisent le même effet. Ainsi l'agent ne les distingue pas : par exemple, il confond lieuA et lieuB et ne voit qu'un lieu.

De même, par la suite, ce qui est valable pour les conteneurs le sera aussi pour leur contenu. Si plusieurs cases contiennent la même matière (ou encore pièce de matière), l'agent ne distinguera pas de différence qualitative entre le contenu de case1 et celui de case 2 : il ne perçoit pas les atomes, donc il confond leur contenu en une même matière (en une même morceau de matière).

5.C) Les règles avec variables décrivent les procès (*transitions d'état des actions*)

Actions et procès changent l'état du monde. Plusieurs raisons nous amènent à décrire les transitions d'état qu'ils produisent en utilisant des règles munies de variables. Par exemple, voyons comment nous obtenons cette formule itérative, qui nous sert pour décrire l'action d'*avancer* :

$\forall n \in [0, 7]$, *sur (Agent, Case*$_n$*)* $\rightarrow$ *sur (Agent, Case*$_{n+1}$*).*

a) Agent sur support <u>simple</u> : le traitement digital induit une régularité du <u>domaine</u>

Dans le paragraphe précédent, nous avons indiqué que le monde présente une grande <u>régularité</u>. Ceci induit plusieurs simplifications dans les traitements que l'agent situé lui fait subir.

b) Cessons d'individualiser les objets : le monde est constitué d'un mobilier standard

En plongeant un SF dans un monde concret, les éléments du domaine deviennent les objets d'un monde digital. Si nous y plaçons des objets de différentes classes, puisque nous les avons choisis de taille unitaire, nous devons leur ajoutons un type afin de pouvoir distinguer leur nature.

Ainsi la perception de l'agent change. Contrairement au moteur de règles des SF, il ne voit plus un symbole distinct de tous les autres, qui étiquette un objet unique. Au contraire il perçoit uniquement l'instance d'une classe : il voit un objet, différencié seulement par son type, par exemple il ne voit qu'un lieu, i.e. un objet anonyme, membre de la classe des lieux.

c) <u>Procès à effet local</u> sur les objets du monde

Un agent situé sur notre manège de cases exécute des actions réduites par l'effet d'horizon : leur portée se limite à la case située face à lui.

Le procès/action que subit le pion dans le manège se réduit à une variation de lieu

Au début, les seules relations existant au sein du séquenceur sont celles de lieu. Donc les seuls procès/actions effectués y sont des variations de lieu. L'actionneur est monté/installé sur l'agent qui lui n'est fixé à rien ; alors quand cet effecteur est activé, l'agent bouge et passe à la case suivante.

d) L'actionneur fait <u>un procès</u> générique, souvent invariant par translation rotation

L'actionneur est activé par une commande qui lui dit d'agir. Bien souvent, dans les simulations que nous étudierons, elle n'est pas paramétrée, ce qui signifie qu'elle dit à l'actionneur : fait ce que tu as à faire. Ainsi, à environnement identique, elle effectue le même procès. Souvent le procès effectué par l'actionneur est invariant par translation et même par rotation.

En effet, par exemple, quand on dit à un agent d'avancer, il avance quelque soit son lieu et sa direction. Quand on lui dit de tourner à gauche, il le fait, quelque soit son cap initial.

e) <u>Le traitement</u> devient générique, il cesse d'individualiser les objets et les confond

Puisque les objets ne sont pas individualisés : l'actionneur ne voit et traite que des lieux ou encore des matières indistinctes ; alors le traitement effectué est <u>général</u> à la classe de chaque objet.

f) La règle qui décrit l'action est régulière, i.e. générique

L'action devient générique : sa formulation comprend des variables.

En passant de séquenceur à agent situé, <u>le procès sur le monde change</u>. Contrairement aux règles de réécriture des SF, où l'opération agit sur un symbole pour le remplacer par un autre, ici

l'agent traite seulement un objet typé, une instance de classe, qui se tient dans son environnement. Et comme, à dessein, le monde est très régulier, les objets traités par l'agent situé sont du même type, alors l'action devient indifférenciée : la même action est appliquée à toutes les instances de la classe. L'action devient générique : ainsi sa formulation comprend des variables.

Regardons de plus près notre agent situé sur le manège de cases.

Nous sommes partis avec la définition initiale du séquenceur, $\delta : Q \rightarrow Q$, la fonction de transition est donnée par la table/dictionnaire à une entrée Q. $\delta : \{0 \rightarrow 1, 1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4, 4 \rightarrow 5, 5 \rightarrow 6, 6 \rightarrow 7, 7 \rightarrow 0\}$.

Puis nous avons incarné le séquenceur dans le monde réel en le matérialisant. Maintenant nous venons d'entrer dans le cadre de l'agent situé minimal. Au début sa relation au monde sera non destructrice et puisque le mode d'action de l'agent situé est le procès : changer quelques relations au sein du monde, son premier mode d'action est le changement de relation de lieu.

Grâce à la généralisation, une règle suffit à décrire le procès du pion sur le manège

Nous avons déjà montré que, dans ce cadre construit à dessein, très peu de procès, de règles sont possibles. Dans ces premières simulations d'agents situés, de pion sur un manège, nous travaillons seulement avec la classe des cases :

Maintenant, pour cet agent situé, au fil du temps, la base de règle pourrait s'écrire ainsi :

$sur\ (Agent, Case_0) \rightarrow sur\ (Agent, Case_1)$ // règle applicable au temps 1

$sur\ (Agent, Case_1) \rightarrow sur\ (Agent, Case_2)$ // règle applicable au temps 2

...

Mais nous avons clairement établi que les $Case_i$ sont interchangeables, alors, rapidement, il apparaît que toutes ces règles peuvent se condenser dans l'unique formulation suivante :

$\forall n \in [0, 7],\ sur\ (Agent, Case_n) \rightarrow sur\ (Agent, Case_{n+1})$ // sous cette écriture, nous // obtenons une formule itérative

Dans le cadre si régulier où nous travaillons, elle suffit à décrire le traitement effectué. À ce stade nous venons d'introduire le procès, puis de le décrire, définir, de façon univoque.

Intérêt de l'actionneur : son activation suffit à provoquer le procès de l'agent.

Nous avons implémenté le support S pour que chacune de ses cases soit en relation avec la case suivante. Nous savons que quand on active un actionneur de mouvement, son travail consiste à passer à la case suivante, alors quand le centre de décision de l'agent active son actionneur, l'agent avance : il passe à la case suivante.

6) Définissons, décrivons la prise de décision qui pilote le procès

Après avoir montré comment décrire le procès, nous travaillons maintenant à modéliser ψ , la psyché.

6.A) Traitement de la partie ($\psi : q_i \rightarrow act°_i$)

Tout d'abord, dans la modélisation de la psyché ψ, nous abordons rapidement la partie $\psi : q_i \rightarrow act°_i$. Il s'agit de générer l'action en fonction de l'état du monde. En fait elle correspond à celle que nous avons déjà décrite pour l'ASM qui avance seul sur son manège de cases. Il ne perçoit rien et active perpétuellement son actionneur *avancer*.

6.B) Traitement de la partie : RéponseDeM(q_i, $act°_i$);

Ensuite, dans la modélisation de la psyché ψ, nous abordons aussi rapidement la partie RéponseDeM(q_i, $act°_i$);. Il s'agit de simuler la façon dont le monde répond aux actions des effecteurs en fonction son état. En fait, cette réponse correspond à celle que nous avons déjà décrite pour l'ASM : nous convenons que, puisque il est seul sur son manège de cases, les déplacements se déroulent toujours bien.

6.C) Conclusion à propos de la prise de décision de l'action dans l'agent situé

Paradoxalement, nous avons tellement simplifié le système que, chez notre agent situé, la prise de décision est déterministe. Mais, rapidement ce ne sera plus le cas dans les modèles suivants de notre progression : elle deviendra non déterministe et empirique.

Ch. 21 Agents situés : le parallélisme induit les capteurs et la causalité

1) Tableau synthétique pour présenter notre démarche

type catégorie	*1) Sys.For. type 3*	*2) Sys.For. type 4*	*3) Machine séquenceuse*	*4) Cellule végétative*	*5) agent situé mini*	*6) Pluralité*
Nom commun	Automate fini	Séquenceur	Machine séquenceuse	Cellule végétative	agent situé minimal	Agent situé (AS)
Exemples :			jaquemart trotteuse	clignotant à deux états	cueilleur:thon requin	2DcueilleSurUnRéseau
Nature du cadre	mathématiques abstraites	mathématiques simplifié	physique, concret, moteur énergie ; solipsiste	vie artificielle solipsiste monde 0D : 1 seul lieu	agent sur un support constant	as en// simplifie support as non déterministe
quelle transition pour arriver ici?	départ depuis ici : l'automate fini	simplification de l'automate fini	*abstrait→concret, réel* se matérialise, s'incarne	introduit la vie, solipsiste autonome interne	situé sur un support, déplacement	mettre des agents actifs ou passifs en parallèle
Nature du procès effectué	règle de réécriture circuit de logique séquentielle	règle de réécriture circuit de logique séquentielle	moniste : mécanisme dualiste : prog. informatique ou système formel	solipsiste = f(interne) perçoit-décide-agit états, horloge internes	décision_cnst action = f (constante) états horloge extern	percept-décision-action = f(environnement) états & horloge externes

2) Nous travaillons à formaliser une partie du domaine des agents situés :

- Col 1 : Nous partons de la base mathématico-informatique de l'automate fini (AF).
- Col 2 : Nous la simplifions pour obtenir un séquenceur solipsiste.
- Col 3 : La matérialisation transforme sa forme abstraite en un mécanisme physique concret.
- Col 4 : Nous digitalisons le fonctionnement autonome solipsiste de la cellule analogique.
- Col 5 : Nous mettons l'agent en situation dans le monde afin qu'il y évolue causalement.
- Col 6 : Le parallélisme induit la causalité, la décision réactive de l'action. Multitude inorganisée, mais pas de groupe.

3) Colonne 6 : introduire le parallélisme induit les capteurs et la causalité

Dans la colonne 6, pour introduire un agent situé causal nous plaçons plusieurs objets ou agents en parallèle sur le support.

Dans le chapitre précédent, qui correspondait à la colonne 5, nous avons introduit un unique agent situé dans un monde très simple, seulement composé d'un chemin de lieux. Nous l'avons muni d'un actionneur et regardé son action : *passer dans la case située devant lui*. Nous avons vu comment ce procès régulier, exécuté sur le monde, pouvait se représenter au moyen d'un énoncé général : *avancer dans la case située devant lui*. C'est pourquoi nous avons pu aussi le représenter au moyen d'une formule mathématique.

De plus, nous avons précisé que cet agent est un assimilateur : il ne fait pas le geste de manger, il est constamment en mouvement afin de pouvoir assimiler son énergie dans son milieu.

Maintenant, dans ce chapitre, pour obtenir des actions causales chez les agents, nous introduisons des objets dans le monde. Selon un point de vue informatique, cette modification peut se modéliser en introduisant le parallélisme dans la simulation : nous positionnons des agents sur le même support. Dans ce cadre multi-agents les êtres interagissent, et ceci induit l'apparition des capteurs et du traitement causal de cette perception. L'agent, au moyen de son fonctionnement *perception-décision-action* calcule ses interventions et anime la simulation.

Cependant, il continue d'être sensible à ses propres pulsions somatiques, introduites dans la colonne 4.

4) Le paradis perdu ou la transition vers les herbivores et les objets

Avertissement : ce texte constitue un jalon plausible de notre progression.

Le texte qui suit n'est pas forcément à prendre au pied de la lettre. Son rôle est double : il pose un jalon le long de notre progression, et s'attache à le rendre plausible.

Au départ l'assimilateur, un être mal adapté.

Sur un monde 2D nous avons commencé avec un agent situé *assimilateur* mal adapté à son environnement. Il est seulement capable d'assimiler la nourriture diffuse de l'environnement : il ne fait pas le geste de stocker la nourriture dans une poche où elle serait plus facilement assimilée.

Introduction de la reproduction.

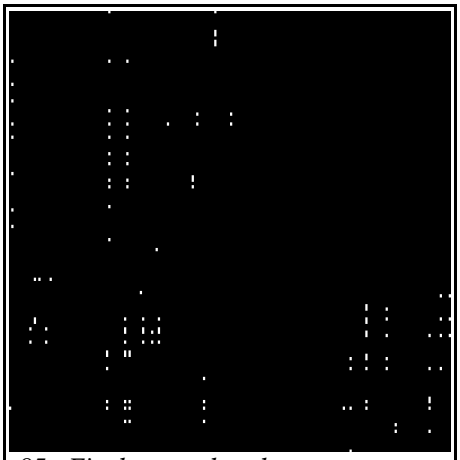
Ensuite, dans une optique d'évolution artificielle, intervient une mutation, l'être qui se reproduit ne disparaît pas de suite. Ceci a pour effet d'augmenter exponentiellement le nombre des herbivores, ce qui induit chez eux une pression de sélection.

Invention d'une autre méthode pour assimiler l'énergie.

Selon le point de vue du darwinisme, cette pression de sélection induit une modification physique chez les agents qui leur permet de mieux assimiler la nourriture : les êtres mangent, ils moissonnent leur nourriture et la stockent dans une poche au contact de l'organe d'assimilation.

Imaginons un monde digital et anguleux.

Imaginons que le support soit une grille de cases (damier, échiquier). Supposons que quand un agent enfante, il pose sa progéniture à 90°. Cette attitude favorise le nouveau né car ainsi il ne cherche pas sa nourriture dans les traces de son parent. Elle constitue un avantage fonctionnel qui est renforcé par la sélection naturelle.



85 - Finalement, dans le paysage apparaissent des îlots de nourriture distribués aléatoirement

Apparaissent les îlots de nourriture.

Sur leur passage, ces agents moissonneurs laissent des traces : là où ils paissent, l'herbe ne repousse pas de sitôt. Ainsi dans le paysage apparaissent des îlots de nourriture. Finalement la distribution initiale de nourriture disparaît. Elle n'est plus distribuée uniformément, mais apparaissent de petit lots ponctuels.

Naissance de la causalité.

Avec une telle distribution de la nourriture, les êtres ont intérêt à adapter leur comportement : ne pas chercher à récolter et stocker de la nourriture s'il n'y en a pas devant eux. Ainsi, apparaît chez eux un comportement causal : ils font le geste de cueillir qui si de la nourriture apparaît dans la case devant eux.

5) Mettre en oeuvre deux agents identiques en parallèle

Auparavant notre problème était d'écrire et modéliser un agent situé, maintenant il devient : *écrire et modéliser deux agents situés en parallèle*. Pour cela nous réinvestissons le résultat de notre travail passé.

5.A) Pour modéliser, nous partons de la définition initiale de l'agent situé :

Le monde M est un quadruplet {S, A, Q, q_0} : avec A = (φ, ψ)
S : Le support de cases, il est défini en extension
φ : Le physique, le soma de l'agent réactif (qui peut mourir de faim).
ψ : La psyché, ses règles. Soit q_i, $q_j \in Q$, ψ : $q_i \rightarrow act°_i$ et $(q_i, act°_i) \rightarrow q_j$, avec q_j = RéponseDeM(q_i, $act°_i$);
Q : L'état de M, les relations entre ses composantes : *sur(A, case$_i$) ;* $\forall i \in [0, 7]$ *précède(Cases$_i$,Cases$_{i+1}$);*
q_0 : L'état du monde au départ : $q_0 \in Q$; q_0 = *sur (A, case$_i$)*.

5.B) Et nous plaçons deux agents situés en parallèle sur le même support

Ceci nous amène à écrire :
M est un quadruplet {S, A, Q, q_0} : avec A = (φ, ψ).
N est un quadruplet {S, B, Q, q_0} : avec B = (φ, ψ).

5.C) Maintenant, explicitons cette notation

Le domaine S est le support de cases.
Nous plaçons les deux agents identiques sur le même support S.

Les agent A et B.
A et B sont deux instances d'une même classe d'objets : ils ont le même soma φ, et la même psyché ψ (le même comportement). Ceci est possible car le bon fonctionnement des règles est invariant par translation sur le support : avancer depuis la *case$_2$* ou la *case$_5$* relève du même mécanisme, et marche pareillement.

L'état du monde Q.
C'est l'ensemble des relations entre les composantes du monde. Déjà avec l'agent situé, il était défini en compréhension. Bien sûr nous continuons dans cette direction, et analysons systématiquement les relations croisées entre les composants du monde (A, B et S).

- La relation A ↔ S :	la position de l'agent sur le support : sur*(A, case$_k$) ;*
- La relation B ↔ S :	la position de l'agent sur le support : sur*(B, case$_l$) ;*
- La relation A ↔ B :	la position relative des deux agents. Ils sont sur deux cases différentes : case$_k$ ≠ *case$_l$*.
- Les relations S ↔ S :	i.e. les relations entre les cases : chaque case est rattachée à la suivante (modulo 8).

Si nécessaire, toutes les autres relations se déduisent de ces 4 familles.

Q_0 : l'état du monde au départ.
$q_a \in Q$; q_a = *sur (A, case$_a$)...* Dans l'état du monde, l'agent est sur la case$_a$.
$q_b \in Q$; q_b = *sur (B, case$_b$)...* Dans l'état du monde, l'agent est sur la case$_b$.
Ici nous retrouvons la contrainte de la position relative des deux agents. Ils sont sur des cases différentes : case$_a$ ≠ *case$_b$*.

6) Mettre en oeuvre deux agents différents en parallèle

Nous avons montré comment décrire et modéliser deux agents identiques en parallèle sur un même support. Maintenant nous allons traiter le cas de deux agents différents. Cette situation correspond par exemple à un agent qui rencontre n'importe quelle matière (de la nourriture ou un obstacle).

Bien sûr nous cherchons à réinvestir le résultat de notre travail passé. Il nous suffit de considérer que *qui peut le plus peut le moins*, et de modéliser une case de matière comme un agent dégradé, i.e. aux compétences moindres.

6.A) Pour modéliser, nous partons de la définition initiale de l'agent situé :

Le monde M est un quadruplet $\{S, A, Q, q_0\}$: avec $A = (\varphi, \psi)$.

6.B) Et nous plaçons deux agents différents en parallèle sur le même support

Ceci nous amène à écrire :
M est un quadruplet $\{S, A, Q, q_0\}$: avec $A = (\varphi, \psi)$.
N est un quadruplet $\{S, B, Q, q_0\}$: avec $B = (\varphi', \psi')$.

6.C) Maintenant, explicitons cette notation

Le domaine S est le support de cases.
Nous plaçons les deux agents identiques sur le même support S.

Les agent A et B.
A et B sont deux instances de deux classes d'objets différents : ils ont des soma psyché différents car ils sont passifs.

L'état du monde Q et l'état du monde au départ Q_0.
La modélisation de l'état du monde relève du positionnement des agents qui est ne diffère pas entre l'agent et les matières. Alors ces deux rubriques demeure les mêmes et elles sont donc toujours valables.

7) Mettre en oeuvre N agents en parallèle

Après avoir mis en parallèle deux agents, identiques ou non, maintenant nous généralisons la démarche, pour animer une simulation de N agents en parallèle.

Pour effectuer cette modélisation, nous partons de la définition initiale de l'agent situé, où le monde M est un quadruplet :	M : {S, A,	Q, q_0} ; avec un unique agent A
Et nous plaçons ces N agents en parallèle sur le même support. Ainsi nous écrivons que M devient :	M : {S, {A_0, A_1 ... A_{N-1}},	Q, q_0} ; avec N agents A_0, A_1 ... A_{N-1}

7.A) Maintenant, explicitons cette notation

Le domaine S est le support de cases.
Nous conservons le même support S, où nous plaçons les N agents.

Les agent de l'ensemble $\{A_0, A_1 ... A_{N-1}\}$.
Les agents A_i sont des instances d'une ou plusieurs classes d'agents. Ils sont définis par héritage : ils reçoivent leurs traits de leur classe mère.

En effet, leur soma (structure physique) et leur psyché (comportement) sont hérités de leur parent. Cette opération suffit à les définir, car, là où ils sont sur le support, ils fonctionnent bien. Rappel : nous avons montré que le fonctionnement de leurs règles est invariant par translation, à condition de prendre en compte le parallélisme[34].

L'état du monde Q.
Avec ces N agents sur le support S, nous devons particulièrement focaliser sur les relations $A_i \leftrightarrow A_j$, quand $i \neq j$; nous retrouvons à l'échelle du groupe la contrainte que nous avons précédemment énoncée pour le couple : deux agents ne peuvent être positionnés sur la même case.

Q_0 : l'état du monde au départ.

34 Le traitement en parallèle du comportement des agents est décrit quelques lignes plus bas.

Avant la simulation, le programmeur doit positionner les N agents. Ce travail est fastidieux, mais bien souvent il peut être effectué de façon simple, au moyen d'une méthode aléatoire.

7.B) La gestion en parallèle de l'action des agents

a) L'effet d'horizon

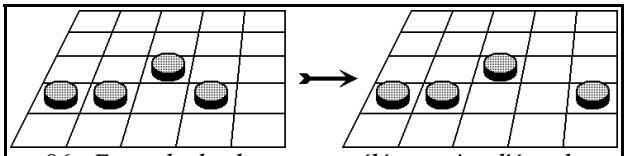
Nous continuons d'utiliser un modèle où les agent subissent un effet d'horizon. La simulation des agents dispersés sur le support n'est pas trop difficile car ils agissent en fonction de leur environnement local : à cause de cet effet, ils ne voient pas plus loin que le bout de leur nez, et donc ne prennent en compte que la case ou les cases immédiatement à côté.

b) Codage des relations statiques inter agents

La description, le codage des relations statiques entre N agents peut devenir fastidieux. La solution consiste à ne coder que les relations agent↔support. En effet, cette opération est suffisante, car, par la suite, il est possible de partir d'elles pour déduire et calculer toutes les relations inter agents. Ceci évite donc au programmeur un travail chronophage.

c) Gestion de l'interactivité

Pour animer la simulation de N agents sur un support, à chaque intervalle élémentaire de temps, nous devons calculer pour chacun, leur nouvelle situation (position direction énergie ...). Cependant nous devons distinguer deux méthodes de gestion du parallélisme.



86 - Exemple de changement élémentaire d'état du monde

Actualisation synchrone et tamponnée.

Normalement, la méthode rigoureuse de calcul de la nouvelle situation au temps *t+1* (à droite) doit se faire en fonction de celle au temps *t* (à gauche). Pour ce faire, il faudra figer cette dernière, effectuer le calcul de la nouvelle position dans un support tampon, et quand il est entièrement rempli, l'utiliser pour actualiser la nouvelle situation.

Cette méthode présente au moins deux inconvénients :
- D'abord elle est lourde à mettre en oeuvre, le registre tampon fait qu'elle prend de la place en mémoire.
- Mais surtout la gestion des collisions est difficile. Aucun agent n'est prioritaire sur l'autre, alors quand deux êtres veulent aller au même endroit, comment gérer la situation ?

Nous avons déjà montré que, dans le cadre des agents situés, nos simulations ne sont plus déterministes pour trois raisons : d'abord le support est défini en compréhension, ensuite l'effet d'horizon limite la perception des agent, enfin parce que leur jeu de règles est écrit selon une stratégie non déterministe en largeur d'abord. Donc la rigueur de cette simulation synchrone induit une charge de travail qui est inutile, puisque déjà le *navire déterministe* prend l'eau de toutes parts.

Alors, en fait, dans toute notre progression de simulations, nous utilisons cette méthode synchrone seulement pour implémenter le jeu de la vie de John Horton Conway, où elle est s'avère absolument nécessaire. Pour le reste des simulations, nous utilisons l'actualisation asynchrone et non tamponnée.

Actualisation asynchrone et non tamponnée.

À chaque intervalle de temps, vu la situation au temps *t*, pour calculer la nouvelle situation au temps *t+1*, nous prospectons l'un après l'autre chacun des agents., nous repérons son environnement, et, au moyen de ses règles nous calculons l'action qu'il décide, puis la réponse du monde à cette action locale, et nous actualisons localement le support.

`Pour chaque intervalle de temps t, faire :`	D'abord on itère sur tous les intervalles de temps
`Pour chaque agent i, faire :`	Ensuite on itère sur tous les agents
`début`	
`env = CalculDeLEnvironnement(i);`	- Pour chaque agent on calcule son env, ie ce qu'il voit
`action= CalculerLActionDécidée(env);`	- En fonct° de ce qu'il voit calcule l'action qu'il décide

`S=ExécuterLActionSurCeSupport(action, S);` `fin`	- Sur le support, on exécute l'action décidée pour obtenir le nouvel état du support.

Nous avons déjà dit que dans le cadre des agents situés, nos simulations ne sont plus déterministes pour trois raisons. Quand nous utilisons cette actualisation asynchrone, nous rajoutons encore de l'empirisme à notre modélisation car cette méthode aussi est non déterministe.

Cette méthode d'actualisation asynchrone est non déterministe à cause ne serait-ce que par sa gestion des collisions. En effet, dans le programme nous écrivons que si la case devant l'agent n'est pas vide, nous ne cherchons pas à avancer. Donc, réciproquement, si deux agents sont susceptibles d'avancer pour occuper la même case, le premier d'entre eux qui sera activé pourra avancer, au détriment du second, qui sera bloqué. Il apparaît clairement ici que l'ordre dans lequel on choisit de traiter les agents joue un rôle, ce que explique l'aspect non déterministe de la méthode.

Mais en fait cet inconvénient est négligeable si on effectue un lissage sur la masse des agents et finalement cette actualisation fonctionne correctement.

Conclusion.

Finalement, nos simulations sont non déterministes pour quatre raisons.

7.C) Introduction des agents passifs

Dans nos simulations, les agent passifs correspondent aux lots de nourriture et aux obstacles distribués sur le support.

Ils sont modélisés comme des agents, mais ils sont passifs : ils n'agissent pas.

Par contre ils peuvent avoir une vie somatique végétative. Par exemple, l'herbe, quand elle est mangée, repousse.

8) Résultat : pour deux raisons l'agent situé est non-déterministe

Après avoir modélisé l'agent situé au moyen d'une formalisation mathématique, nous pouvons clairement dire que son fonctionnement en situation est non déterministe pour au moins deux raisons.

8.A) Énoncer les deux raisons du non déterminisme

a) Le non déterminisme du comportement de l'agent en situation dans le monde

L'effet d'horizon de la perception de l'agent situé l'amène à une prise de décision non déterministe.

Décrire l'effet d'horizon de la perception.

Pour représenter l'agent en relation avec les autres composantes du monde (lieux, matières, obstacles ou autres agents), nous délaissons la <u>représentation rigoureuse en extension</u>, pour préférer une <u>représentation simplifiée en compréhension</u>.

Pour ce faire, nous limitons les relations de l'agent avec son voisinage, au moyen d'un horizon de profondeur 0 ou 1 : parfois nous décrivons seulement ses relations avec la case où il est situé (profondeur 0), et parfois nous rajoutons ses relations aux cases voisines(profondeur 1).

L'agent dispose d'une information incomplète : son comportement est non déterministe.

Ainsi, ces deux représentations sont limitées et sont donc incomplètes. En effet, pour planifier son comportement, il ne dispose pas d'une connaissance parfaite i.e. entière du monde. Dans ces conditions, il ne peut plus générer une tactique parfaite, mais seulement mettre en place une stratégie empirique : un raisonnement basé seulement sur ces environnement restreint est forcément imparfait. Finalement, l'agent n'est plus déterministe.

b) Le non déterminisme de la réponse du monde

Les simplifications abusives pré-supposent un monde régulier et des interactions locales, mais ce n'est pas forcément le cas en tous lieux du support, c'est pourquoi la réponse du monde à l'action de l'agent est elle aussi non déterministe.

Les règles de l'agent situé est basé sur des suppositions qui ne sont pas toujours exactes.

Pour agir en situation dans le monde, l'agent situé active des règles comportementales très simples, qui se formulent en termes de variables. Mais elles présupposent une régularité du monde, qui n'est pas toujours vérifiée : elle peuvent donc se révéler parfois fausses. Mais alors, quel est l'intérêt d'utiliser des telles règles ?

Mais par lissage statistique, l'agent trouve un bénéfice à les utiliser.

Pour en comprendre la raison, il faut revenir à notre introduction qui explique la position de l' émergence relativement à l'optimisation. Dans le cadre de la vie artificielle, si l'être en situation de compétition évolue, i.e. si un mutant est doté de cette nouvelle règle de conduite, alors il y trouve généralement un avantage. Finalement, même si la règle n'est pas parfaite et peut s'avérer fausse, il peut jeûner car il dispose sur lui d'une petite réserve d'énergie. Alors, par effet de lissage, puisque globalement, grâce à cette règle, il trouve un bénéfice, cette mutation est renforcée, i.e. sélectionnée.

8.B) Conclusion

a) Une transition très paradoxale du déterminisme au non-déterminisme

Nous avons décrit une transition depuis l'automate fini déterministe, et nous arrivons finalement à un agent non- déterministe. Paradoxalement, sur ce registre du déterminisme nous sommes passé du blanc au noir, d'un extrême à l'autre.

b) Ensuite ce non-déterminisme marque une transition lourde de conséquences

Dans le système que nous obtenons, à la fois, le comportement de l'agent situé et la réponse du support à son action sont non déterministes. C'est typiquement ce qui révulse les mathématiciens. Pourtant il faudra faire avec.

c) Puis ce non-déterminisme est intéressant : il modélise toute la dimension humaine

Et, dans un sens, bien au contraire, cette absence de déterminisme du système est intéressant car il constitue un modèle de toute la dimension humaine.

En tant que chercheur dans une UFR de sciences humaines, je m'en réjouis.

Avec l'agent situé nous disposons d'un modèle, directement issu des mathématiques, extrêmement épuré, et qui modélise déjà un comportement non déterministe : il ne faut pas s'en offusquer mais s'en réjouir. Nous disposons d'un outil qu'il faut prolonger vers les sciences humaine. Le but de notre thèse est bien de modéliser la causalité, les phénomènes de communication et les phénomènes cognitifs.

En tant qu'être humain, je m'en réjouis aussi.

Qui a dit : C'est quand l'ordinateur commencera à faire des erreurs qu'il deviendra intéressant.

Ch. 22 Aspect théorique : formaliser les Agents réactifs tropiques

1) Rappel - construction progressive : de l'automate fini vers l'agent CDI

1.A) Nous avons déjà traité ce tableau 1 : de l'automate fini (AF) à l'agent situé

Dans le cadre de ce tableau synthétique 1 (de la colonne 1 à la colonne 5), nous disséquons la transition qui part de l'automate fini et va jusqu'à l'agent situé. Afin d'obtenir un outil pédagogique simple, nous vidons le modèle de tout le superflu, puis nous y introduisons juste le minimum nécessaire afin de construire le plus simple des agents situés : l'agent situé minimal (ASM).

À chaque intervalle élémentaire de temps, il passe à la case suivante. Il tourne dans un manège dont la structure est très régulière. Mais l'activation de son actionneur est particulière.

En fait, il est seul dans un monde régulier, donc il n'a pas besoin de capteur/senseur pour scruter le monde extérieur afin de décider son action : son effecteur est actionné, activé perpétuellement. En conclusion, à chaque cycle, l'action est exécutée inconditionnellement : l'ASM n'est pas tout à fait un être causal.

type catégorie	***1) Sys.For. type 3***	***2) Sys.For. type 4***	***3) Machine séquenceuse***	***4) Cellule végétative***	***5) agent situé mini***	***6) Pluralité***
Nom commun	Automate fini	Séquenceur	Machine séquenceuse	Cellule végétative	agent situé minimal	Agent situé (AS)
Exemples :			jaquemart trotteuse	clignotant à deux états	cueilleur:thon requin	2DcueilleSurUnRéseau
Nature du cadre	mathématiques abstraites	mathématiques simplifié	physique, concret, moteur énergie ; solipsiste	vie artificielle solipsiste monde 0D : 1 seul lieu	agent sur un support constant	as en// simplifie support as non déterministe
quelle transition pour arriver ici?	départ depuis ici : l'automate fini	simplification de l'automate fini	*abstrait→concret, réel* se matérialise, s'incarne	introduit la vie, solipsiste autonome interne	situé sur un support, déplacement	mettre des agents actifs ou passifs en parallèle
Nature du procès effectué	règle de réécriture circuit de logique séquentielle	règle de réécriture circuit de logique séquentielle	moniste : mécanisme dualiste : prog. informatique ou système formel	solipsiste = f(interne) perçoit-décide-agit états, horloge internes	décision_cnst action = f (constante) états horloge extern	percept-décision-action = f(environnement) états & horloge externes

1.B) Tab 2, le parallélisme, de l'agent situé(A6S[35]) jusqu'à l'agent BDI rationnel

a) Étape 2, introduire le parallélisme : perception et décision pour piloter l'effecteur

Dans le chapitre précédent colonne 6 du tableau 1, nous avons introduit le parallélisme dans notre modélisation, ce qui induit la causalité, la décision réactive de l'action. Cependant nous avons seulement obtenu une multitude inorganisée d'agents, mais pas de groupe structuré.

Au moyen de l'introduction du parallélisme, nous plaçons du mobilier sur les supports où évolue l'agent situé. Ce sont des objets, des matières et même d'autres agents. Pour interagir avec eux, il utilise ses capteurs et devient un agent causal : nous avons construit un agent réactif.

Maintenant, en colonne 7 du tableau synthétique 2, nous formalisons la description du domaine où sont situés les agents. Nous simplifions la représentation des relations *agent-support* et nous les représentons au moyen de variables.

35 Agent Somatique Situé en Survie Sur un Support Simple

type	*6) Pluralité : 156*	*7) formaliser l'AS*	*8) Exemples*	*9) de l'AT à l'AP*	*10)A Intensions*	*11)A Croyances*	*12) Agt. CDI*
nom	agent situé sur un support	idem	Ag. Tropique(AT)	Ag.Pulsionnel (AP)	Agents intention	Agt. Croyances	Agent CDI
exemple	2DCueilleSurUnRéseau	idem	2DOpportunistes	1DChevron	2DGroupCoop	Scouter	
Nature du cadre	indirectement orienté vers un but. Simplifie la représentation du support	modéliser l'agent situé sur un plan mathématique	Aspect tropique de l'agent situé tropique	Agent pulsionnel	Agents intention puis agent intentionnel	Agnt croyances pour simuler et anticiper	Agent CDI
transi° pr arriver ici	mettre des agents actifs ou passifs en parallèle	Généralise l'état avec variables	Utiliser l'agent de Braitenberg	Forçage somatique de mémoire	Forçage raisonné de mémoire	Apprentissage	croyances désirs intention
Nature du procès effectué	percept-décision-action = f(environnement) états & horloge externes agent tropiques indirects	Herbrand simplifie les règles/procès avec des relations agent-support avec variables	implémentation de l'aspect tropique (avec Braitenberg)	gérer plusieurs comportements	interprétation de programmes	mémorise des croyances simule et anticipe	CDI + une couche de méta = rationnel

1.C) Présentation mathématique du parallélisme

Pour décrire des agents réactifs, commençons par utiliser la méthode formelle des mathématiques :

Les agents A_n et le support S permettent de construire le monde M.

Le monde est un quadruplet M = {S, Σ, Q, q_0} ; avec N agents $\Sigma = \{A_0, A_1 \ldots A_{N-1}\}$ et $A_n = (\varphi_n, \psi_n)$
S : Le support de cases, il est défini en extension
φ : Le physique, le soma de l'élément de mobilier : l'agent, ou la matière posée sur la case.
ψ : La psyché, ses règles. Soit $q_i, q_j \in Q$, $\psi : q_i \rightarrow act°_i$ et $(q_i, act°_i) \rightarrow q_j$, avec $q_j = RéponseDeM(q_i, act°_i)$;
Q : L'état de M, les relations entre ses composantes : *sur(A_n, $case_l$) ;* $\forall i \in [0, 7]$ *précède($Cases_i$,$Cases_{i+1}$);*
q_0 : L'état du monde au départ : $q_0 \in Q$; q_0 = *sur (A_n, $case_l$).*

1.D) Le monde M est un quadruplet {S, Σ, Q, q_0} :

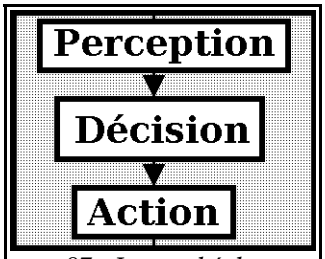
a) Le domaine S est le support de cases

S le domaine est un ensemble fini de cases. Il est déterministe, i.e. totalement connu : décrit en extension.

b) Les N agents $\Sigma = \{A_0, A_1 \ldots A_{N-1}\}$ avec $A_n = (\varphi_n, \psi_n)$

Sur le support S nous mettons un ensemble Σ de N agents :
$\Sigma = \{A_0, A_1 \ldots A_{N-1}\}$

φ : le soma, l'état physique de l'agent.
C'est l'agent réactif correspondant. De plus, si nous sommes dans le cadre d'un agent en survie, si l'état interne devient déficient alors l'agent meurt.



87 - La psyché de l'agent réactif

ψ : la psyché de l'agent.
C'est l'ensemble de ses règles comportementales. $\psi : Q \rightarrow Q$, la fonction de transition donnée par la psyché ψ.

Soient $q_i, q_j \in Q$; ψ est de la forme $\psi : q_i \rightarrow q_j$;
et plus précisément ψ est de la forme $\psi : q_i \rightarrow act°_i$; puis $(act°_i, q_i) \rightarrow q_j$; Avec :
- $q_i \rightarrow act°_i$; À l'état du monde perçu on associe une action.
- $(act°_i, q_i) \rightarrow q_j$; Cette action change l'état du monde.
- $q_j = RéponseDeM(q_i, act°_i)$; L'état du monde obtenu est la réponse du monde M à l'action.

c) L'état du monde q

C'est l'ensemble des relations entre les composantes du monde. Il était défini en compréhension. Nous analysons systématiquement les relations croisées entre les composants du monde (A_n et S_i).
- Les relations $A_n \leftrightarrow S$: la position des agents sur le support : sur*(A_n, $case_i$) ;*
- La relation $A_n \leftrightarrow A_m$: la position relative des N agents. Ils sont sur deux cases différentes.

- Les relation $S \leftrightarrow S$; i.e. les relations entre les cases : chaque case est rattachée à la suivante (modulo 8).

Si nécessaire, toutes les autres relations se déduisent de ces trois familles.

d) q_0 : l'état du monde au départ

$q_0 \in Q$; $\forall n \in [0, N-1]$: $q = sur(A_n, case_n)$; Tous les agents sont sur des cases du support.
$\forall i$ et $j \in [0, N-1]$ $i \neq j \Rightarrow (A_i \neq A_j \wedge Case_i \neq Case_j)$;Tous les agents sont sur des cases différentes.

2) Décrivons formellement les agents réactifs tropiques

2.A) Introduction : dans la simulation nous faisons apparaître la causalité

a) Causalité due à l'énergie, l'agent fait le geste de prendre la nourriture et la mange

Dans les dernières modélisations que maintenant nous construisons, l'énergie est distribuée de façon ponctuelle, par tas : certaines cases du support reçoivent un lot, un morceau, une pièce d'énergie. Pour en faciliter l'assimilation, l'agent est doté d'un corps somatique, où une poche conserve l'énergie en contact avec l'organe assimilateur afin d'en aider l'absorption.

Pour remplir cette poche l'être doit effectuer le geste requis au bon moment. On débouche ici sur un comportement dédié, et causal.

b) La causalité due aux obstacles répartis sur le support

L'introduction du parallélisme permet de disposer un obstacle sur certaines des cases. Pour des raisons vitales, l'être doit savoir gérer les obstacles, i.e. les éviter : il doit se comporter de façon causale.

2.B) Le fonctionnement réactif causal *perception décision action*

a) L'intervalle élémentaire de temps de l'agent réactif

Le système formel (SF) est rythmé par un intervalle élémentaire de temps relatif à l'application d'une règle de réécriture : à chaque instant élémentaire le SF applique une règle et son état change. Le monde de l'agent situé est, lui aussi, rythmé par ce même intervalle élémentaire de temps.

L'agent réactif, fabriqué en groupant en parallèle un agent situé et des agents passifs, unifie aussi leur base de temps : seul l'agent agit donc sa base de temps devient celle de l'agent réactif. De cette façon, l'agent réagit instantanément à chaque intervalle élémentaire de temps

b) La règle de réécriture de l'agent réactif redevient causale

Tout d'abord faisons un petit rappel :

Nous sommes partis avec la règle causale de l'AF :	$\delta : (Q \times \Sigma) \rightarrow Q$: fonction de transition donnée par la table à 2 entrées.
Nous la simplifions avec le séquenceur solipsiste :	$\delta : Q \rightarrow Q$: la fonction de transition donnée par la table à 1 entrée.
Puis vient ψ : la psyché de l'agent situé.	$\psi : q_i \rightarrow act°_i$ avec $q_i, q_j \in Q$, et $(q_i, act°_i) \rightarrow q_j$, réponse de M à l'action.

Détaillons $\psi : q_i \rightarrow act°_i$ le format des règles de réécriture de l'agent situé.

Chaque règle de réécriture associe à l'état du monde q_i une action $act°_i$.

Pour l'agent situé, qui est seul sur le support, l'état du monde est de la forme : $sur(A, case_i)$

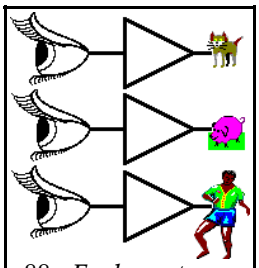
Pour l'agent réactif, qui est accompagné sur le support, la forme de l'état du monde devient : $sur(A_n, case_i)$

Détaillons $(q_i, act°_i) \rightarrow q_j$, réponse de M à l'action.

Ensuite, pour l'agent situé, comme pour l'agent réactif, cet état du monde q_i, est transformé par l'action $act°_i$ et nous obtenons q_j, nouvel état du monde M correspondant à cette action.

c) Globalement, le fonctionnement *perception décision action*

Tout d'abord nous décrivons globalement le fonctionnement *perception décision action* de l'agent réactif.



88 - Ex de capteurs sub-symboliques

Les capteurs ou senseurs pour percevoir l'état du monde.

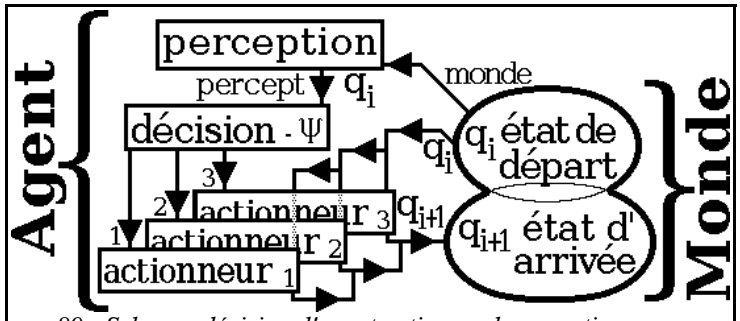
Pour percevoir *q* l'état du monde l'agent réactif doit disposer d'un capteur, encore appelé senseur. Puisque nous nous plaçons selon une approche connexionniste, sub-symbolique les capteurs sont dédiés à une tâche, ici à une nature de case. Nous aurons donc un capteur qui répond à *nourriture* et un autre à *obstacle*.

La psyché, fonction de décision $\psi : q_i \rightarrow act°_i$.

À chaque état Q du monde elle associe l'action à effectuer dans la situation donnée. Cette fonction peut être implémentée par une table, mais souvent la combinatoire devient grande et elle s'écrit plus facilement au moyen d'une règle.

L'actionneur, l'effecteur pour changer l'ordonnancement du monde.

L'actionneur, l'effecteur est un système physique qui change l'ordonnancement du monde quand il est activé par la commande $act°_i$.



89 - Selon sa décision, l'agent active un de ses actionneurs

Puisque nous nous plaçons selon une approche connexionniste, sub-symbolique les actionneurs sont dédiés à une tâche, ici à une action. Par exemple, nous aurons donc un actionneur pour manger, un pour avancer, un pour tourner à droite...

$(q_i, act°_i) \rightarrow q_j$: réponse du monde à l'action faite.

Puisque nous sommes inscrit dans un monde concret, parfois les actions peuvent être entravées. Donc nous devons aussi modéliser la réponse du monde q_j aux actions $act°_i$ des actionneurs en fonction de la situation q_i.

2.C) Précisons, définissons, décrivons la <u>prise de décision</u> qui pilote le procès

Maintenant nous nous attachons à expliciter la <u>prise de décision</u> qui pilote le procès. Elle correspond à cette ligne de la modélisation d'un agent réactif : soit $q_i, q_j \in Q$, $\psi : q_i \rightarrow act°_i$ et $(q_i, act°_i) \rightarrow q_j$, avec $q_j = \text{RéponseDeM}(q_i, act°_i)$;

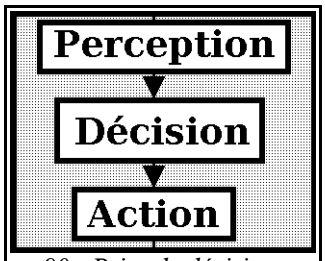
a) Pour décider, l'agent appréhende le monde localement avec des déictiques

Introduction d'une perception locale.

La perception de l'agent est limitée dans sa direction et sa portée. À dessein, il regarde droit devant lui et son horizon se limite aux objets les plus proches. Finalement, nous convenons même qu'il ne perçoit que la case devant lui.

L'agent utilise les déictiques pour caractériser les objets qu'il traite.

De plus l'agent présente un comportement déictique. Pour caractériser cet objet qui est devant lui, il emploie une variable *x*, qui pour lui signifie *l'objet, qui est devant moi*. En fait, cette terminologie implique une double déictique : d'abord une déictique classique *moi* pour parler de lui-même. Ensuite la déictique *l'objet qui est devant* caractérise l'objet du traitement en considérant qu'il se tient dans le lieu focus, i.e. la case qui est juste devant l'agent.



90 - Prise de décision de l'agent situé

La prise de décision locale de l'agent situé est non déterministe.

À cause de l'effet d'horizon, la prise de décision locale de l'agent situé est non déterministe. En effet, quand il prend une décision, il n'a pas connaissance de toutes les données du problème. Dans ce cas, elle ne peut pas être optimale.

b) La décision des commandes est basée sur la méthode de Herbrand

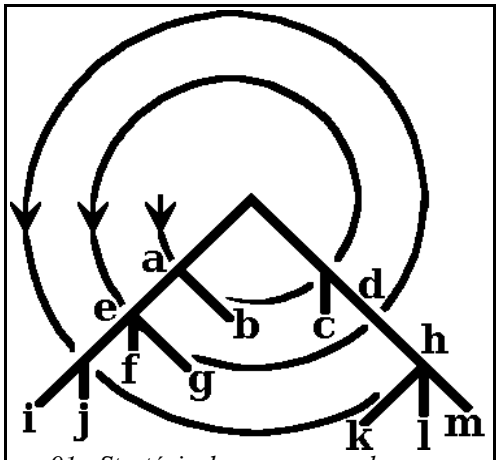
Ici : une décision sommaire.

Dans les premières modélisation d'agent réactif nous le plaçons sur un support 1D avec de la nourriture et/ou des obstacles. Il possède peu de capteurs. Sa prise de décision est vraiment sommaire.

Pour continuer notre exposé, nous devons donc anticiper. Nous prendrons donc des exemples plus complexes parmi les cas qui seront traités plus loin dans notre classification.

Une stratégie de parcours d'arbre en largeur d'abord.

Dans nos futures simulations, quand les agents auront plusieurs capteurs, leur module de décision devra piloter plusieurs actionneurs. En appliquant la méthode que Herbrand utilise pour construire ses univers, nous interconnecterons les capteurs aux actionneurs. Cette méthode empirique correspond à une stratégie en largeur d'abord, et s'avère bien pratique en évolution artificielle.



91 - Stratégie de parcours en largeur d'abord

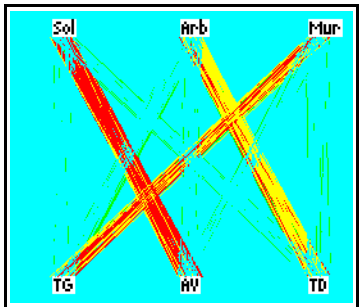
Exemple : La simulation *ApprendRegle.*

Pour illustrer notre propos, regardons sommairement la simulation *ApprendRegle* tirée de notre progression d'exemples. Dans cet exemple, illustré ci-dessous, pour connecter trois capteurs à trois actionneurs, nous utilisons un parcours en largeur d'abord, et nous nous arrêtons au premier niveau : à celui des connexions directes. Ainsi nous obtenons un résultat simple mais qui fonctionne déjà.

Le procès générique basée sur cette méthode est non déterministe.

Mais l'inconvénient de cette méthode est qu'elle est empirique, i.e. non déterministe car elle est basée sur un décodage partiel des conditions.

Par exemple, dans la simulation *ApprendRegle,* on remarque aisément que beaucoup de cas possibles ne sont pas décodés. Signalons par exemple ¬ la négation d'un capteur ou encore les fonctions logiques binaires ∧ et ∨sur ces capteurs.



92 - 9 neurones pour connecter 3 capteurs à 3 actionneurs

c) Conclusion à propos de la prise de décision de l'action

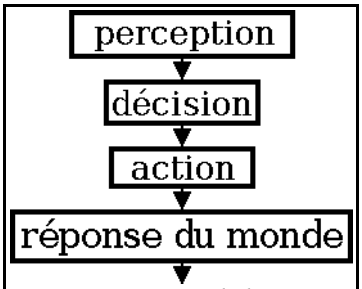
Finalement, nous pouvons conclure que la prise de décision de l'agent situé que nous utilisons est doublement non déterministe car elle subit l'effet d'horizon et relève d'une stratégie empirique.

2.D) Préciser la réponse du monde dans le synoptique de l'agent

a) La réponse du monde aux actions de l'agent

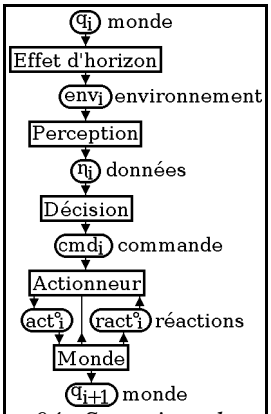
Dans le paragraphe précédent, nous avons montré combien l' agent situé est non déterministe. Quand le module de décision commande une action aux effecteurs, leur procès n'est pas déterministe : le déroulement de l'action peut très bien être perturbé plus ou moins gravement.

Puisque dans ce chapitre, notre but est de représenter son fonctionnement général, maintenant, dans notre synoptique, nous devons prendre en compte un module supplémentaire : la réponse du monde.



93 Synoptique de l'agent réactif intégrant la réponse du monde

b) Synoptique détaillé d'une action lors d'une transition d'état



94 - Synoptique du traitement d'une action

Perception.

$q_i \in Q$	L'état du monde avant l'action.
env_i = EffetDHorizon(q_i) ;	L'effet d'horizon transforme le monde en un environnement
η_i= Perception(env_i)	La fonction de perception scrute l'environnement immédiat.
η_i= *lieu(A,case$_i$)*	On obtient la perception : *tel agent est sur la case$_i$.*

En entrée de l'agent réactif se trouve q_i l'état du monde avant l'action. Mais le module de perception est affecté par l'effet d'horizon et ne perçoit que env_i : l'environnement de l'agent ; il ne restitue donc que η_i.

Décision.

cmd_i = Décision(η_i)	Vu η_i la fonction de décision décide la commande cmd_i.
cmd_i = Avance	cmd_i. correspond à l'ordre d'avancer.

La fonction de décision, qui reçoit le paramètre η_i retourne cmd_i, la commande à exécuter, qui correspond à l'action *Avance*.

Action de l'effecteur et réaction du support.

$act°_i$= Actionneur(q_i, cmd_i, $ract°_i$)	L'actionneur effectue son action vu cmd_i et le monde.
$act°_i$ = ActionAvance	L'action effectuée est *sur (A, case$_i$)* → *sur (A, Case$_{i+1}$)*.
q_{i+1} = *sur (A, case$_{i+1}$)*...	Dans l'état du monde, la position de l'agent a changé.

Sur la case *case$_i$*, l'actionneur, qui, avec le support *S* est dans l'état q_i, reçoit la commande *cmd$_i$*, et la réaction *ract°$_i$*, effectue l'action physique *ActionAvance*. Finalement l'agent se retrouve sur la case suivante.

3) Le problème épistémologique du procès et de l'action

3.A) Introduire, poser le problème : procès, but, action d'un agent situé réactif

Le procès effectué par un être devient une action seulement s'il est téléologique.

Nous avons vu que l'agent situé effectue un procès sur le monde au moyen de ses actionneurs, il change son ordre : il fait varier sa relation à son support, ou change la relation qu'un autre objet entretient avec le monde. Selon des critères épistémologiques, nous ne pouvons appeler cette opération une action que si le procès qu'elle effectue est finaliste, i.e. que s'il est sous-tendu par un but.

Or l'agent réactif, par définition, n'est pas explicitement téléologique[36], il n'entretient pas de but puisqu'en son sein, aucune mémorisation de cette sorte n'est effectivement implémentée. Cependant, nous pouvons employer le terme d'*action* pour décrire ses actes, à condition que cet être agisse implicitement pour un but, i.e. si nous pensons que le procès qu'il effectue est sous-tendu par une finalité implicite, indirecte.

Dans les systèmes d'IA, nous pouvons trouver différents niveaux de finalité.

Le procès effectué par un agent peut être plus ou moins téléologique. Tout d'abord, quand il est consiste en une séquence d'actions élaborée par un agent qui planifie son comportement, assurément il l'est.

Mais ce but peut apparaître indirectement dans le cadre de la vie artificielle (VA) où les agents sont en position de survie ou de compétition sur un support.

Finalement, ce but peut apparaître très indirectement dans le design de simples agents réactifs juste parce que leur concepteur sous-tend leur organisation vers un objectif ou un simple souci d'efficacité. Par exemple si un être se tourne vers la nourriture et la mange, il peut très bien présenter la structure d'un simple agent réactif, mais dans ce cas, nous considérons sans l'ombre d'un doute, que son organisation est sous-tendue par le désir de manger.

Conclusion : souvent les procès des simulations d'IA peuvent être vues comme des actions.

Ainsi, dans certaines conditions, cet être non-intentionnel possède déjà un sens, une sémantique et nous pouvons considérer son procès comme une action.

En conclusion, quand nous travaillons avec des agents réactifs conçus de façon cohérente, nous pouvons dire qu'avec leurs actionneurs, ils effectuent des actions.

Voyons cela plus en détail !

3.B) Un agent réactif est indirectement finaliste

Il est indirectement finaliste car il est conçu pour générer indirectement un comportement finaliste : son concepteur étudie sa structure et sa programmation pour qu'en réagissant dans le monde, il génère indirectement un comportement plus ou moins finaliste.

a) Un agent réactif fonctionnel est conçu de façon finaliste

Tout d'abord, nous pouvons imaginer différentes sortes d'agents finalistes. Leur conception est effectuée au préalable par un designer qui les façonne, et y imprime sa marque. Il met indirectement ses intentions dans la structure et la programmation de cet être qui, ensuite, est sous-tendu pour le reste de sa vie vers de but divers : le chaud, le froid, le groupe, l'eau, son parent, une direction donnée, un éclairage particulier...

b) Un agent, jeté dans le monde, en situation de survie

Dans ce cas, l'agent est finaliste. Il est orienté vers un but particulier : la recherche d'énergie. Il est jeté dans le monde, en situation de survie. Sa conception doit résoudre une contrainte introduite par la vie artificielle, elle doit lui fournir les moyens de trouver de la nourriture pour survivre et se reproduire.

c) Générer indirectement un comportement macroscopique

L'agent peut générer indirectement un comportement macroscopique pour lequel il n'est pas programmé.

Initialement, l'agent réactif est spontané, il n'est pas finaliste.

Il est programmé pour avoir un comportement instantané, il ne réagit qu'en fonction de l'instant présent. Donc, par nature, il n'est guère capable de générer des séquences longues qui se dérouleraient sur plusieurs instants.

En effet, en situation sur un support, quand il cherche sa nourriture et évite les obstacles, il agit de façon réactive : chacune de ses actions élémentaires est calculée instantanément, elle est élaborée par une structure *perception décision action.* Elle ne s'inscrit pas directement dans une

36 Téléologique : finaliste ; orienté, sous-tendu vers un but.

finalité à long terme, elle ne fait pas partie d'une séquence d'actions directement planifiées afin d'atteindre un but.

En général l'action d'un agent réactif est causale.

L'agent réactif est causal, il agit selon le processus *perception décision action*. Donc en général ses actions sont finalistes à profondeur unaire. Ex : quand il voit de la nourriture devant lui, il la saisit.

Cependant, attachons-nous à trouver des contre-exemples à cette généralité.

L'agent réactif peut être finaliste sur deux instants.

Certaines de ses actions sont finalistes à profondeur 2-aire : quand il voit de la nourriture sur le côté, dans un premier temps il se tourne vers elle, et ensuite, il la saisit.

L'agent réactif peut être finaliste sur une profondeur n-aire.

Le comportement d'un agent réactif peut très bien être finaliste. En cherchant, on peut encore imaginer des situations où il s'inscrit dans une finalité à grande profondeur. On peut construire des simulations où il agit et indirectement génère un comportement téléologique sur une profondeur n-aire.

Par exemple, il peut forcer des marqueurs dans le paysage, à la manière d'un enfant qui fait un nœud à son mouchoir pour ne pas oublier d'effectuer, beaucoup plus tard, une action demandée par ses parents.

Tout repose sur la situation locale, i.e. sur l'aménagement du mobilier du monde. À ce propos, le lecteur peut se reporter aux chapitres suivants sur les agents réactifs pulsionnels et analyser les deux alternateurs (*1DAlternateur* et *Chevrons*). Il y trouvera bien deux exemples d'agents réactifs qui, dans une organisation du mobilier du monde donnée, effectuent des cycles d'ampleur *n*.

d) Comportement macroscopique d'un agent

À un niveau macroscopique, nous observons chez l'agent un comportement pour lequel il n'est pas directement programmé.

La vie artificielle (VA) introduit une contrainte, un cadre externe autour du système.

La VA introduit dans le système, deux critères à propos de l'énergie. D'abord elle induit une *sortie sanction* par le bas : l'agent qui dépérit finit par mourir ; ensuite, une *sortie valorisation* par le haut : l'agent bien portant se reproduit.

Ces critères se positionnent dans un cadre externe, autour du système, en position finale. Dans ce cadre, l'agent est autonome, il décide de son action et l'exécute. Ces critères n'interviennent pas directement dans la génération de ses actes immédiats, ils viennent seulement mesurer leur efficacité, et parfois même les taxer abruptement en les frappant de mort ou en leur permettant de donner la vie.

Comment la chose est-elle implémentée ? Dans le quotidien l'être vit sa vie, décide de ses actions. Mais chacune consomme de l'énergie et ampute la réserve correspondante de sa quote-part. Finalement les critères d'énergie apparaissent à la fin, à l'extérieur de ce processus. Il répondent à la question : y a-t-il encore de l'énergie pour mener l'action au bout.

Les critères à appliquer sont naturels, ils sont donc faciles à simuler. Pour réaliser les modélisations correspondant à cette VA, le concepteur doit implémenter des opérations additives, et des comparaisons relatives à un seuil. Ce sont des critères naturels que l'évolution des espèces a implémenté dans des structures physiques. Par conséquent, ils sont simples et aisés à transcoder dans un langage de programmation : ils sont donc faciles à simuler.

À l'intérieur de ce cadre, l'agent n'est pas finaliste, mais il est indirectement finaliste.

L'agent n'est pas directement finaliste, mais il est conçut pour générer un comportement téléologique : donc il est indirectement finaliste. En effet, les actions de l'agent ne sont pas directement planifiées pour atteindre un but, mais son comportement émerge de sa conception qui elle est finaliste. il est conçu pour se débrouiller à trouver à manger, afin de survivre et se reproduire.

Le concepteur de l'agent est finaliste. Il reçoit un cadre contraint par les critères de la VA à propos de l'énergie. À l'intérieur de ce cadre macroscopique, il conçoit l'agent pour qu'il génère

un comportement qui soit plus ou moins adapté à son but, à trouver de la nourriture. Le concepteur de l'agent élabore sa structure et sa programmation pour qu'il vive sa vie : à l'agent de gérer le quotidien, de décider ses actions : il doit se débrouiller en situation de survie sur le support.

L'agent n'est pas finaliste, il est indirectement finaliste.

e) Le comportement d'un agent réactif est indirectement finaliste

En conclusion, le comportement d'un agent réactif peut être indirectement finaliste, s'il a été bien programmé.

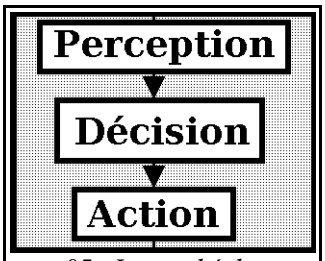
3.C) Conclusion : le procès devient action

Dans un paragraphe précédent nous avons déjà décrit la notion de procès : à chaque étape du cycle, le pion change sa position sur le support. Puisque dans notre simulation nous avons introduit la *vie artificielle*, nous pouvons ajouter qu'il est sous-tendu par une finalité. Ainsi, de *procès* il devient *action*. Donc, maintenant dans notre présentation, nous pouvons employer un verbe plus connoté. Au lieu de dire que l'agent change de lieu, nous disons qu'il bouge.

4) Bilan - L'agent réactif tropique (agent tropique)

4.A) Nous disposons d'un agent réactif, i.e. au comportement spontané

Ainsi nous avons construit un agent réactif, qui fonctionne selon le mode *perception décision action* : il vit dans l'instant *hic et nunc*[37].



95 - La psyché de l'agent réactif

4.B) Définition, présentation de l'agent réactif

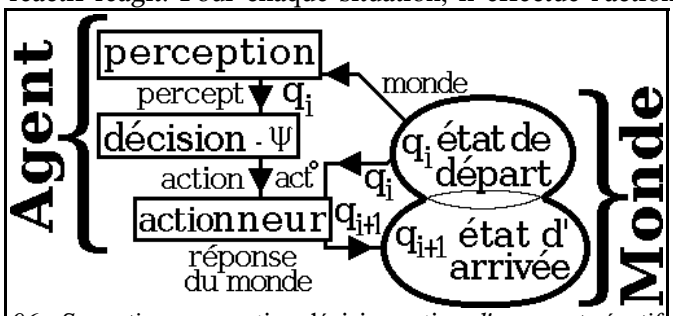
L'entité réactive agit selon le schéma *stimulus → réponse.*

Utilisons les critères de la psychologie expérimentale, pour décrire, définir le comportement de l'être réactif. Il est behavioriste (comportementaliste) : il agit selon le schéma *stimulus → réponse* :

- Au moyen de son étage de perception, il recueille le stimulus en provenance du monde.
- En fonction de ce percept, de ce stimulus, il cherche dans sa table *stimulus → réponse*, quelle réponse appliquer.
- Il applique cette réponse en effectuant l'action correspondante.

Présentation d'un agent réactif situé.

En situation dans le monde, l'agent réactif réagit. Pour chaque situation, il effectue l'action correspondante : à chaque unité élémentaire de temps, il perçoit son environnement, et en fonction de ce stimulus il répond, il agit : au moyen d'un réflexe, il effectue le geste adapté, en activant un de ses actionneurs.



96 - Synoptique perception décision action *d'un agent réactif*

L'agent situé possède des capteurs afin de percevoir l'environnement où il se tient. Au vu des stimulations qu'il perçoit (q_i), il analyse la situation et prend une décision (act°) consistant à changer l'ordonnancement du monde.

37 La locution latine *Hic et nunc* se traduit par *ici et maintenant*.

Pour cela, il effectue un procès : soit il change sa propre relation à son support, soit il change la relation qui lie une autre pièce du mobilier à ce support ; par exemple, il déplace un obstacle.

Finalement nous obtenons le synoptique, ci-contre, à droite, qui représente le synoptique d'un agent réagissant dans le monde.

L'agent réactif est inconséquent : il vit dans l'immédiat.

L'agent réactif vit dans l'instant, pour chaque situation perçue, il effectue une action réflexe, qui n'est pas inscrite dans la durée : il ne se retourne pas sur son passé, ni ne se projette dans le futur, il ne déroule pas de séquence thématique sur plusieurs étapes : il vit seulement au jour le jour.

Comportement = Σ (stimulus → réponse).

Ainsi, selon un point de vue macroscopique, le comportement d'un être réactif demeure pointilliste : il est constitué de l'ensemble de chaque réponse instantanée fournie à chaque stimulus perçu.

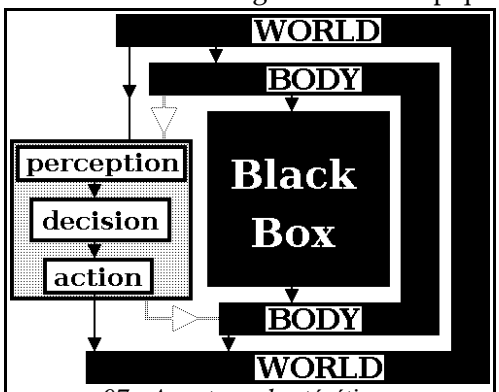
4.C) L'agent réactif tropique s'inscrit dans l'instant, i.e. hors du temps

a) Rappel préalable

Dans notre modélisation, tout agent situé possède au moins un état somatique interne : le niveau de sa réserve d'énergie.

b) L'agent réactif tropique s'inscrit hors du temps

Le dessin de gauche présente le synoptique de fonctionnement d'un agent réactif tropique. Les flèches en pointillés marquent des liaisons qui peuvent exister chez d'autres types d'agents, mais sont absentes chez lui.



97 - Agent non hystérétique

Par définition, l'agent réactif tropique est hors du temps : il est non hystérétique[38], il ne possède pas de mémoire, et n'utilise pas ses états internes pour inscrire son action dans le temps.

En conséquence, l'agent tropique ne possède pas de règle pour forcer un état interne. Ainsi, sur le dessin, la liaison *Actionneur → Corps* a disparu.

De même, un agent tropique ne possède pas de règle analysant un de ses états interne car il ne dispose pas de capteur pour les tester. Ainsi, sur le dessin, la liaison *Corps → Capteur* a disparu.

c) Précisons le terme tropique selon son étymologie

Selon son étymologie ce terme signifie : *qui possède la propriété de se tourner en direction quelque chose*. Ceci nous amène à distinguer deux sortes d'agents tropiques.

La plupart des agents tropiques ne sont qu'indirectement tropiques car nous pouvons seulement dire qu' *ils sont indirectement finalistes, i.e. sous-tendus vers un but*[39] . En effet dans ce cas là, leur but est abstrait : il n'est pas directement géographique.

Cependant, dans le chapitre suivant *ExempleDAgentSituésRéactifsTropiques* avec les agents de Valentino Braitenberg [Bra 1984], nous traiterons de vrais agents réactifs tropiques, qui s'orientent vraiment en direction d'un lieu géographique.

d) Contre-exemple : l'agent hystérétique (avec mémorisation)

Afin d'aider le lecteur à cerner le profil de cet agent réactif tropique, présentons toute une famille de contre-exemples : les agents hystérétiques qui ne peuvent vraiment pas être aussi

38 Du grec husterein, être en retard. Historiquement, en informatique, le cycle d'hystérésis du tore de ferrite a servi à construire des mémoires binaires élémentaires.

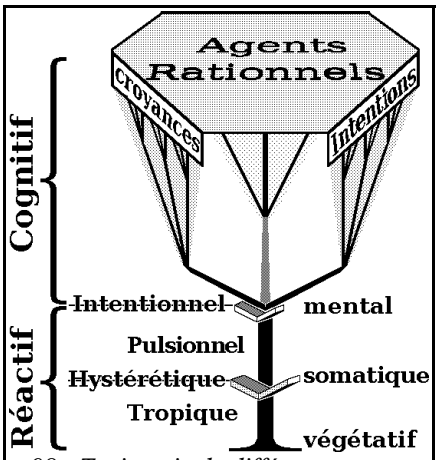
39 Voir dans ce chapitre, ci-avant au paragraphe 5 : *Le problème épistémologique du procès et de l'action.*

spontanés et inconséquents car ils sont inscrits dans le temps : l'agent Intention, l'agent Croyances et l'agent rationnel.

Un agent finaliste ou encore *agent Intentions* forme des projets, des intentions qu'il mémorise : il planifie ses actions en fonction des buts qu'il s'est donné et les mémorise sous forme d' intentions, de planning. Parce qu'il est doté de mémoire et de capacités de traitement de l' information, il ne peut pas être réactif.

Un *agent Croyances* possède et entretient dans son fort intérieur, une représentation de son monde. Doté d'une mémoire de grande capacité, il ne peut pas être réactif car ainsi sa prise de décision est fonction de deux paramètres : sa perception et son état interne (sa mémoire).

Un *agent rationnel* ne peut vraiment pas être réactif car il possède à la fois des croyances, désirs et intentions. Il forme des buts et utilise sa connaissance à propos du monde pour planifier ses actions.



98 - Taxinomie de différents agents

5) Bilan et conclusion :

5.A) L'agent situé minimal : base d'une reconstruction

La base de départ : l'agent situé minimal.

Dans le cadre de cette partie théorique, nous avons disséqué la transition qui part de l'AF et va jusqu'à l'agent situé. Afin d'obtenir un outil pédagogique simple, nous avons vidé notre modèle de tout le superflu. Puis nous y avons introduit juste le minimum nécessaire pour construire le plus simple des êtres. Dans ce cadre, nous avons tellement contraint le monde, qu'il ne reste plus à l'agent qu'une action possible : *avancer*. Ainsi, nous obtenons le plus simple des êtres : l'agent situé minimal (ASM) qui tourne en rond sur la piste d'un manège.

Mais l'ASM est non causal.

Regardons ce premier agent situé que nous avons décrit. À chaque intervalle élémentaire de temps, il passe à la case suivante. Il tourne dans un manège dont la structure est très régulière. Mais l'activation de son actionneur est encore peu élaborée. En fait, cet agent n'a pas besoin de capteur/senseur pour scruter le monde extérieur afin de décider comment agir : son effecteur est activé perpétuellement. En conclusion, à chaque cycle, l'action est exécutée inconditionnellement : il n'est pas causal.

5.B) L'agent réactif tropique

Ensuite, au moyen du parallélisme des agents, sur le support de lieux de l'agent situé nous avons placé du mobilier. Il consiste en divers objets, des matières et même d'autres agents. Pour entrer en interaction avec eux, il utilise ses capteurs. Il est devenu cet agent réactif tropique que nous venons d'étudier en détail.

5.C) À terme : construire en couches progressives vers les agents rationnels

Partant de l'ASM, muni de cette simple base, maintenant nous disposons d'un outil formel pour nous lancer dans une démarche constructive déterministe : en libérant progressivement les contraintes imposées à ce cadre initial, nous allons modifier la structure interne du monde et de ses êtres.

Au moyen de ce mécano, nous allons construire notre agent en couches concentriques. À terme, ceci donnera naissance à différentes forme d'activités chez les agents. Ainsi dans ce

document, au fil du temps, nous allons fabriquer progressivement, des agents tropiques, pulsionnels, finalistes.

Nous finirons cette présentation en évoquant des agents rationnels, plus puissants et plus complexes car construits à partir des agent CDI, munis de Croyances, il sont aussi finalistes (Désirs) et possèdent des Intentions.

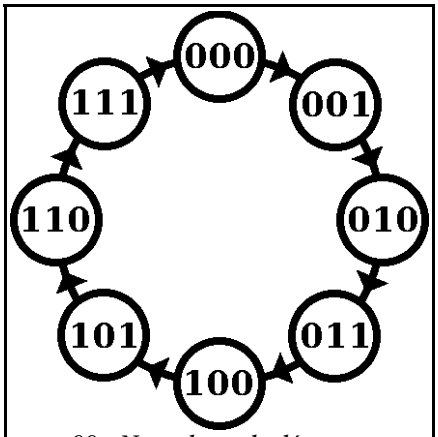
Ch. 23 Volet pratique : aspects tropiques de l'agent réactif tropique

Colonne 8 : exemples d'agents situés réactifs tropiques

type	*6) Pluralité : 156*	*7) formaliser l'AS*	*8) Exemples*	*9) de l'AT à l'AP*	*10)A Intensions*	*11)A Croyances*	*12) Agt. CDI*
nom	agent situé sur un support	idem	Ag. Tropique(AT)	Ag.Pulsionnel (AP)	Agents intention	Agt. Croyances	Agent CDI
exemple	2DCueilleSurUnRéseau	idem	2DOpportunistes	1DChevron	2DGroupCoop	Scouter	
Nature du cadre	indirectement orienté vers un but. Simplifie la représentation du support	modéliser l'agent situé sur un plan mathématique	Aspect tropique de l'agent situé tropique	Agent pulsionnel	Agents intention puis agent intentionnel	Agnt croyances pour simuler et anticiper	Agent CDI
transi° pr arriver ici	mettre des agents actifs ou passifs en parallèle	Généralise l'état avec variables	Utiliser l'agent de Braitenberg	Forçage somatique de mémoire	Forçage raisonné de mémoire	Apprentissage	croyances désirs intention
Nature du procès effectué	percept-décision-action = f(environnement) états & horloge externes agent tropiques indirects	Herbrand simplifie les règles/procès avec des relations agent-support avec variables	implémentation de l'aspect tropique (avec Braitenberg)	gérer plusieurs comportements	interprétation de programmes	mémorise des croyances simule et anticipe	CDI + une couche de méta = rationnel

1) Rappel : notre base de départ

À l'issue de notre travail précédent *Passer du système formel à l'agent situé* nous disposons en colonne 6, d'un agent situé qui tourne sur manège.



99 - Notre base de départ : le cycle du manège

a) Nous avons introduit le monde :

Le monde est constitué de l'agent figuré par un pion ; et de son support, un espace 1D qui se reboucle sur lui-même ; ex : le cycle du manège.

b) Nous avons introduit la notion de procès

L'agent est seul sur le support, il fait varier sa relation au support. À ce stade, cette opération ne constitue pas encore une action mais s'appelle plutôt un procès. Nous la décrivons seulement comme un changement de la position du pion sur le support.

c) Nous avons introduit l'énergie

Le pion est posé sur un support où l'énergie est distribuée de façon diffuse. Convenons que dans notre modélisation, l'agent l'assimile proportionnellement à sa vitesse de déplacement : pour assimiler l'oxygène dissout dans l'eau, à la manière de certains poissons (requins, thons, etc.), il doit toujours être en mouvement.

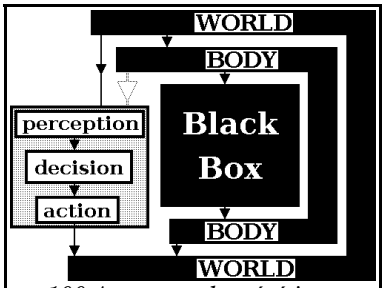
d) Nous avons introduit la vie artificielle

À chaque fin de cycle, l'être se reproduit.

La survie de l'être, dépend de sa reproduction. Elle dépend donc indirectement de son déplacement au sein du milieu.

- Exemple 1 : ce fonctionnement peut être illustré par le cycle graine-cellule : C→G→C→G→C : la cellule disparaît pour donner naissance à une cellule qui grandit ou point de donner une graine.

- Exemple 2 : ce fonctionnement peut être illustré par le cycle classique de l'agent qui tourne dans un manège octogonal : 0→1→2→3→4→5→6→7→0. Quand l'agent



100 Agent non hystérétique

arrive à sa taille maximale 7, il disparaît et donne naissance à l'état 0, ce qui correspond à une graine qui recommence le cycle et grandit progressivement.

Conclusion : le procès devient action.

Dans un paragraphe précédent nous avons déjà décrit la notion de procès : à chaque étape du cycle, le pion change sa position sur le support. Puis, après avoir introduit la vie artificielle dans notre simulation, nous pouvons maintenant ajouter qu'il est sous-tendu par une finalité. Donc, de *procès* il devient *action*. Ainsi, dans notre présentation, nous pouvons employer un verbe plus connoté. Au lieu de dire que l'agent change de lieu, nous disons qu'il bouge.

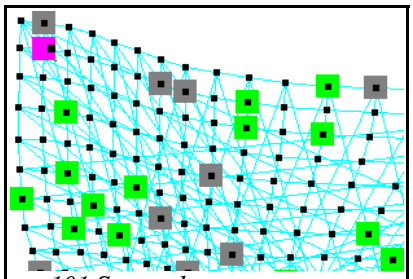
e) Bilan : comme base de départ, nous disposons d'un agent réactif

Ainsi nous avons construit un agent réactif. Il ne force pas d'état interne, et il ne possède pas d'état externe car il ne dispose pas de capteur pour les tester : il vit dans l'instant *hic et nunc*[40].

2) L'agent réactif tropique sur un espace 1D introduit une *causalité discrète*

À partir de notre base initiale, construisons maintenant un agent réactif tropique.

2.A) Introduction du parallélisme



101 Sur quelques cases, nous ajoutons de la nourriture et des obstacles

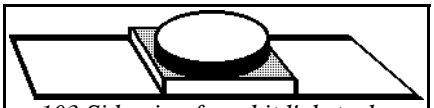
Dans le monde, nous introduisons son mobilier. Ce sont des objets que nous posons sur les lieux du support. Leur taille est celle d'une case, et ils sont constitués de différentes matières. Ainsi, dans notre modélisation, à la suite de l'agent, nous introduirons progressivement des cases socles, des obstacles, de l'eau, de la nourriture... Chacun de ces objets est donc typé, mais au sein d'une même classe, les instances sont équivalentes, donc indifférentiables. Exemple ci-contre, à droite : dans un monde posé sur une structure de graphe, nous remarquons, en haut à gauche, l'agent magenta. Sur chaque lieu, un carré noir marque la case. Les lieux contenant de la nourriture sont en vert, et les obstacles en gris.

Donc, dans notre modèle, nous introduisons de nouveaux objets...

2.B) Les objets que nous introduisons sur les lieux bloquent l'agent

L'agent se déplace sur les cases du graphe. Mais, le long de ce parcours, il rencontre une matière que nous appelons *Obstacle*.

102 L'agent (le pion) vient butter contre l'obstacle carré gris.



103 Si le pion franchit l'obstacle, alors il ne touche plus au support.

L'agent ne peut pas le franchir en le chevauchant car ceci entraînerait la relation *Sur(Agent, Obstacle)*, mais cette relation est impossible dans notre modélisation, car l'agent ne serait plus en relation avec le support. Or nous avons introduit le mobilier du monde en précisant que les éléments du mobilier sont en relation avec le support. Donc, dans ce cadre, ce mouvement est impossible.

L'agent ne peut pas contourner l'obstacle en passant à côté de lui sur la même case. En effet, nous avons introduit le mobilier du monde en disant : nous ne plaçons qu'un meuble par case, et qu'une case pour soutenir un meuble. Donc ce mouvement de contournement est impossible car dans ce cas, à la fois l'agent et l'obstacle devraient être en relation avec le support, or cette configuration n'est pas prévue dans la modélisation.

40 La locution latine *Hic et nunc* se traduit par *Ici et maintenant*.

Finalement nous constatons que l'obstacle bloque le passage de l'agent.

2.C) Rappel : limite de profondeur des interactions

Les interactions relevant du domaine audio (ou parfois visuel), sont à portée infinie : elles ne sont pas limitées en profondeur.

Par contre, toutes les autres interactions sont limitées par un horizon proche, qui lui-même est souvent limité à une profondeur unaire. Ainsi, dans un espace 1D, nous déduisons que l'obstacle bloque l'agent au moment où il veut passer.

2.D) Introduction du sens : l'agent repart dans l'autre sens

Quand l'agent rencontre un obstacle, la seule adaptation qu'il lui reste est de repartir dans l'autre sens. À ce stade, nous devons définir un sens à un agent, afin qu'il puisse faire demi-tour. Nous savons que quand l'agent est tourné dans le sens des abscisses croissantes son déplacement peut se décrire par cette règle :

$Sur\ (Agent,\ Case_n) \rightarrow Sur\ (Agent,\ Case_{n+1})$.

Une fois qu'il a fait demi-tour, le résultat obtenu est le dual de la règle précédente et peut se décrire au moyen de celle-ci :

$Sur\ (Agent,\ Case_n) \rightarrow Sur\ (Agent,\ Case_{n-1})$.

Note : quand un agent est représenté par un carré, son sens est donné par un petit carré noir qui pointe dans la direction de son déplacement.

2.E) Introduction de la finalité

Nous avons écrit que le système est sous tendu par les lois de l'énergie introduites au début de ce chapitre. Il doit absorber/collecter plus d'énergie qu'il n'en dépense, et comme il capte son énergie en se déplaçant, quand il rencontre un obstacle, il doit immédiatement repartir dans l'autre sens.

2.F) Intervention du concepteur : introduction de règles de causalité

Quand nous introduisons des obstacles dans le système, le concepteur doit écrire les règles de conduite en conséquence. Alors il rajoute une règle qui dit que si l'agent rencontre un obstacle, il fait demi-tour.

2.G) Conclusion : du procès à l'action

Procès, but et action d'un agent réactif.

Nous avons vu que l'agent effectue un procès au moyen de ses actionneurs, il change l'ordre du monde : il fait varier sa relation à son support, ou change la relation qu'un autre objet entretient avec le monde. Selon des critères épistémologiques, nous ne pouvons appeler cette opération une action que si le procès qu'elle effectue est finaliste, i.e. sous-tendu par un but.

Or, par définition, l'agent réactif n'est pas explicitement téléologique[41], il n'entretient pas de but puisqu'en son sein, aucune mémorisation de cette sorte n'est effectivement implémentée. Cependant, nous pouvons employer le terme d'*action* pour décrire ses actes, à condition que cet être agisse implicitement pour un but, i.e. si nous pensons que le procès qu'il effectue est sous-tendu par une finalité implicite, indirecte.

Par exemple, ce but apparaît indirectement dans le cadre de la vie artificielle (VA) où les agents sont en position de survie ou de compétition sur un support. Ce but peut apparaître très indirectement dans le design de simples agents réactifs juste parce que leur concepteur sous-tend leur organisation vers un objectif ou un simple souci d'efficacité. Par exemple si un être se

41 Téléologique : finaliste ; orienté, sous-tendu vers un but.

tourne vers la nourriture et la mange, il peut très bien présenter la structure d'un simple agent réactif, mais dans ce cas, nous considérons sans l'ombre d'un doute, que son organisation est sous-tendue par le désir de manger. Dans ce cas, un tel système est sous-tendu par une finalité qui est induite indirectement par le *designer*. Alors les procès que l'agent effectue deviennent des actions.

Ainsi, dans certaines conditions, cet être non-intentionnel possède déjà un sens, une sémantique et nous pouvons considérer son procès comme une action.

En conclusion, quand nous travaillons avec des agents réactifs conçus de façon cohérente, nous pouvons considérer qu'au moyen de leurs actionneurs, ils effectuent des actions.

3) Agents réactifs tropiques dans un espace 2D : *une démarche causale riche*

3.A) La finalité demeure mais le concepteur introduit des règles plus complexes

Le fait de passer en 2D n'introduit rien de nouveau, sauf une grande souplesse et diversité des fonctionnements.

Le système reste sous-tendu par les lois de l'énergie que nous avons introduites au début de ce chapitre. La finalité demeure la même, mais le concepteur introduit des règles plus complexes. Il doit écrire les règles de conduite qui expliquent ce que l'agent doit faire en présence d'obstacles dans le système. Voici un exemple de règles conditionnelles qui permettent de guider un agent au milieu d'un monde 2D parsemé d'obstacles.

Si RienDevant	alors avance()	break;.
Si ObstacleDevant et pas ObstacleAGauche	alors tourneàgauche().	break;
Si ObstacleDevant et ObstacleAGauche et pas ObstacleADroite	alors tourneàdroite().	break;
Si ObstacleDevant et ObstacleAGauche et ObstacleADroite	alors faisdemitour().	break;

3.B) Introduction d'une grande richesse

Introduction du parallélisme.

Avec cette introduction de la gestion du parallélisme dans le modèle, maintenant, dans la modélisation, nous pouvons modéliser des obstacles

Les objets que nous introduisons sur les lieux bloquent l'agent.

Sur une grille, selon leur quantité, les obstacles bloquent plus ou moins le passage :

- Introduits en trop grand nombre ils immobilisent ou enferment l'agent.
- Un nombre moyen d'obstacles constitue un chemin, et même une frontière.
- Quand il ne reste que quelques obstacles, l'agent peut les contourner. Ainsi, dans ce monde en 2 dimensions, ce mobilier introduit sur la grille, induit une autre attitude, un autre comportement : les agents peuvent circuler au milieu des obstacles en les contournant.

3.C) Introduction d'états externes :

Généralisation de la notion de sens : l'agent tourne dans plusieurs directions.

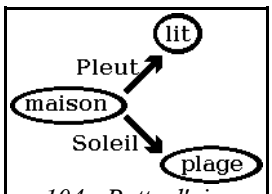
Tout d'abord, dans un monde 1D nous avons introduit un sens (croissant ou décroissant) et le verbe correspondant : *faire demi-tour*. Ensuite, dans un monde 2D nous introduisons un sens qui prend plusieurs directions.

Nous travaillons avec des agents tropiques, qui par nature, ne possèdent pas d'état interne, cependant nous réussissons à en obtenir en utilisant un autre moyen, un biais différent. Par exemple, quand nous prenons le sens d'un agent, nous attribuons à ce dernier un prédicat, qui est défini par la relation *agent*↔*monde*, et nous lui faisons jouer le rôle d'un état. Exemple : Cap(est) - (le cap de l'agent est à l'Est). Mais en fait, cet état est codé extérieurement : il constitue un état externe.

Introduction d'états externes chez les agents tropiques.

En dotant les agents tropiques de capteurs, nous introduisons dans le système différents états externes : la direction de l'agent, sa position par rapport à un repère.

Ainsi munis de ces repères, les agents tropiques, qui sont dotés de causalité, peuvent les tester. Ceci permet de générer, à un niveau local, un parcours en forme de patte d'oie, i.e. une bifurcation, ou encore une croisée des chemins. Et plus globalement, nous obtenons un parcours de graphe.



104 - Patte d'oie, bifurcation.

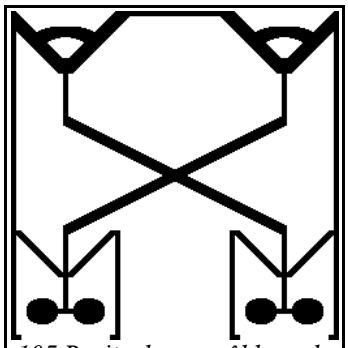
Introduction des marqueurs drapeaux externes.

Le concepteur peut encore introduire une seconde forme d'état : des drapeaux qu'il pose dans le monde. Ensuite, il les force et les lit, à la manière de la machine de Turing. Ainsi il met en place une programmation séquentielle. Donc il obtient des agents qui parcourent des trajets et effectuent un travail à plus ou moins long terme. Ce faisant, il procède un peu comme une machine de Turing qui déroule un programme long et conséquent, et augmente ainsi sa puissance de traitement.

3.D) Introduire l'influence à distance

Les agents tropiques, sont tellement simples que leur registre de fonctionnement est limité. De façon générale, en programmation multi-agents, il s'avère difficile de leur faire exécuter un traitement séquentiel d'envergure. De même, chez des agents situés, nous peinons à donner de l'ampleur aux trajectoires qu'ils parcourent.

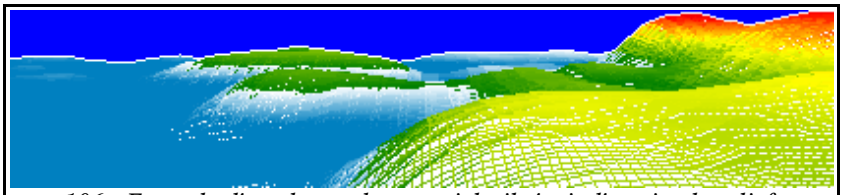
Cependant, dans un premier temps, nous avons montré comment utiliser des états ou des marqueurs externes pour générer avec les agents tropiques, des parcours complexes. Maintenant, à la lumière des travaux de Braitenberg, étudions comment l'utilisation de champs scalaires ou vectoriels peut, elle aussi, générer des cheminements intéressants au moyen d'agents situés munis de deux capteurs et de deux actionneurs de mouvements.


105 Braitenberg : câblage de l'agent tropique amour

L'illustration ci-contre présente un véhicule de Braitenberg orienté vers le haut. Sur sa face avant, nous remarquons deux capteurs/senseurs de vision, symbolisés par deux yeux. Sur sa face arrière, sont dessinés deux actionneurs/effecteurs de mouvements, symbolisés par deux hélices vues de dessus. Au milieu de l'agent, une circuiterie interconnecte les capteurs aux effecteurs et détermine le comportement de l'agent en réponse aux sollicitations extérieures.

a) Cas 1 : la source du champ est passive ou statique

La source qui émet ou est analysée est passive ou statique : elle demeure constante et n'est pas émise par quelque agent qui veut communiquer.


106 - Exemple d'un champ de potentiel : il s'agit d'un simple relief.

Dans ce paragraphe, nous traiterons deux exemples : celui d'un champ de potentiel, et le cas d'une source de lumière.

Dans son approche, Braitenberg distingue 3 démarches différentes, nous développerons principalement les deux premières *amour* et *haine*, et plus rarement la dernière : l'attitude *tempérée* ou *médiane*.

Attitude *amour* :

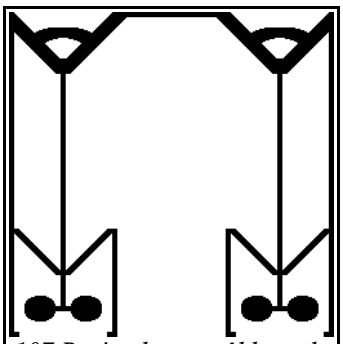
Dans l'agent tropique *amour* le câblage *censeur→actionneur* est croisé : le capteur gauche active l'actionneur droit et réciproquement, le senseur droit active l'effecteur gauche. Un tel

véhicule, mis en présence d'une source de lumière, se tourne et progresse vers elle. En effet, s'il advient que sa direction s'éloigne de la source, et par exemple s'infléchisse vers la gauche, son capteur droit reçoit plus de lumière, et, en réaction, augmente l'activation du moteur gauche, ce qui corrige la trajectoire. Évidemment, nous pouvons effectuer le raisonnement dual en permutant les termes *gauche* et *droite* dans l'énoncé.

Le véhicule *amour*, placé dans un champ de potentiel, i.e. sur une colline, est capable de monter un gradient[42] au prix d'une adaptation. Son câblage demeure croisé, mais l'activation des actionneurs se fait par des capteurs de position : plus la roue est levée plus elle active son effecteur, plus elle est baissée plus cette activation diminue.

Attitude *haine* (ou encore *crainte*) :

Dans l'agent tropique *haine* le câblage *censeur-actionneur* est direct : le capteur droit active l'actionneur droit et symétriquement du côté gauche. Un tel véhicule, mis en présence d'une source de lumière, s'en détourne et la fuit. En effet, s'il advient que sa direction s'éloigne de l'axe de fuite de la source, et par exemple s'infléchisse vers la gauche, son capteur gauche reçoit plus de lumière, et, en réaction, augmente l'activation du moteur gauche, ce qui corrige la trajectoire. Évidemment, nous pouvons effectuer le raisonnement dual en permutant les termes *gauche* et *droite* dans l'énoncé.

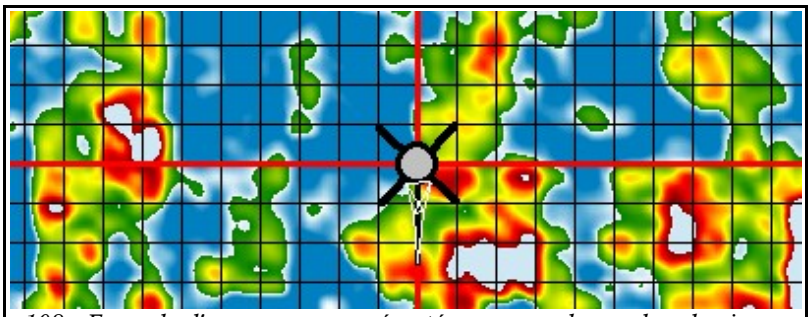


107 Braitenberg : câblage de l'agent tropique haine

Le véhicule *haine*, placé dans un champ de potentiel, i.e. sur une colline, descend les gradients au prix d'une adaptation. Son câblage demeure droit, mais l'activation des actionneurs se fait par des capteurs de position : plus une roue est levée plus elle active son effecteur, plus elle est baissée moins elle l'alimente.

Attitude tempérée, médiane.

Dans une démarche de vie artificielle, l'action des agents est sous-tendue par la recherche de nourriture. Alors le modélisateur cherche à simuler les migrations saisonnières des chasseurs-cueilleurs, qui parcourent le domaine à la recherche de ressources épisodiques et périssables.



108 - Exemple d'un paysage représenté au moyen de courbes de niveau

Dans un champ de potentiel, i.e. dans un monde 2D en relief, le parcours à potentiel constant (suivre les courbes de niveau), induit chez les agents des trajectoires amples. Pour ce type de déplacement, à la manière des dahus, les agents sont de deux natures : dextrogyres[43] ou lévogyres[44].

Le câblage de ces véhicules est différentiel mais demeure simple. Prenons le cas d'un agent dextrogyre se déplaçant autour d'un monticule. Il tourne donc dans le sens horaire. Si le niveau des roues avant dépasse celui des roues arrières, il faut accélérer la roue droite et donc ralentir la roue gauche. Le raisonnement dual est valable si le véhicule plonge. De même, il est transposable pour l'agent lévogyre et pour un déplacement dans une cuvette. Il faut cependant préciser que, pour les véhicules tempérés comme pour les dahus, les changements de milieu ou de sens de rotation sont problématiques !

b) Cas 2 : un réseau de marques de phéromones (insectes sociaux)

Maintenant nous changeons de domaine et passons au cas où la source analysée est dynamique : elle évolue progressivement au fil du temps. L'exemple typique est celui du réseau

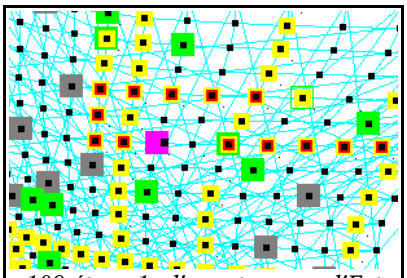
42 Monté de gradient : Hill climbing, soit en français grimpé de colline.
43 Qui tourne vers la droite (dextre = droite).
44 Qui tourne vers la gauche (left = gauche).

de phéromones des insectes sociaux : d'un côté, il s'évapore un peu à chaque instant, mais de l'autre, il se renforce en fonction des contributions des individus.

Ici encore, nous retrouvons les deux démarches *amour* et *haine*.

Attitude *amour* :

Les insectes remontent les champs de potentiel, i.e. la piste de phéromones pour retourner vers un point focus : le nid ou la source de nourriture.

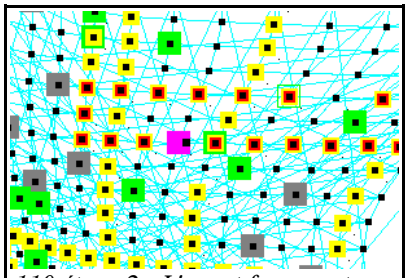


109 étape 1 : l'agent va vers l'Est

Attitude *haine* :

S'il s'agit d'une attitude de haine envers une phéromone agréable, l'insecte fuit la source. Il peut descendre un champ de potentiel, un gradient le long de la piste de phéromone pour fuir le point focus, le nid, afin d'explorer le domaine, et chercher une nouvelle source pérenne de nourriture.

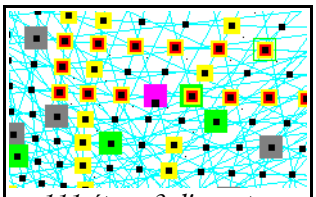
S'il s'agit d'une phéromone désagréable, par exemple, une phéromone de stress, l'insecte qui rencontre une montée soudaine de son gradient doit s'alerter, et détourner son chemin afin de fuir le danger.



110 étape 2 : L'agent face au stress

Étudions les trois étapes de l'exemple illustré ici :

- À l'étape 1, L'agent, de couleur magenta est au centre de l'image. Il porte un petit carré noir (à droite) qui indique sa direction : il est tourné vers l'Est. Devant lui, un petit rectangle noir figure un lieu vide ; donc il avance.
- À l'étape 2, il se trouve face à une case qui contient une hormone rouge, marqueuse de stress : donc il décide de tourner.
- À l'étape 3, il a tourné vers le Sud. Ainsi il s'est détourné de ce chemin marqué dangereux par une phéromone de stress.



111 étape 3. l'agent a tourné

c) Cas 3 - source active : le son

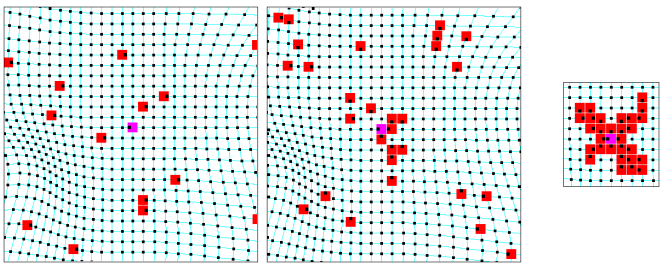
Maintenant nous passons à une source dynamique : la pression acoustique ou tout simplement le son. Il permet deux attitudes : d'abord l'écholocalisation, et ensuite la communication .

L'écholocalisation.

Elle permet le contact audio entre deux personnes, ou entre le groupe et son leader. Celui qui entend, écoute, peut localiser la position de l'émetteur. L'application typique de cette faculté est le rassemblement des membres du groupe autour du meneur (patriarche, matriarche).

La communication (l'échange de données).

Selon un protocole riche, la source qui est active et dynamique fluctue rapidement dans le temps. Ainsi sa modulation porte de l'information : Elle émet des données et sert à communiquer des informations ou des états d'âme, mais en fait nous ne pouvons la traiter ici car elle concerne les agents intentionnels.



112 Les agents rouges convergent vers le leader magenta

Remarque :

En écholocalisation au moyen du son, nous retrouvons les trois attitudes de Braitenberg : amour, haine et l'attitude médiane.

Attitude 1 de Braitenberg : *amour* :

Exemple 1 - Des agents opportunistes :

L'écholocalisation permet aux membres du groupe de se tourner vers la source de bruit, puis de converger vers elle. Regardons l'image ci-contre.

À gauche, les agents rouges d'un groupe sont répartis dans la nature : il sont dispersés autour de leur leader de couleur magenta.

Au milieu, ils se tournent vers la source audio et convergent vers elle.

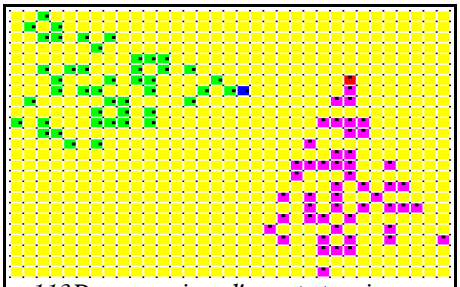
À droite, à la fin du processus, les agents ont convergé vers la source et se retrouvent maintenant agglutinés à elle.

Exemple 2 : deux essaims d'agents.

L'écholocalisation permet aux membres du groupe de se positionner par rapport au leader.

Dans l'image ci-contre nous observons deux essaims d'agents tropiques. Les agents verts suivent le leader bleu qui chemine vers l'Est. Ceux en magenta suivent le leader rouge qui progresse vers le Nord.

L'algorithme de positionnement des agents par rapport à leur leader reste le même que dans les exemples précédents, mais comme ce dernier est en mouvement, les agents sont derrière lui et restent positionnés dans un quadrant dont le sommet est tenu par le meneur. Ainsi nous obtenons cette forme particulière d'essaim.

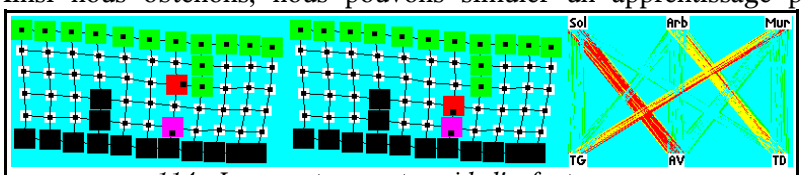


113Deux essaims d'agents tropiques. L'un progresse vers l'est, l'autre vers le nord

Exemple 3 : éducation par l'exemple.

L'écholocalisation permet à l'enfant rouge de se tourner vers son parent magenta, qui le guide le long d'un parcours. Ainsi nous obtenons, nous pouvons simuler un apprentissage par l'exemple du geste.

Dans la situation de gauche, l'enfant voit devant lui une case verte (un arbre). Pour suivre son parent qui l'appelle, il va tourner à droite et renforcer le neurone *arbre→tourneDroite*.



114 - Le parent magenta guide l'enfant rouge, qui apprend 3 règles dans son réseau de 9 neurones

Attitude 2 de Braitenberg : *haine* :

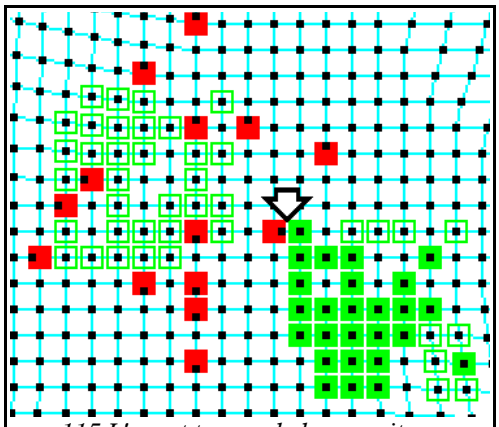
Exemple 1 :

C'est l'exemple classique de l'alerte. Il consiste à fuir le prédateur.

Exemple 2 : La diaspora (ci-contre)

Les agents rouges ont mangé toute l'herbe, alors ils passent en phase de recherche de nourriture, et se répartissent dans tout l'espace. Pour ce faire, ils sont bruyants et chacun s'éloigne des autres en fuyant toute source de bruit.

L'agent rouge marqué par la flèche noire à trouvé de la nourriture (verte). Il va appeler ses collègues qui vont rappliquer.



115 L'agent trouve de la nourriture

Attitude 3 de Braitenberg : *tempérée, médiane*

Exemple 1 : Dans ce chapitre, nous avons déjà évoqué ces parcours à potentiel constant, qui

consistent à suivre les courbes de niveau. Une application intéressante de cette méthode peut fournier une alternative aux trajets de recherche et de ralliement en ligne droite en paysage vallonné : partant de l'origine A pour chercher une ressource, au lieu de suivre la ligne droite, l'agent se déplace à potentiel constant, ce qui lui économise de l'énergie. S'il trouve sa ressource B ensuite il inverse la méthode pour retourner au point A.

Exemple 2 : Les enfants jouent en zone protégée, sous la protection des parents. Pour ne pas s'éloigner du groupe, ils se déplacent à potentiel constant, sans jamais descendre au-dessous d'un potentiel minimal de sécurité : ils restent à portée de voix.

Exemple 3 : À la limite d'un territoire (tessellation de Voronoï/Dirichlet) de jeune mâles devenus matures, viennent de se faire bannir du groupe. Leur seule solution : circuler aux confins des territoires des mâles dominants, se faufiler entre les marquages de propriété que les grands mâles ont laissés.

3.E) L'agent tropique : brique de base de l'agent pulsionnel

Le chapitre suivant traite de l'agent pulsionnel, c'est pourquoi nous précisons que l'agent tropique lui sert de brique de base. En effet, en montant en parallèle plusieurs agents tropiques nous pouvons fabriquer un agent pulsionnel. Cette considération est développée plus longuement dans le chapitre consacré aux agents réactifs pulsionnels.

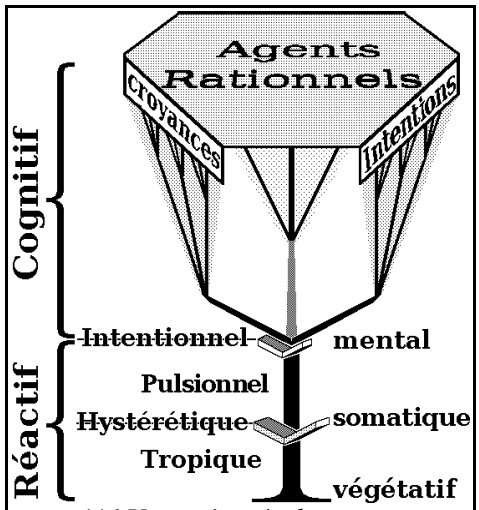
4) L'agent tropique, brique de base pour d'autres agents

L'agent tropique est simple, c'est pourquoi il sert de brique de base pour construire les autres agents :

D'abord, malgré sa simplicité, en marquant le mobilier du monde, il peut exécuter des traitements séquentiels conséquents.

Ensuite, il sert de base à l'agent pulsionnel, puis à l'agent intentionnel.

Finalement, quand nous créerons nos premiers systèmes capables de simuler, les agents CDI mis-en-oeuvre commenceront d'imaginer en utilisant des agent tropiques, à cause de la simplicité de ces derniers.



116 Une taxinomie des agents

5) Transition vers l'agent pulsionnel, puis vers l'agent intentionnel

Par définition l'agent pulsionnel possède un ou des états somatiques internes qui interviennent dans sa prise de décision. Par exemple nous pouvons évoquer les simulations où l'agent oscille entre la faim et soif ; ou entre chaud et froid. C'est pourquoi, dans la taxinomie des agents, il apparaît, au-dessus de la frontière des agent hystérétiques.

Nous qualifions ces états de *somatiques* car ils sont positionnés par les pulsions, i.e. par un traitement réflexe provenant du corps de l'être (sôma en grec) sur lequel la réflexion n'a pas de prise.

Par la suite, quand ils seront gérés par un traitement de l'information effectué volontairement par l'être, nous passerons aux agents intentionnels.

Ch. 24 De l'agent réactif tropique à l'agent réactif pulsionnel (AP)

Colonne 9 : De l'agent réactif tropique (AT) à l'agent réactif pulsionnel (AP)

type	*6) Pluralité : 156*	*7) formaliser l'AS*	*8) Exemples*	*9) de l'AT à l'AP*	*10)A Intensions*	*11)A Croyances*	*12) Agt. CDI*
nom	agent situé sur un support	idem	Ag. Tropique(AT)	Ag.Pulsionnel (AP)	Agents intention	Agt. Croyances	Agent CDI
exemple	2DCueilleSurUnRéseau	idem	2DOpportunistes	1DChevron	2DGroupCoop	Scouter	
Nature du cadre	indirectement orienté vers un but. Simplifie la représentation du support	modéliser l'agent situé sur un plan mathématique	Aspect tropique de l'agent situé tropique	Agent pulsionnel	Agents intention puis agent intentionnel	Agnt croyances pour simuler et anticiper	Agent CDI
transi° pr arriver ici	mettre des agents actifs ou passifs en parallèle	Généralise l'état avec variables	Utiliser l'agent de Braitenberg	Forçage somatique de mémoire	Forçage raisonné de mémoire	Apprentissage	croyances désirs intention
Nature du procès effectué	percept-décision-action = f(environnement) états & horloge externes agent tropiques indirects	Herbrand simplifie les règles/procès avec des relations agent-support avec variables	implémentation de l'aspect tropique (avec Braitenberg)	gérer plusieurs comportements	interprétation de programmes	mémorise des croyances simule et anticipe	CDI + une couche de méta = rationnel

1) Au départ : le forçage instinctif d'un état somatique interne

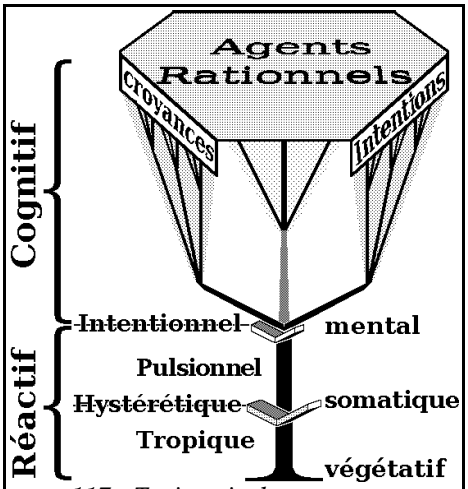
Nous construisons, couche après couche, la structure de nos agents, car elle détermine leur comportement. Après la cellule végétative, nous avons construit l'agent tropique (colonne 6 du tableau ci-dessus), qui ne possède pas d'état interne, mais qui peut gérer des états externes (les états du mobilier du monde en relation avec leur support).

1.A) Des états hystérétiques[45]

Avant de rentrer dans le vif du sujet, précisons la notion d'étape hystérétique. Elle marque l'introduction de la mémoire au sein de l'agent.

Le passage de l'agent réactif tropique à l'agent réactif pulsionnel (AP) est déterminé l'introduction d'états hystérétiques au sein de l'agent. Par définition, un agent hystérétique possède au moins un état interne, qui peut être forcé, et qui ensuite sert de mémoire, i.e. qui dure dans le temps, ce qui oriente le traitement de l'information effectué par l'agent.

Note : ce forçage de l'état interne de l'agent peut être raisonné ou instinctif, i.e. il peut être volontaire ou non.



117 - Taxinomie de nos agents

1.B) Aspect somatique : un forçage instinctif d'un état corporel

Ce passage de l'agent réactif tropique à l'agent réactif pulsionnel est doublement somatique. D'abord, les états internes forcés sont somatiques car ce sont des états du corps : faim, soif, chaud, froid, besoin de sel, de glucose... Et ensuite cette opération est aussi somatique, car chez l'être vivant, elle est effectuée par des réflexes instinctifs qui court-circuitent le cerveau car ils ne dépendent pas du néo-cortex ; au sein d'un agent, elle est effectuée par une machine interne qui est parfois cachée dans une boîte noire : l'utilisateur ne soulève pas toujours le capot de la mécanique. Car, quand un agent a faim, il le constate, mais il ne peut l'anticiper : en général, il ne voit pas sa jauge d'énergie !

45 Du grec *husterein*, être en retard. Se dit d'un système où on introduit une dimension mémoire.

1.C) Importance du niveau d'énergie dans ce scénario de vie artificielle

Dans nos précédentes simulations d'agent tropique, nous avons déjà remplacé la nourriture diffuse par de petits stocks ponctuels posés sur certaines cases. Cette nouvelle distribution spatiale a induit chez les agents un comportement causal car ils devaient faire le geste de manger pour absorber la nourriture.

Chez l'agent pulsionnel, à nouveau nous retrouvons la contrainte de la nourriture : elle est directement en relation avec le niveau d'énergie de l'être, qui correspond à un état interne à l'agent. Il peut donc être testé. Un niveau trop bas de la réserve d'énergie induit dans le corps, l'état interne de faim, qui constitue un état interne pulsionnel.

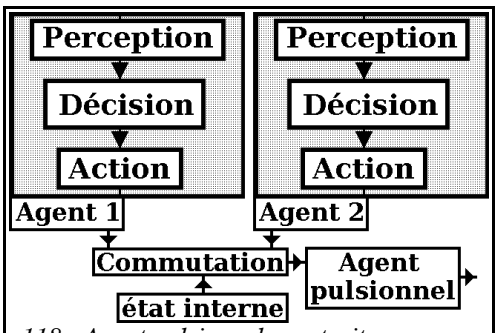
2) Quels sont les rôles et effets de ces états ?

2.A) Des états finalistes

Ces états internes sont interprétés comme des buts par la logique interne causale de l'agent qui, selon sa complexité, génère des stratégies ou des tactiques afin de les satisfaire. Ainsi les procès effectués par les agents sont sous-tendus par des buts, ils deviennent finalistes et peuvent être considérés comme des actions.

2.B) L'AP : N agents tropiques en parallèle, commutés par des états internes

Nous avons déjà introduite cette considération dans le chapitre à propos des agents tropiques et, dans ce chapitre, nous l'illustrerons par deux exemples : l'agent pulsionnel peut être vu comme plusieurs agents tropiques montés en parallèle, et commutés par un drapeau, i.e. un des états internes de l'agent.



118 - Agent pulsionnel construit au moyen de deux agents tropiques : son état interne est interprété par une fonction de commutation qui inhibe l'un et valide l'autre.

2.C) Agent au comportement séquentiel

La construction d'un agent pulsionnel est basée sur des agents tropiques dont le comportement est réactif. Chacun d'eux réagit dans l'instant au percept qu'il reçoit : il applique immédiatement la règle énonçant quelle action faire. Donc, à priori, ce mode de fonctionnement n'est pas inscrit dans la durée. Cependant, ces règles *conditions-action* sont causales, elles peuvent porter sur un ou des états internes. Ceci induit deux aspects : le fonctionnement de l'agent est rééclairé par les états internes, et il peut ainsi s'inscrire dans la durée.

Rééclairage du traitement en fonction de l'état interne.

Les règles *conditions-action* de l'agent sont causales. Leur tête de règle peut porter sur une combinaison logiques de conditions, parmi lesquelles se trouvent justement un ou des états internes. Alors on peut très bien imaginer un mode de fonctionnement distribué et contrasté : le traitement effectué par l'agent est rééclairé en fonction de son état interne. Toutes les règles qui traitent un domaine donné (faim soif chaud froid énergie position orientation...) testent le drapeau correspondant, et selon la réponse reçue, orientent le comportement de l'agent vers des fonctionnements opposés.

Inscription dans la durée des états internes.

Les règles des agents réactifs sont totalement rééclairées par des états internes qui peuvent très bien s'inscrire dans la durée. Dans la réalité quotidienne la plupart de ces variables sont souvent le résultat d'un processus d'accumulation, de sommation : la réserve d'énergie s'accumule en mangeant, pour se désaltérer le chameau boit longuement, pour se chauffer le

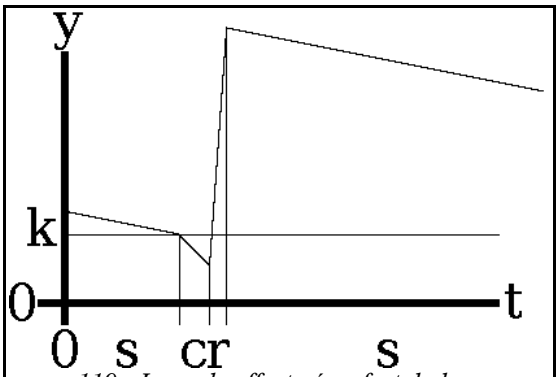
reptile demeure au soleil, et pour rallier un autre lieu l'Homo ergaster devait marcher des kilomètres...

En conséquence, les comportements des agents pulsionnels, quand ils sont commutés par des états internes basés sur l'accumulation, peuvent eux aussi s'inscrire dans la durée.

3) Application A : le cycle effort/réconfort

Tout d'abord, commençons par l'étude d'un exemple très simple d'agent pulsionnel : le cycle action/repos ou encore effort/réconfort. Il est basé sur la gestion d'un unique état : l'énergie interne.

À la fin de la sieste (période *s*), quand le niveau d'énergie interne descend au-dessous du seuil d'alerte *k*, l'agent doit chercher à manger. Il devient un cueilleur, il se lève pour aller gagner sa croûte, il part arpenter le monde afin de trouver sa pitance. Sa dépense d'énergie croît (période *c*).



119 - Le cycle effort-réconfort du boa

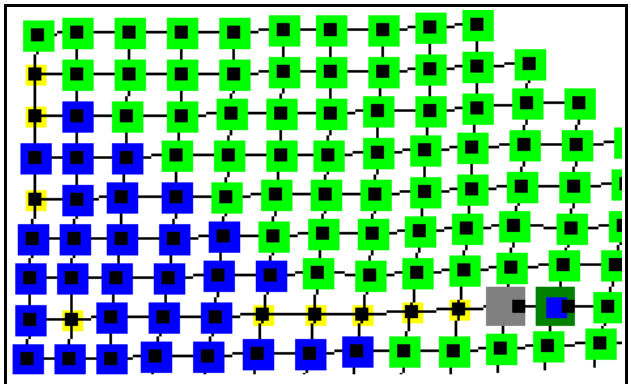
Quand il trouve de la nourriture, il mange jusqu'à être rassasié (période *r*), puis il reste sur place et fait la sieste (période *s*). Pendant ce temps, il ne fait que maintenir son potentiel vital. À chaque instant cette tâche consomme un peu de sa réserve d'énergie interne. Cette sieste dure jusqu'à ce qu'il ait faim à nouveau.

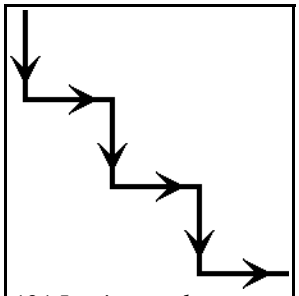
4) Application B : deux exemples d'alternateur

Ensuite continuons par l'étude de deux exemples binaires d'agent pulsionnel : l'alternateur. Ce dernier est basé sur la gestion de deux états internes : le niveau de la réserve d'eau et celui de l'énergie.

4.A) L'agent tropique : brique de base de l'agent pulsionnel

Nous avons considéré que l'agent tropique constitue la brique de base de l'agent pulsionnel, car ce dernier peut être vu comme N agents tropiques en montés en parallèle. Pour illustrer le cas N=2 nous pouvons prendre l'exemple typique de l'alternateur illustré dans deux implémentations différentes : *le monde en chevrons* et *l'alternateur eau/nourriture*.


120 - Parcours en zig-zag dans un monde en chevrons


121 Le zig-zag des agents

Agent simple dans un monde complexe en chevrons.

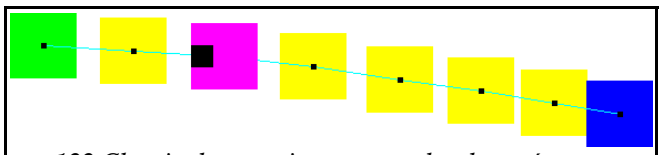
Le *monde en chevrons* est construit de façon un peu complexe, en répartissant les cases de nourriture (vert) et d'eau (bleu) autour de la diagonale nord-ouest/sud-est.

Dans l'illustration ci-dessus, initialement, l'agent est positionné en haut à gauche, et il part vers le bas pour boire les cases bleues. Ensuite il marche à l'est pour aller manger les cases vertes. Puis il repartira plein sud pour boire... Finalement, il décrit le parcours en zig-zag représenté par le schéma de gauche.

Ainsi ce monde complexe correspond à un agent pulsionnel très simple, fabriqué en assemblant deux fonctionnements tropiques élémentaires : *tourner à droite et marcher vers le sud*, et *tourner à gauche et marcher vers l'est.*

Cas d'un monde simple pour un agent complexe : l'alternateur.

Le monde de l'alternateur est construit de façon simple en reliant un stock de nourriture (vert) à une réserve d'eau (bleu) au moyen d'un chemin de cases jaunes. L'agent magenta oscille entre les deux réserves pour absorber alternativement de l'eau et de la nourriture.



122 Chemin de cases jaunes entre les deux réserves

Ainsi ce monde simple correspond à un agent pulsionnel plus complexe, fabriqué en assemblant deux fonctionnements tropiques simples : *faire demi-tour et marcher à l'est pour aller chercher de l'eau*, et *faire demi-tour et marcher à l'ouest vers la nourriture.*

5) Conclusion :

5.A) Quelle représentation du monde pour un agent pulsionnel ?

Si on généralise les exemples précédents, on tire des conclusions qu'on peut étendre et prolonger : un agent pulsionnel peut avoir des états internes, sur un ou quelques bits, mais pas au-delà. Il ne peut posséder de représentation explicite du monde, par exemple sous la forme attribut-valeur (eau-est ; nourriture-ouest). Par contre il peut disposer de représentation implicite du monde dans ses règles. Exemple : *si j'ai soif je vais à l'est.*

5.B) Perspective à terme : en couches progressives vers les agents intentionnels

Partis de la base simple des agents tropiques, nous disposions d'un outil formel pour nous lancer dans une démarche constructive déterministe : en libérant progressivement les contraintes imposées par le cadre initial, nous avons modifié la structure interne du monde et de ses êtres.

À terme, ceci a donné naissance à différentes formes d'activités chez les agents. Ainsi, au moyen de ce mécano nous construisons notre agent en couches concentriques. Au fil de notre progression, nous avons fabriqué des agents tropiques puis pulsionnels. Nous finirons cette partie théorique en évoquant des agents intentionnels, dont certains sont beaucoup plus puissants et plus complexes car finalistes et cognitifs.

5.C) Annoncer la double transition de l'intentionnalité

Dans les chapitres suivants nous quittons les agents réactifs pour ceux qui sont intentionnels. À la lumière de ce polycopié sur les agents réactifs, disons que la transition vers l'intentionnalité repose sur une double étape :

a) L'agent intentionnel

D'abord le forçage des états internes devient explicite et volontaire : l'agent décide en fonction de ses états internes et de sa perception de l'extérieur.

b) L'agent Croyances Désirs Intentions (Agent CDI)

Ces explications rapides sont données à titre indicatif, elles seront détaillées par la suite.

Agent C : l'agent Croyances.

Ensuite ces état internes deviennent plus nombreux en taille et en contenus. Ils ne se limitent plus à quelques bits et s'ils représentent encore un état neuronal interne, ils peuvent aussi

mémoriser l'état du monde et/ou les règles pour le transformer. L'agent tient ces informations pour vraies, ce sont ses croyances.

Agent D : l'agent Désir.

L'agent est finaliste/téléologique : il possède des désirs, i.e. des états internes constituant des buts qui ensuite sous-tendent ses actions.

Agent I : l'agent Intention.

L'agent planifie pour satisfaire ses buts ; ainsi il génère un planning de ses actions, qu'il stoke dans sa liste d'intentions : il a l'intention de les exécuter.

Ch. 25 De l'agent intentionnel à l'*agent Intentions*

Colonne 10 : De l'agent intentionnel à l'*agent Intentions*

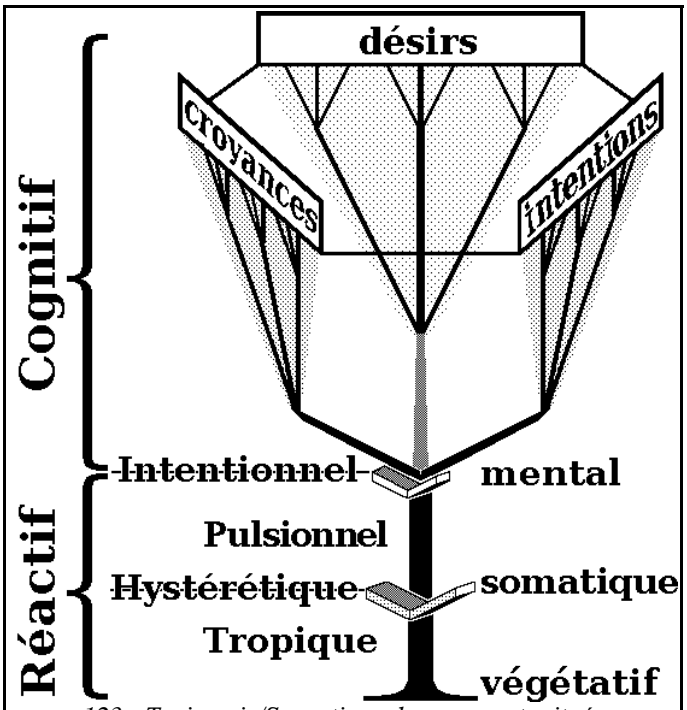
type	*6) Pluralité : 156*	*7) formaliser l'AS*	*8) Exemples*	*9) de l'AT à l'AP*	*10)A Intensions*	*11)A Croyances*	*12) Agt. CDI*
nom	agent situé sur un support	idem	Ag. Tropique(AT)	Ag.Pulsionnel (AP)	Agents intention	Agt. Croyances	Agent CDI
exemple	2DCueilleSurUnRéseau	idem	2DOpportunistes	1DChevron	2DGroupCoop	Scouter	
Nature du cadre	indirectement orienté vers un but. Simplifie la représentation du support	modéliser l'agent situé sur un plan mathématique	Aspect tropique de l'agent situé tropique	Agent pulsionnel	Agents intention puis agent intentionnel	Agnt croyances pour simuler et anticiper	Agent CDI
transi° pr arriver ici	mettre des agents actifs ou passifs en parallèle	Généralise l'état avec variables	Utiliser l'agent de Braitenberg	Forçage somatique de mémoire	Forçage raisonné de mémoire	Apprentissage	croyances désirs intention
Nature du procès effectué	percept-décision-action = f(environnement) états & horloge externes agent tropiques indirects	Herbrand simplifie les règles/procès avec des relations agent-support avec variables	implémentation de l'aspect tropique (avec Braitenberg)	gérer plusieurs comportements	interprétation de programmes	mémorise des croyances simule et anticipe	CDI + une couche de méta = rationnel

1) L'agent intentionnel

1.A) Introduction

Maintenant, dans notre étude, nous délaissons les agents réactifs pour passer à des agents supérieurs (intentionnels – colonne 10 du tableau ci-dessus).

Dans le dessin ci-contre, au-dessus des agents pulsionnels, une ligne verticale étiquetée *intentionnel* marque le passage depuis les agent réactifs, en direction des agents cognitifs : nous franchissons la frontière de l'intentionnalité. C'est pourquoi nous devrons donc maintenant préciser sa définition.



123 - Taxinomie/Synoptique de nos agents situés

1.B) Définir l'agent intentionnel

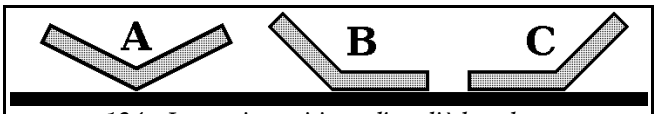
Dans notre travail sur l'intelligence artificielle distribuée (IAD) nous avons déjà défini l'intentionnalité : au sein d'un agent, un support (bit, symbole ou groupe de symboles) est intentionnel à trois conditions :

- Il doit être la représentation de quelque chose,
- Son enregistrement, sa mémorisation doit résulter de l'application d'un traitement de l' information effectué par cet agent.
- Il doit être interprété : la prise en compte de son contenu doit orienter le traitement et/ou le comportement de l'agent.

1.C) Quel type de connaissance est représenté au sein d'un *agent Intentions* ?

a) Introduction d'une convention graphique pour représenter le drapeau

Graphiquement, nous choisissons de représenter le drapeau du programme informatique, la mémoire élémentaire d'un bit, par un dièdre obtus. En effet, le dessin ci-contre montre comment une telle forme, un tel pliage, présente deux lieux de stabilité, un à droite *B* et un à gauche *C*. Ainsi, il symbolise bien la bascule RS de la logique séquentielle utilisée comme bistable, i.e. qui présente deux positions stables.



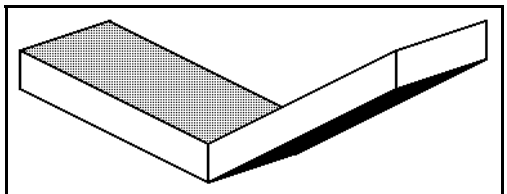
124 - Les trois positions d'un dièdre obtus

b) Drapeau mémorisé par un bit, mais étiqueté par un symbole

Mémorisé par un bit.

Dans notre classification, la mémoire de chacun des premiers agents intentionnels que nous décrivons, est uniquement basée sur un digit binaire, un bit.

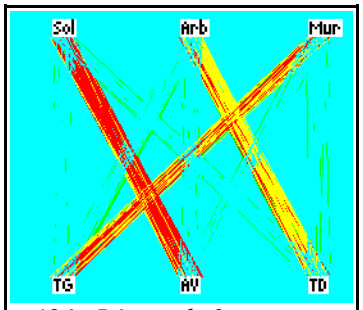
La mémorisation de cette information élémentaire est bénéfique à cet être, elle oriente son comportement, pour l'adapter à la situation en cours. Ainsi elle introduit un avantage, un progrès décisif.


125 - Dièdre obtus fonctionnant en bistable

Finalement, afin d'être classé dans la catégorie des agents intentionnels, un être doit disposer d'une faculté de représentation au moins aussi puissante,.

Étiquetée par un symbole.

Cependant, cette simple information est codée au sein de l'agent par un drapeau, mais ce dernier reçoit un nom. Certes l' information est mémorisée par un bit, mais elle est étiquetée par un symbole. Par exemple, dans le réseau de neurones ci-contre les trois entrées du haut correspondent à trois digits élémentaires (*sol arbre mur*), ainsi chacun d'eux est étiqueté par un symbole : *Sol*, *Arbre* ou *Mur*.



126 - Réseau de 9 neurones

Information sur l'état du monde ou règle d'action.

Dans les agents intentionnels les plus simples (de bas niveau), nous nous demandons quelle est la nature, le type de l'information mémorisée par un drapeau : est-ce un symbole décrivant l'état du monde, ou le programme à exécuter vu cet état ?

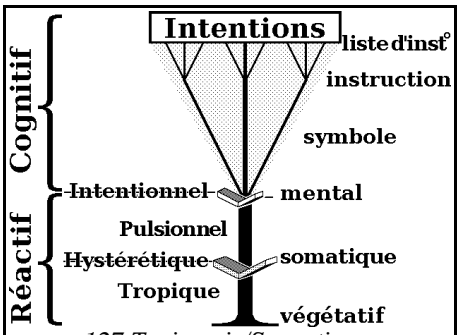
En fait, le type de cette information est ambivalent. Par exemple, d'un côté, ce bit peut être vu comme un symbole qui signifie *faim*, *stress*, *soif*, etc. et représente l'état du monde, de son support ou de son mobilier (l'agent a faim, un prédateur est sur le support, l'agent à soif). De l'autre, ce bit peut être interprété comme un programme qui commute le fonctionnement de l'agent entre plusieurs attitudes et démarches (chercher de la nourriture, fuir le danger, se procurer à boire).

Donc, en conclusion, chez les agents les plus simples, le digit élémentaire d'information peut être interprété comme la donnée d'informations à propos du monde, ou comme le programme comportemental à exécuter dans la situation donnée.

c) Une information intentionnelle mémorisée dans une structure de N symboles

Chez l'être le plus simple, la nature du drapeau intentionnel est difficilement catégorisable

L'information qu'ils portent et mémorisent peut être classée dans deux catégories : *nature de l'état interne* ou *savoir sur l'état du monde*. Pour illustrer cette ambiguïté regardons le drapeau *alerte.* Il peut être interprété comme un état interne qui détermine le comportement à tenir : *je dois passer dans un état de stress, d'attention*. Mais il peut encore être interprété comme une information à propos de l'état du monde : *dehors il y a danger, car rode un prédateur.*



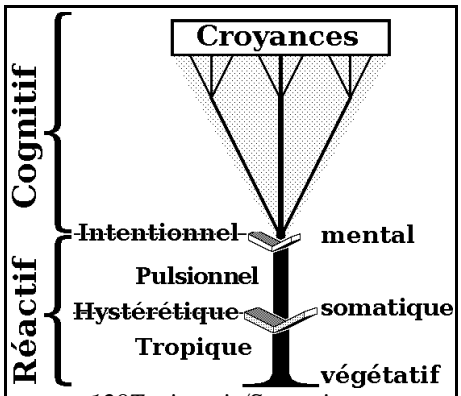
127 Taxinomie/Synoptique de l' agent Intentions

Mais, par la suite, le tri devient plus facile.

L'information mémorisée s'enrichit, et maintenant nous distinguons nettement entre l'action à exécuter des *agents Intentions* et l'information à propos du monde des *agents Croyances*. Ce qui nous donne l'occasion de présenter ces deux catégories d'agents.

Les agents Intentions sont une forme avancée d'agent intentionnel.

Le drapeau intentionnel d'un seul bit qu'ils portaient a évolué, il est passé par l'état de symbole, et maintenant il est devenu une liste de symboles, et même une liste d'arbres. Ils mémorisent les projets, les programmes que les agents planifient, prévoient d'exécuter. Ils possèdent la nature de règles d'action, d'appels de programmes, que les agents continuent d'interpréter pour générer leurs comportement immédiats ou à venir.



128Taxinomie/Synoptique de l'agent Croyances

Les *agents Croyances.*

D'abord les plus simples, basés sur des agents réactifs, portent des savoirs, des croyances, des informations cognitives décrivant l'état du monde. Ils les utilisent pour générer leur comportement immédiate, qui est en fait peu efficace, car il s'agit là d'une stratégie à court terme.

Ensuite, les agents Croyances finalistes et rationnels, basés sur des structures plus complexes, portent en supplément des informations décrivant les procès : elles montrent comment l'application de règles, l'exécution de programmes transforment l'état du monde. Alors ces connaissances peuvent être utilisées pour générer des tactiques, i.e. une action sur des séquences déterministes plus longues. Ceci est plus efficace car ces agents parviennent à générer leur comportement, sur une profondeur n-aire.

2) Communication pour transmettre une info intentionnelle entre agents

2.A) Introduction

Les agents de haut niveau portent et traitent une information riche, qui n'est pas toujours innée. Ceci implique qu'ils l'ont acquise ou reçue. Donc, dans les paragraphes suivants nous abordons la communication entre agents, qui leur permet de transmettre cette information entre collègues ou d'une génération à l'autre.

Dans le cadre de ce chapitre, nous ne traiterons que :
- Le transfert de symbole entre agents intentionnels
- Le transfert d'ordres entre *agents Intentions*.

2.B) Transfert de symbole entre agents intentionnels

Drapeau mémorisé par un bit, mais étiqueté par un symbole

Tout d'abord, nous traitons l'information la plus simple. Au final, quand elle se trouve mémorisée au sein de l'agent, elle possède la dimension d'un bit.

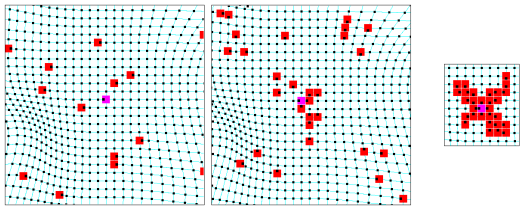
Par contre, puisque l'être peut maintenir plusieurs drapeaux, alors pour caractériser chaque bit, pour le distinguer parmi les autres, nous devons l'étiqueter avec un symbole (ex : faim soif chaud froid peur...).

Quelle est la provenance de cette information ?

Dans le cadre des systèmes multi-agents nous pouvons efficacement utiliser une organisation sociale minimale. Chaque agent perçoit son environnement et analyse les situations les plus simples. Quand il déduit la nécessité de forcer un état interne (stress alerte danger nourriture grooming[46]...) il le force chez lui et il le communique à ses collègues. Pour ce faire la communication au moyen d'un signal audio est toute indiquée. Au sein du groupe, il est relativement facile d'émettre et de distinguer entre les quelques signaux nécessaires.

Introduire la communication au sein du groupe social.

Quand un individu découvre une information, il la communique aux autres. Cette organisation sociale se justifie par l'augmentation de l'efficacité induite pour le groupe. Ainsi la communication entre deux individus apparaît comme la réduction à ce couple, de la communication globale d'un être macroscopique : le groupe social.



129 - Illustration de la communication au sein d'un groupe de chasseurs-cueilleurs

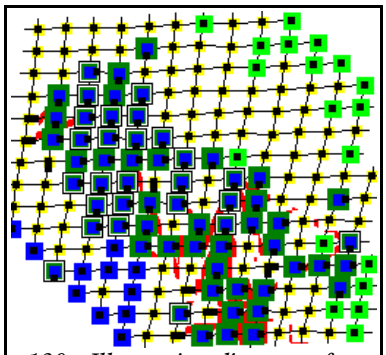
Le dessin ci-contre, tiré de *2DGroupCoop*, illustre l'utilisation d'une communication au sein d'un groupe de chasseurs-cueilleurs :

- Dans la partie gauche, les agents sont dispersés pour chercher de la nourriture. L'agent magenta vient d'en trouver, il appelle les autres.
- Au centre, les agents tropiques convergent vers lui.
- À droite, ils sont tous agglomérés autour de lui.

2.C) Transfert entre agents d'une information codée sur plusieurs symboles

Introduction.

Après avoir évoqué le transfert d'une information réduite à un bit, nous traitons maintenant celui d'une information plus complexe, codée sur plusieurs symboles : sa nature est plus facilement analysable.



130 - Illustration d'un transfert de règles entre agents. Ils sont matérialisés par des traits rouges

Transfert d'informations pour l'agent Croyances

Si elle décrit l'état du monde, ce transfert concerne l'agent *Croyances*, et il sort de notre propos.

Transfert de programme, projets entre agents Intentions.

Si elle est intentionnelle, ce transfert concerne l'*agent Intentions* et possède la dimension d'un appel de programme, d'un planning. Son appellation varie selon la position hiérarchique des agents

- Si le receveur est subordonné de l'émetteur, c'est un ordre.
- Si le receveur est le supérieur de l'émetteur, c'est une suggestion, une invitation.
- Si les deux agents sont au même niveau hiérarchique, il s'agit d'un conseil, d'une remarque.

Exemple.

46 Les relations entre deux individus qui renforcent leur lien social. Ce peut être des gestes (épouillage), ou des signaux (je suis untel, je vais bien, je suis bien avec toi, je suis avec vous).

La figure ci-dessus, tirée de *TransmetCulture*, illustre un transfert *parent→enfant* de règles de conduite : il s'agit donc d'un apprentissage de règles.

3) Conclusion : nous affirmons l'importance des *agents Intensions*

Quand les *agents Intensions* se complexifient, ils deviennent des agents CDI (croyances désirs intentions) très performants. Nous les avons un peu évoqués dans ce chapitre quand nous parlions des *agents Croyances* finalistes et rationnels, nous les verrons plus en détail dans les chapitres suivants.

Ch. 26 Les *agents Croyances* pour simuler et anticiper

Simuler et anticiper à profondeur unaire ou n-aire, au moyen des *agents Croyances*, qui disposent de croyances, mais ne forment ni désir ni intention.

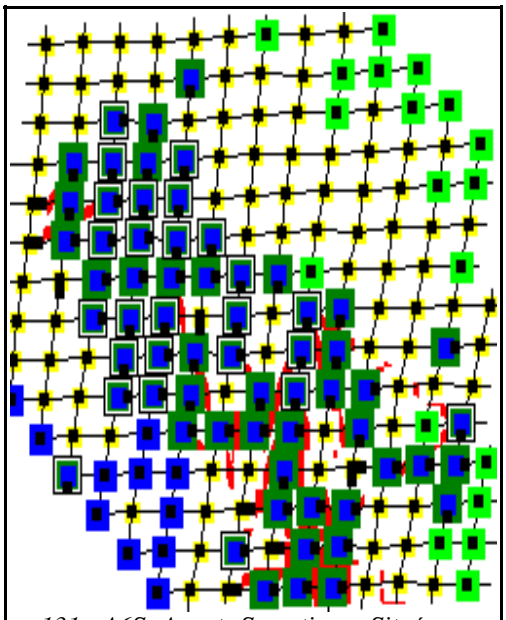
type	*6) Pluralité : 156*	*7) formaliser l'AS*	*8) Exemples*	*9) de l'AT à l'AP*	*10)A Intensions*	*11)A Croyances*	*12) Agt. CDI*
nom	agent situé sur un support	idem	Ag. Tropique(AT)	Ag.Pulsionnel (AP)	Agents intention	Agt. Croyances	Agent CDI
exemple	2DCueilleSurUnRéseau	idem	2DOpportunistes	1DChevron	2DGroupCoop	Scouter	
Nature du cadre	indirectement orienté vers un but. Simplifie la représentation du support	modéliser l'agent situé sur un plan mathématique	Aspect tropique de l'agent situé tropique	Agent pulsionnel	Agents intention puis agent intentionnel	Agnt croyances pour simuler et anticiper	Agent CDI
transi° pr arriver ici	mettre des agents actifs ou passifs en parallèle	Généralise l'état avec variables	Utiliser l'agent de Braitenberg	Forçage somatique de mémoire	Forçage raisonné de mémoire	Apprentissage	croyances désirs intention
Nature du procès effectué	percept-décision-action = f(environnement) états & horloge externes agent tropiques indirects	Herbrand simplifie les règles/procès avec des relations agent-support avec variables	implémentation de l'aspect tropique (avec Braitenberg)	gérer plusieurs comportements	interprétation de programmes	mémorise des croyances simule et anticipe	CDI + une couche de méta = rationnel

1) Implémentation des *agents Croyances*

1.A) Se baser sur les agents situés

Notre but est d'avancer vers un traitement complexe de l'information (colonne 11), basé sur des outils simples, dans le cadre d'agents situés[47] qui maintenant sont sociaux (A6S sociaux). Ils sont souvent réactifs, et, sur un support, ils recherchent de la nourriture pour survivre et s'y reproduire.

Ci contre, une illustration de ces agents sociaux : les liens rouges soulignent les échanges d' information quand ils communiquent entre eux.



131 - A6S: Agents Somatiques Situés en Survie Sur un Support Simple

1.B) S'accrocher à une description de structures

Afin de ne pas trop partir dans des considérations épistémologiques, nous choisissons d'accrocher notre travail à la description de la structure du monde i.e. de son support et de son mobilier ; et donc en particulier de l'implémentation de ses agents.

1.C) Précisons les notions de croyances, désirs et intention chez l'agent CDI

C - croyances : il possède des croyances, i.e. une représentation du monde où il évolue.

Ce sont l'information, les connaissances qui représentent le monde, i.e. son mobilier, son support, les relations mobilier/support, et la réponse du monde aux actions des effecteurs des agents. Par exemple, dans ce chapitre nous travaillerons avec des *agents Croyances* dont les croyances représentent tout le fonctionnement d'un monde d'agents réactifs situés sur un support.

47 Rappel : par agent situé nous entendons A6S (Agent Somatique Situé en Survie Sur un Support Simple).

Précisons ici un point de vocabulaire pour expliquer le choix du terme *croyances* plutôt que celui de *connaissances* ou de *savoir* : l'agent dispose d'une connaissance sur le monde, et il la tient pour vraie, et s'en sert en accord avec ce point de vue. Mais il se trouve que son savoir peut être faux[48], et donc nous considérons qu'il possède des croyances mais pas des connaissances. Alors, en toute rigueur, nous devons attribuer à l'agent une aptitude *C* à croire, mais pas *S* à savoir.

D - désirs : L'agent gère des désirs, i.e. des formes plus élaborées de buts.

Quand l'agent est finaliste, téléologique, il possède des états internes. Ils sont assimilables à des buts car ils sous-tendent et orientent toute son activité. Cependant leur structure est un peu plus complexe que des drapeaux, représentables par un simple bit ; elles consistent plutôt en des symboles, des relations, ou des listes de ces derniers. Ex : *Agent1 -but→ sur(Table, Panier)* : *l'agent1 veut que le panier soit sur la table.*

L'agent intentionnel est gestionnaire de ses buts car il possède la faculté de forcer ou d'abandonner ses états mentaux internes.

I - intentions : l'agent maintient une mémoire de ses intention.

L'agent dispose d'un planning : en son sein, il possède une structure de données où il mémorise une liste d'actions qu'il a l'intention d'exécuter.

Parfois il reçoit cette liste de ses collègues, mais en général elle provient plutôt de ses supérieurs. Certains agents, plus autonomes sont capables de l'élaborer[49] eux-mêmes.

L'*agent Intension* est capable d'exécuter ce planning, cette liste. Cette aptitude correspond à celle d'interpréter les programmes.

1.D) Pour simuler et anticiper il nous suffit de disposer d'*agents Croyances*

Il est possible de fabriquer des *agents Croyances*, i.e. des agents qui disposent de *Croyances*, sans qu'ils possèdent ni *Désirs*, ni *Intentions*. Pour cela il suffit de donner une croyance à des agents réactifs. Maintenant voyons comment un tel être serait viable.

Les *agents Croyances* possèdent des *Croyances.*

Pour être capable de simuler ou d'anticiper dans le monde, l'agent doit au moins posséder des croyances, i.e. disposer d'informations sur le monde.

Les *agents Croyances* ne possèdent pas de désirs.

L'*agent Croyances* n'est pas une aberration, c'est seulement un rêveur. Dans ce cas il n'est pas obligé de gérer des désirs et d'interpréter ses intentions. D'ailleurs nous ne préciserons même pas le contenu de ces champs chez lui. Nous l'utiliserons en simulation et anticipation.

- Quand il simule, il se laisse porter, sans but, par son imagination : il ne possède pas de désirs.
- Quand il anticipe, il simule et stoppe son rêve avant qu'il ne devienne cauchemar.

Les *agents Croyances* que nous utiliserons, ne possèdent pas d'intention

Ici, nous avons fait le choix de travailler avec des *agents Croyances* : des agents réactifs munis de croyances. À ce titre, ils ne forment pas d'intention. Comme leur nom l'indique, ils réagissent, i.e. ils agissent spontanément : à chaque pas il génèrent leur action dans l'immédiat. Ainsi, dans ce cadre, l'intention, se réduit à la commande de l'action qui provient de la fonction décision et active l'effecteur. Elle n'est donc pas stockée, mémorisée dans un plan, dans un programme qui sera exécuté ensuite instruction après instruction. À aucun endroit dans notre synoptique elle ne se présente sous la forme d'une liste d'actions qui serait ensuite interprétée. C'est pourquoi nous disons que ces *agents Croyances* ne possèdent pas d'intention. C'est pour cela que nous considérons qu'avec notre modèle, nous ne sommes pas en présence d'*agent Intentions*.

48 C'est par exemple l'histoire des cueilleurs et de la pomme. Deux agents voient la même pomme. Le premier le plus rapide la prend. Dans ce cas, le second croit toujours que la pomme est sur la case devant lui, mais ce n'est plus le cas.

49 Ils sont capables d'effectuer cela, au moyen de la démarche dite de *planification de tâches*.

2) Communication pour transmettre une information entre agents

2.A) Introduction

Les agents de haut niveau portent et traitent une information riche, qui n'est pas toujours innée. Ceci implique qu'ils l'ont acquise ou reçue. Donc, dans les paragraphes suivants nous abordons la communication entre agents, qui leur permet de transmettre cette information entre collègues ou d'une génération à l'autre. Pour chacun d'eux, nous distinguons la nature, i.e. la taille, la complexité de ce transfert.

2.B) Transfert de symbole entre agents

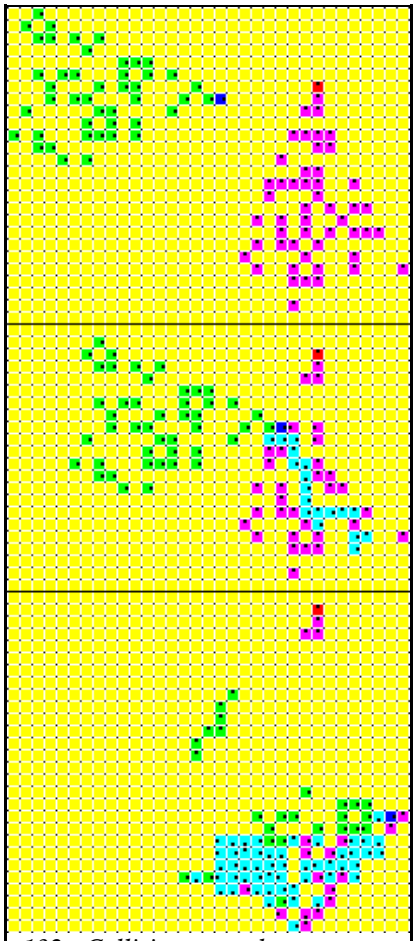
Dans le chapitre sur les agents intentionnels et les agents Intentions, nous avons déjà traité le transfert de symbole entre agents et avons expliqué qu'il pouvait être interprété comme un échange d'information (concernant les *agents Croyances*) ou un échange d'intentions concernant les *agents Intentions*.

2.C) Transfert entre agents d'une information de lieu, de position

Après avoir évoqué une information réduite à un bit, nous traitons maintenant une information de lieu ou de position ; elle décrit l'état du monde. Elle est donc un peu plus complexe. Au final, quand elle se trouve mémorisée au sein de l'agent, elle se code sur plusieurs dizaines de bits.

Par exemple, l'information échangée signifie *Ici il y a de quoi manger* ou *Je suis ici...* Elle se fait par écholocation au moyen d'une communication audio entre les agents situés.

En fait, cette thématique a déjà été abordée dans le cadre des agents tropiques et les *agents Intensions*, donc nous ne l'approfondirons pas.



132 - Collision entre deux groupes

2.D) Transfert de règles entre agents

Finalement, nous décrivons un traitement riche de l' information, un transfert de règles qui décrivent un procès à effectuer. Elles possèdent donc la dimension d'un programme et sont nécessaires aux *agents Croyances*. Si ils ne sont pas des *agents Intentions*, ils peuvent utiliser ces règles pour simuler. Et, nous verrons que s'ils sont des *agents CDI*, ils les utiliseront pour former des projets, planifier des programmes[50]. Dans les deux cas, elles sont complexes et ne peuvent être codées que par l'assemblage de plusieurs symboles au moyen de structures (liste, tableau ou chaîne).

Les trois exemples que nous traitons concernent un transfert de règles entre générations, cependant ils ne sont pas exactement semblables. Le premier consiste en un croisement de patrimoine culturel, le second à un apprentissage et le dernier à un enseignement.

50 Attention, voici une seconde définition de l'intention. Elle est à prendre dans le sens de programme, projet, planning. Dans le langage quotidien elle correspond à la phrase : J'ai l'intention de faire ceci. Elle est portée par les *agents Intentions*.

a) Échange d'individus entre 2 groupes ≃ échange d'info génétique entre 2 bactéries

Apprentissage par échange de gènes entre individus.

Notre but est de transférer des règles de comportement vers un individu. Mais, dans nos implémentations informatiques, nous cherchons à effectuer de façon simple des traitements complexes. Au lieu d'écrire des simulation où les agents communiquent pour échanger des règles de comportement, dans une démarche bionique, nous choisissons de copier la nature. Nous regardons comment les bactéries apprennent à lutter contre les germes pathogènes extérieurs : elles échangent des gènes. À l'instar des algorithmes génétiques, nous décidons de seulement effectuer des croisements de gènes entre deux individus.

Simuler un échange de gènes entre bactéries par un échange d'individus entre groupes.

Cet échange d'ADN entre les bactéries de la biologie peut se simuler, dans le cadre de l'IAD, par un échange d'individus entre deux essaims d'agents situés. Pour cela, il suffit de prendre des groupes où les individus possèdent des compétences différentes : en échangeant leurs spécialistes, les communautés échangent des compétences.

Simuler la membrane d'une bactérie par le tropisme vers le leader.

En biologie, la cohésion d'une bactérie est assurée par une membrane, et, dans notre simulation, pour obtenir le regroupement des agents en essaim, nous utilisons un tropisme vers le meneur. Quand il se déplace, il appelle ses ouailles, et les membres du groupe se tiennent derrière lui, le plus proche possible. Nous avons déjà expliqué que ce type de simulation s'implémente facilement au moyen d'agents tropiques.

Le croisement des groupes induit celui des patrimoines.

Pour induire l'échange d'individus entre deux essaims nous croisons, plus ou moins serrées, les trajectoires des deux leaders. Ainsi, dans chaque groupe, quelques agents qui suivaient un meneur, se retrouvent à suivre l'autre. Finalement ils changent de groupe. À l'issue de l'opération, en échangeant des individus, les groupes ont échangé quelques bribes de culture.

Exemple : la simulation *Croiser*.

Au sein de notre progression de simulations, le programme *Croiser* illustre cette démarche. La figure ci-contre illustre une collision tangentielle entre le groupe vert, mené par un leader bleu qui marche vers l'est et le groupe magenta, mené par un leader rouge qui va au nord.

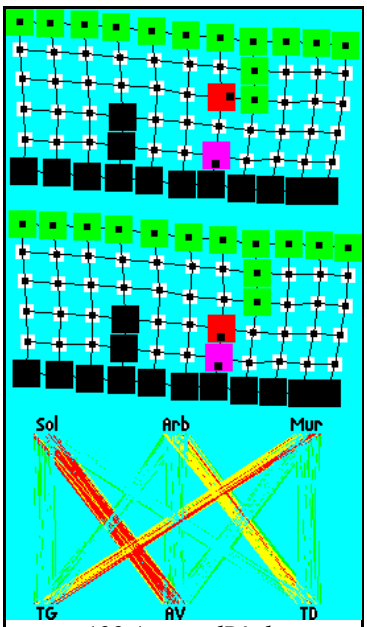
- Dans la partie haute de l'illustration, nous voyons les groupes en marche.
- Dans la partie du milieu, ils entrent en collision. Dans cette illustration, les agents de couleur cyan correspondent à des êtres qui sont bloqués parce qu'ils ne peuvent pas avancer.
- En bas, nous remarquons que le leader rouge se retrouve avec seulement trois agents magenta et sept verts. L'exemple choisi est caricatural, l'échange entre les deux groupes n'est pas équilibré : dans cette configuration le leader bleu perd très peu des siens et récupère la plupart des agents cyan.

b) Un réseau de neurones apprend par l'exemple

Dans la simulation *2DApprendRegle*, un réseau constitué d'une couche de 9 neurones, apprend trois règles par l'exemple.

Dans un monde muni d'obstacles disposés très régulièrement, un parent doit enseigner trois règles à son enfant (*avancer, tourner à droite, tourner à gauche*). Le but est d'effectuer ce transfert de culture autrement que par échange langagier. La méthode retenue est celle de l'exemple par le geste : le parent effectue le trajet et son enfant le suit par écholocation.

Dans un monde classique à 4 voisinages le suivi effectué par l'enfant est parfait et il apprend facilement. Dans un monde à 6 voisinages (nid d'abeille), le suivi demeure de bonne qualité au point que l'apprentissage reste possible au prix d'un lissage statistique.



133 ApprendRègle

Dans l'illustration ci-contre, le parent cyan guide son enfant rouge, parmi les murs noirs et les arbres verts.

Dans la partie haute de l'illustration, l'enfant voit un arbre dans la case devant lui et il doit tourner à droite pour rejoindre son parent.

Dans la partie médiane, il l'a rejoint. Ainsi il apprend la règle : *arbre→tourne droit*. De même, il apprend la règle : *sol→avance,* et *mur→tourne gauche*.

En bas, nous voyons l'apprentissage progressif des règles par les 9 neurones au moyen de deux démarches antagonistes : les 3 neurones activés (en jaune-orange) renforcent leur poids, ceux qui sont délaissés (en vert clair) s'atrophient progressivement.

c) Éducation : transfert langagier de règles de conduite (Ex: *TransmetCulture*)

Démarche éducative.

Il s'agit d'éducation, le transfert se passe selon une relation hiérarchique *parent→enfant* : l'enfant accepte l'information transférée et donc ne la remet pas en cause.

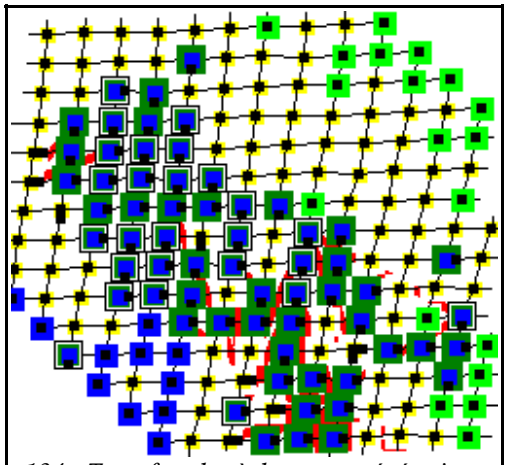
Comme le transfert consiste en règles de conduite, l'auditeur les place dans sa base de règles, et elles vont ensuite déterminer son comportement.

Transfert riche au moyen d'un protocole langagier.

Selon un point de vue structurel et implémentation, ce transfert transite par une RAM, i.e. par une mémoire temporaire, qui est très facile à écrire (capable de grande performance). Nous sommes au moins en présence d'un *agent Croyances*, qui entretient un savoir à propos du monde. Ici il s'agit d'un comportement.

Donc, c'est aussi un transfert qui met en oeuvre un protocole riche (lexique syntaxe sémantique), correspondant à celui du traitement du langage naturel (TLN).

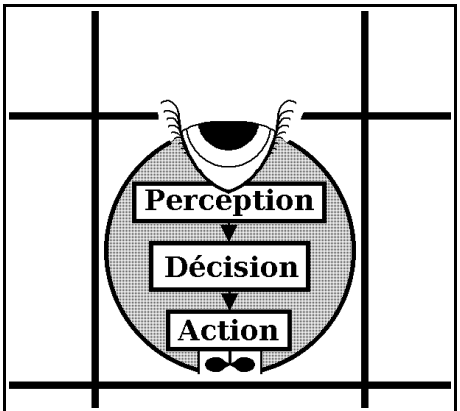
Dans l'illustration ci-dessus les transferts de règle *parent→enfant* sont matérialisés par des lignes rouges.



134 - Transfert de règles entre générations

2.E) Remarque

Le transfert de culture entre individus est très efficace à cause de sa relative rapidité. Dès qu'un individu invente un nouveau comportement, il peut le transmettre à ses enfants et même à ses proches. Ainsi la vitesse de ce transfert oral de culture est nettement plus grande que celui du patrimoine génétique dans le cadre de l'évolution des espèces. En effet, si une mutation génétique apparaît au sein d'un groupe, il faut plusieurs générations avant que son efficacité soit confirmée par la sélection naturelle.



135 - Simulation d'un pion sur une grille

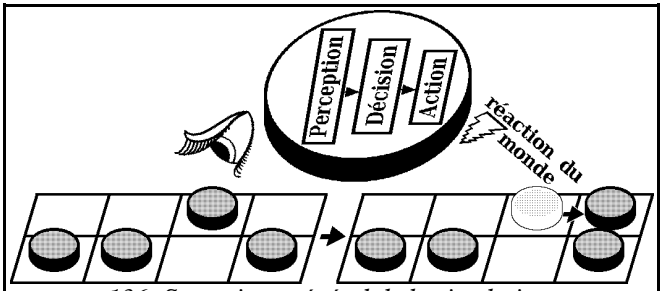
3) Retour sur le synoptique de la simulation

Avant d'entrer dans le vif du sujet faisons quelques rappels. Revenons un peu sur le synoptique générale de la simulation d'une action élémentaire, car nous en aurons besoin dans ce chapitre. L'action élémentaire d'un agent réactif situé se décompose ainsi :

3.A) Cycle *élémentaire perception décision action*

Perception.
L'agent perçoit l'état du monde au travers de ses capteurs.

Décision.
L'agent réactif adresse sa table, utilise sa base de règles ou ses fonctions combinatoires câblées dans sa circuiterie. Ainsi il prend la décision d'une action, i.e. d'activer un actionneur.



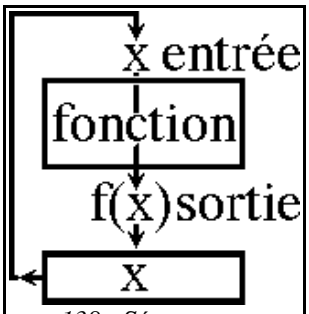
136- Synoptique général de la simulation d'une action élémentaire

Action.
L'agent active un de ses effecteurs, qui agit sur le monde. Ce dernier répond à cette action en fonction de ses lois internes. Finalement, nous obtenons un nouvel état du monde, et le cycle peut recommencer.

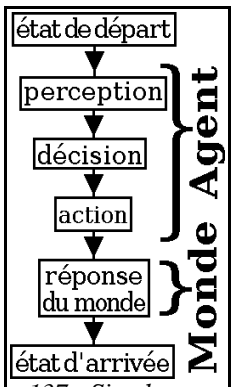
3.B) Le cycle de simulation fait intervenir la réponse de l'agent et du monde

De façon évidente, la simulation d'un cycle élémentaire, i.e. du déplacement unitaire du pion sur le support fait intervenir le comportement du pion au-travers des trois opérations *perception décision action* (ci-contre, à droite).

Mais il faut garder à l'esprit que cette simulation doit encore être capable de modéliser le fonctionnement du monde : le programmeur doit aussi décrire la réponse du support aux actions de l'agent (avancer tourner manger...).

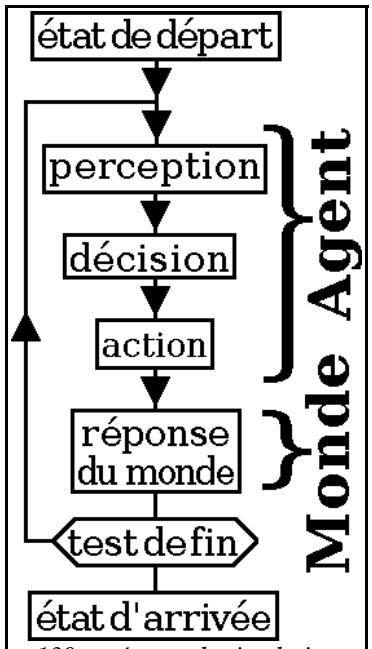


138 - Séquenceur à fonction câblée



137 - Simuler un pas

Pour programmer un cycle de simulation :
- *Commencer depuis l'état de départ du monde*
- *Une mémoire fournit la décision de l'agent situé dans le monde.*
- *Transformer l'état du monde en fonction de l'action de l'agent.*
- *Finalement nous obtenons l'état d'arrivée du monde.*



139 - n étapes de simulation

3.C) Un séquenceur pour simuler à <u>profondeur *n*</u>

Revenons sur la simulation à profondeur *n* au moyen d'un séquenceur (ci-dessus, à gauche). Pour simuler au moyen d'un séquenceur, nous avons utilisé une fonction combinatoire couplée à un registre. La fonction reçoit en entrée l'état du système et fournit en sortie son prochain état, qui sera mémorisé dans le registre lors de la prochaine impulsion d'horloge. La sortie du registre fournit l'état du système.

3.D) Passer de simulation sur l'ordinateur à imagination par *l'agent Croyances*

a) La simulation à profondeur *n*

Pour simuler à profondeur *n* il faut rajouter aux éléments précédents beaucoup d'éléments :

- Une étape d'initialisation qui fournit l'état de départ du monde.
- Une mémoire qui reconstitue la décision de l'agent situé dans le monde.
- Une procédure qui fait évoluer l'état du monde en fonction de l'action de l'agent.
- Un registre qui mémorise l'état du monde quand il évolue sous les actions de l'agent.
- Un test de fin qui fait cycler le processus et l'arrête au bout de *n* boucles.

b) Un *agent Croyances* dispose de la représentation du monde

Sur un ordinateur nous modélisons un *agent Croyances*. Par définition, il dispose de la représentation du monde. Elle consiste en ceci : le support, son mobilier, les relations mobilier/support, et la réponse du monde aux actions des effecteurs. Convenons de choisir un agent réactif, un pion qui se déplace sur une grille au milieu des obstacles.

- Le support consiste en une grille.
- Le mobilier est constitué du pion et des obstacles.
- Les relations mobilier/support : le pion et les obstacles sont disposés aléatoirement sur la grille.
- La réponse du monde aux actions des effecteurs : Le pion avance si la case devant lui est libre.
- La grille se reboucle sur elle-même : si l'agent sort au sud, il réapparaît au nord, etc...

c) Notre *agent Croyances* simule un agent réactif

Par définition, cet *agent Croyances* dispose de la représentation d'un monde. Cependant, quand il utilise cette information pour imaginer l'évolution du monde sous l'action des agents, il met seulement en oeuvre des agents réactifs.

Or justement, à ce stade nous venons d'étudier comment utiliser des agents réactifs pour modéliser un fonctionnement en simulation. Donc nous pouvons récupérer tous les résultats précédents et les prolonger pour voir comment simuler au moyen d'un agent réactif.

d) *agent Croyances* pour simuler un pion sur une grille 2D

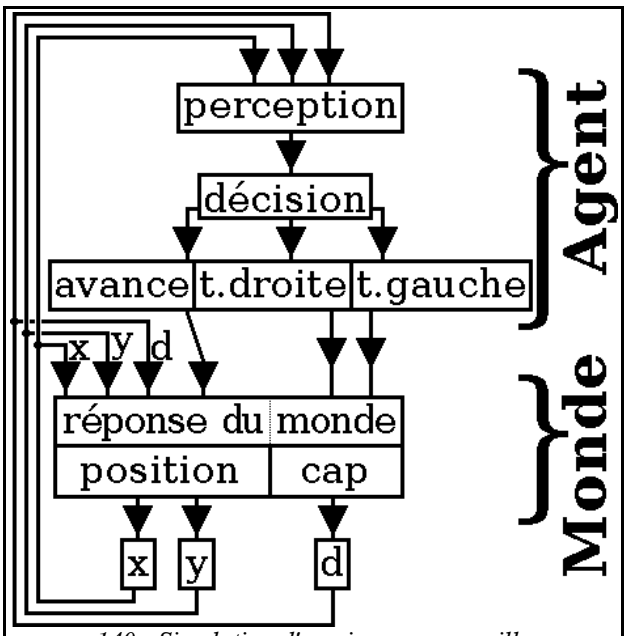
Pour simuler un pion dans un couloir au moyen d'un séquenceur, nous avons utilisé une fonction et un registre. Maintenant nous décrivons la circuiterie nécessaire à l'*agent Croyances* pour qu'il soit capable d'imaginer l'évolution du monde sous l'action du pion ?

Simuler la perception du monde par l'agent.

Quelle fonction utiliser pour modéliser la réponse du monde en perception ?

La perception du pion dans le couloir était uniquement fonction de sa position *x*, aussi nous utilisions seulement une fonction f(*x*).

Ici celle du pion sur la grille est déterminée par sa position *(x, y)* et sa direction *d*. Il nous faut donc utiliser une fonction à trois variables : f*(x, y, d)*, où *d* correspond à la direction.



140 - Simulation d'un pion sur une grille

Quels registres pour mémoriser l'état du système ?

Pour mémoriser l'état du système, le séquenceur que nous venons de décrire n'avait qu'à enregistrer un état : la position en x du pion sur le support.

Maintenant l'*agent Croyances* doit mémoriser trois valeurs : les positions *x, y* et la direction *d*.

Quelle fonction utiliser pour décrire la réponse du monde à l'action d'avancer

Disons que si le pion butte sur un obstacle, il est bloqué. Comment définir cette fonction pour décrire la réponse du monde à l'actionneur *avancer*. Ceci peut se faire facilement en disant que si le pion perçoit un obstacle dans la case devant lui, même s'il commande l'action d'avancer, elle est inhibée.

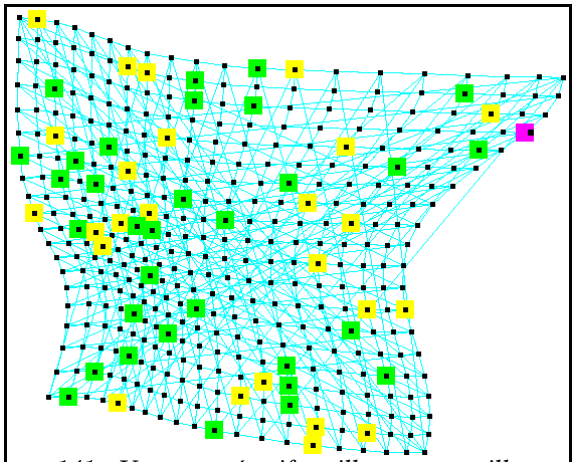
Quelle fonction utiliser pour décrire la réponse du monde aux actions pour tourner

Nous convenons que le pion peut toujours pivoter sur lui-même. La fonction qui calcule la direction doit savoir le cap actuel. Quand il faut tourner à droite, elle l'incrémente de 90° (modulo 360). Quand il faut tourner à gauche, elle le décrémente...

e) Exemple de simulation d'un agent réactif sur une grille

Ci-contre, une illustration tirée de *2DCueilleSurNet* un programme qui se rapproche de ce profil : l'agent est de couleur cyan, les obstacles sont jaunes et la nourriture verte.

Il simule un agent réactif en situation de cueillette sur un réseau de lieux.



141 - Un agent réactif cueille sur une grille

3.E) Fonctionnement en anticipation

a) Synoptique général de l'anticipation

Globalement, la transition de la simulation à l'anticipation peut se résumer ainsi : à chaque intervalle de temps, l'agent perçoit son environnement, et décide l'action à effectuer.

Et, de façon générale l'anticipation intervient ici : avant d'agir, un module intercepte l'action prévue, et anticipe son résultat afin de tester si elle n'est pas néfaste ; si c'est le cas, il l'inhibe carrément.

Chez les différents *agents Croyances*, à l'intérieur de ce fonctionnement général, nous distinguons plusieurs synoptiques, variant du plus léger au plus complexe.

b) Remarque sur la profondeur de l'anticipation

En simulation, selon que nous utilisons une structure itérative ou non, nous travaillons respectivement à profondeur *1* ou à profondeur *n*. Puisque l'anticipation est basée sur la simulation, nous retrouverons ce même clivage. Nous traiterons deux cas d'anticipation qui se distinguent en fonction de leur profondeur :

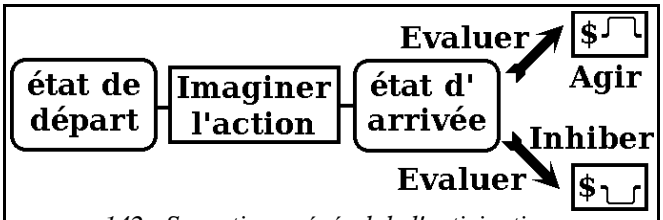
- L'anticipation à profondeur 1 ou anticipation à profondeur unaire.
- L'anticipation à profondeur n ou anticipation à profondeur n-aire.

3.F) Anticipation à profondeur unaire

a) Synoptique simplifié de l'anticipation

Dans un premier temps, le cadre de l'anticipation à profondeur unaire permet une simplification de la représentation des données nécessaires :

Dans un fonctionnement classique (illustration ci-contre) selon un synoptique général mais lourd, nous simulons l'action afin d'imaginer l'état d'arrivée obtenu, et ensuite nous évaluons ce dernier.



142 - Synoptique général de l'anticipation d'une action élémentaire

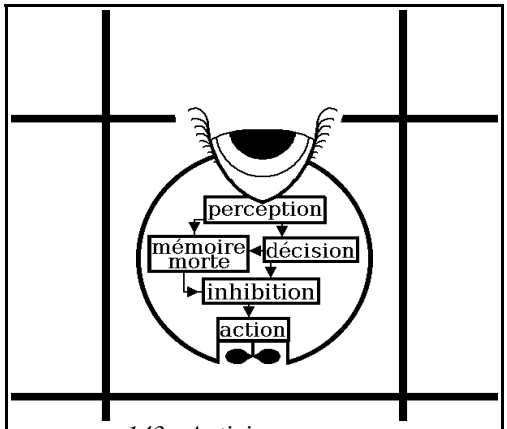
Dans un fonctionnement, selon un synoptique simplifié (ci-contre), nous utilisons une mémoire pour restituer directement le résultat d'arrivée correspondant à l'état de départ. Dans ce cas le synoptique obtenu est plus simple car il court-circuite la phase d'évaluation ; mais pour l'agent, le travail de mémorisation/apprentissage est plus grand.

b) Anticiper avec un *agent Croyances* de base

Nous commençons avec une simulation simple qui nécessite seulement un agent tropique amélioré.

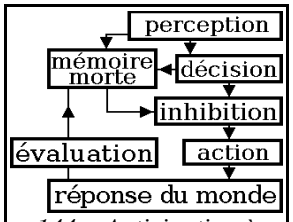
Dans son synoptique, d'abord nous retrouvons le fonctionnement classique *perception décision action* de l'agent réactif. Cependant, nous remarquons un étage *inhibition* entre les fonctions *décision* et *action*.

Cette fonction intermédiaire est constituée d'une mémoire morte qui, pour chacune des situations en entrée, restitue le résultat mémorisé précédemment dans les mêmes conditions, pour chaque action effectuée. Si ce dernier est néfaste, ce module inhibe purement et simplement l'action prévue.



143 - Anticiper avec un agent croyances *de base*

Nous sommes donc ici en présence d'un fonctionnement en restitution. La mémoire restitue une connaissance accumulée précédemment, sans que nous ne précisions comment.



144 - Anticipation à profondeur unaire (phase d'apprentissage)

Ce mode de fonctionnement est très simple, car, en phase d'apprentissage, la mémoire n'a pas besoin d'apprendre la réponse du monde aux actions de l'agent, mais il lui suffit de mémoriser l'évaluation du résultat obtenu dans cette configuration.

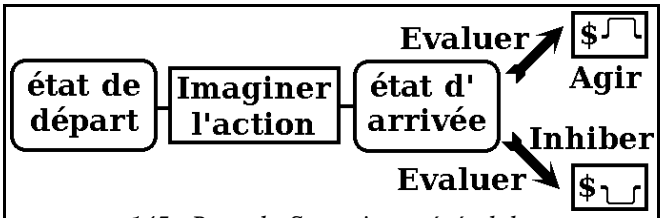
Le dessin ci-contre précise comment est effectué l'apprentissage : en phase d'entraînement la mémoire continue de recevoir les entrées *perception* et *décision*, mais elle doit en plus recevoir *l'évaluation de la réponse du monde* qu'elle mémorise.

Lors du fonctionnement en anticipation, la lecture de cette mémoire sert à l'agent pour évaluer la justesse de l'action prévue puis effectuée dans cette configuration. Ainsi il se fait une idée de l'intérêt de l'action planifiée et peut savoir s'il doit l'inhiber ou la laisser se dérouler tranquillement.

c) L'*agent Croyances* riche pour anticiper à un coup

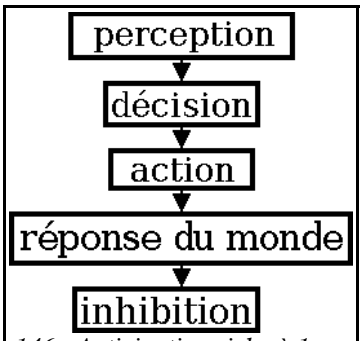
Introduction.

Dans l'anticipation au moyen d'un *agent Croyances* élémentaire, pour chaque situation donnée, ce dernier ne connaît que l'évaluation du résultat de l'action prévue : c'est un peu comme avec les phéromones des insectes sociaux, il sait juste la valeur de l'état d'arrivée obtenu, mais ignore la réponse du monde aux actions des agents.



145 - Rappel : Synoptique général de l'anticipation d'une action élémentaire

Synoptique d'un *agent Croyances* riche capable d'anticipation unaire.



146 - Anticipation riche à 1pas

Par contre, ce n'est pas le cas de l'*agent Croyances* riche : il est capable de prévoir l'état d'arrivée du monde, et à partir de celui-ci, il calcule l'intérêt de l'action projetée. Ainsi nous obtenons ce synoptique :

À chaque cycle élémentaire, il effectue toutes ces opérations :

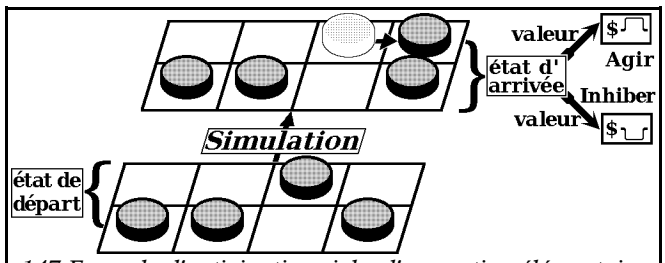
- Perception : il perçoit et analyse l'état de départ du monde.
- Décision réactive : dans ce contexte, il décide instantanément quelle action effectuer.

- Simulation de l'action : il imagine quel sera l'état d'arrivée du monde transformé par cette action.

- Évaluation : considérant cet état d'arrivée imaginé, il évalue quel est pour lui l'intérêt de cette action prévue.

- Si le résultat obtenu est intéressant pour lui il active son actionneur et l'état obtenu est la réponse du monde à cette action.

- Sinon, le résultat ne lui plaît pas, il inhibe l'action en cours et exécute une action de sortie : l'action d'évitement.



147 Exemple d'anticipation riche d'une action élémentaire

d) Exemple de programme d'anticipation à profondeur 1

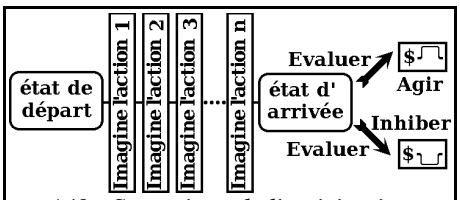
Nous n'avons pas directement implémenté de programme d'anticipation à profondeur 1 au moyen d'*agents Croyances*. Nous avons seulement contourné le problème et implémenté une application succédanée d'anticipation. Ainsi le programme *InsecSoc* illustre un exemple d'anticipation à profondeur 1, au moyen d'agents situés.

En effet, cette simulation ne met pas en oeuvre des *agents Croyances*, car les êtres simulés n'entretiennent pas en leur sein une représentation du monde : il ne font que marquer le support au moyen d'une phéromone de stress, et quand il en rencontrent une, il modifient leur comportement : il inhibe l'action prévue pour effectuer un geste d'évitement.

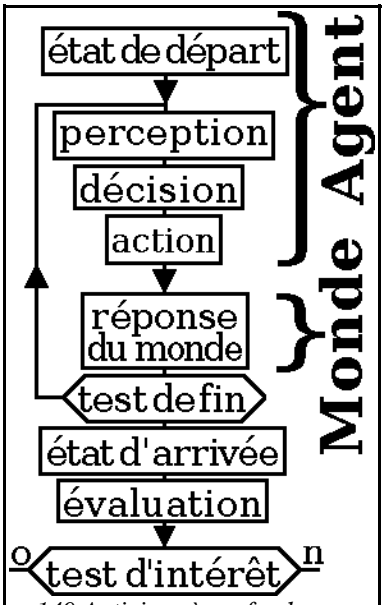
3.G) Anticipation n-aire (à profondeur *n*)

a) D'anticipation unaire à anticipation n-aire

L'anticipation à profondeur n-aire conserve le même synoptique que précédemment (Synoptique général de l'anticipation - à droite). Son profil général consiste simplement à imaginer à terme plutôt que sur l'étape suivante, i.e. à simuler l'évolution du monde sur N pas, sur N actions.



148 - Synoptique de l'anticipation à profondeur n



149 Anticiper à profondeur n

Elle peut se résumer ainsi : avant d'agir, l'agent vérifie si en prolongeant N fois l'action dans cette direction elle débouche sur un résultat intéressant ; et si ce n'est le pas le cas, il l'inhibe et exécute une action de sortie : l'acte d'évitement.

b) Synoptique de l'anticipation à profondeur *n*

Description d'un cycle élémentaire.

Maintenant attachons-nous à décrire un cycle élémentaire de ce travail en anticipation.

D'abord, partant de l'état initial du monde, l'agent simule *n* fois l'application de l'action élémentaire, i.e. de la boucle de l' agent réactif : *perception, décision réactive.*

À l'issue de ce travail d'imagination, il obtient l'état d'arrivée simulé du monde calculé par l'application de ces *n* actions élémentaires.

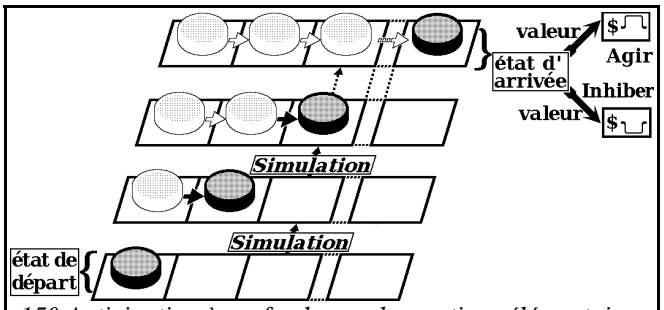
Finalement, il soumet ce résultat intermédiaire à une évaluation, afin d'apprécier son intérêt.

- Si, le résultat évalué ne lui plaît pas, il évite d'orienter son action dans cette direction et exécute une action de sortie : l'acte d'évitement, dont le rôle est de changer de direction et/ou l'état du monde. Et c'est la fin du cycle élémentaire.

- Si le résultat obtenu est intéressant pour lui, il part dans ce sens, ce qui, dans le monde réel l'amène à effectuer cette séquence d'actions et l'état obtenu est la réponse du monde à cette action cette direction. Et c'est la fin du cycle élémentaire.

Description du fonctionnement complet.

Le dessin ci-dessus à droite illustre l'anticipation à profondeur *n* du déplacement d'un pion sur une couloir de cases. Au moment du départ (au bas du dessin), le pion est totalement à la gauche du couloir. À chaque étape, il progresse d'une case vers la droite. Au bout de n cases, il est rendu à l'extrémité droite du couloir (au haut du dessin). À cet endroit, l'agent évalue la configuration obtenue pour décider s'il doit agir ou non dans cette direction...



150 Anticipation à profondeur n de n actions élémentaires

c) Exemple de programme d'anticipation à profondeur *n*

Nous n'avons pas directement programmé d'exemple d'anticipation à profondeur *n* au moyen d'agents situés. Cependant nous décrivons dans le polycopié suivant *Scouter* une simulation à profondeur *3*. D'un côté, elle n'est pas utilisée pour anticiper mais pour planifier, cependant elle constitue un exemple intéressant car elle applique concrètement les techniques de simulation au moyen des *agents Croyances* étudiées ici.

4) Conclusion

Le mécanisme d'anticipation complet à profondeur *n* améliore encore la puissance de traitement de l'information de l'agent. La prochaine étape sera la planification qui améliore notablement son fonctionnement.

Ch. 27 L'agent *Croyances Désirs Intentions* utilisé pour planifier

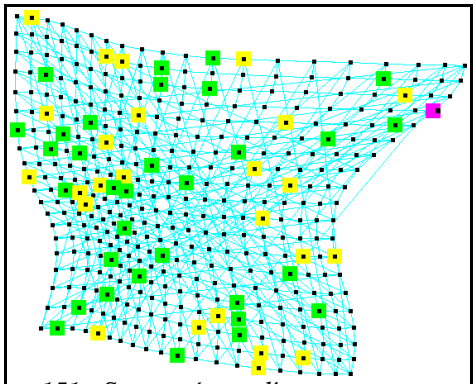
Colonne 12 : L'agent *Croyances Désirs Intentions* (CDI)

type	*6) Pluralité : 156*	*7) formaliser l'AS*	*8) Exemples*	*9) de l'AT à l'AP*	*10)A Intensions*	*11)A Croyances*	*12) Agt. CDI*
nom	agent situé sur un support	idem	Ag. Tropique(AT)	Ag.Pulsionnel (AP)	Agents intention	Agt. Croyances	Agent CDI
exemple	2DCueilleSurUnRéseau	idem	2DOpportunistes	1DChevron	2DGroupCoop	Scouter	
Nature du cadre	indirectement orienté vers un but. Simplifie la représentation du support	modéliser l'agent situé sur un plan mathématique	Aspect tropique de l'agent situé tropique	Agent pulsionnel	Agents intention puis agent intentionnel	Agnt croyances pour simuler et anticiper	Agent CDI
transi° pr arriver ici	mettre des agents actifs ou passifs en parallèle	Généralise l'état avec variables	Utiliser l'agent de Braitenberg	Forçage somatique de mémoire	Forçage raisonné de mémoire	Apprentissage	croyances désirs intention
Nature du procès effectué	percept-décision-action = f(environnement) états & horloge externes agent tropiques indirects	Herbrand simplifie les règles/procès avec des relations agent-support avec variables	implémentation de l'aspect tropique (avec Braitenberg)	gérer plusieurs comportements	interprétation de programmes	mémorise des croyances simule et anticipe	CDI + une couche de méta = rationnel

1) Introduction : l'agent réactif

1.A) Position du problème

Nous continuons de nous placer dans le cadre des agents situés de type A6S[51]. Mais à cette étape, le support possède deux dimensions : c'est une grille ou une certaine forme de réseau. L'agent est donc placé au milieu d'une certaine diversité. Maintenant, dans certaines configurations, avant d'agir, il doit choisir entre plusieurs règles car plus d'une sont susceptibles d'être déclenchées. Par exemple si l'agent se trouve devant un obstacle, il peut tourner à gauche ou à droite. Ainsi, à ce stade, nous devons régler les problèmes de séquencement des règles, i.e. de l'ordre de leur application.



151 - Sur un réseau, l'agent magenta cherche de la nourriture verte, au milieu des obstacles jaunes

1.B) Agents réactifs avec un choix heuristique dans les règles

a) La solution de l'inhibition des règles

La solution la plus simple est d'utiliser différentes heuristiques pour choisir une unique règle parmi toutes celles qui, à chaque instant, sont potentiellement déclenchables.

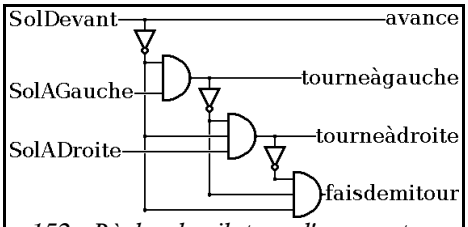
La méthode la plus utilisée consiste en un classement ou câblage hiérarchique : la première règle dont les conditions d'application sont satisfaites est déclenchée, et elle inhibe les suivantes.

Voici grosso-modo la hiérarchie utilisée :
- les règles de survie immédiates : réflexe de retrait ou d'inhibition de l'action dangereuse...
- Les règles pour la survie du groupe social : attention danger, il faut fuir...
- Les règles de gestion des besoins somatiques : faim, soif, chaud, froid...
- Les règles de déplacement : pour éviter les obstacles.
- ...
- règles plus abstraites...

51 A6S : Agents Somatiques Situés en Survie sur un Support Simple.

b) Implémentation électrique

Le schéma électronique ci-contre montre la circuiterie de logique combinatoire utilisée pour piloter un robot fourrageur. C'est agent réactif situé sur un réseau où se trouvent des obstacles. Elle implémente l'algorithme suivant selon une structure dite de subsumption : quand nous montons dans la hiérarchie des règles, une règle supérieure recouvre, domine sa subordonnée, elle la subsume.



152 - Règles de pilotage d'un agent se déplaçant au milieu d'obstacles

c) Implémentation informatique

- S'il n'y a pas d'obstacle devant, alors avancer ;
- Sinon s'il n'y a pas d'obstacle à gauche, alors tourner à gauche ;
- Sinon s'il n'y a pas d'obstacle à droite, alors tourner à droite ;
- Sinon faire demi-tour.

d) Remarque

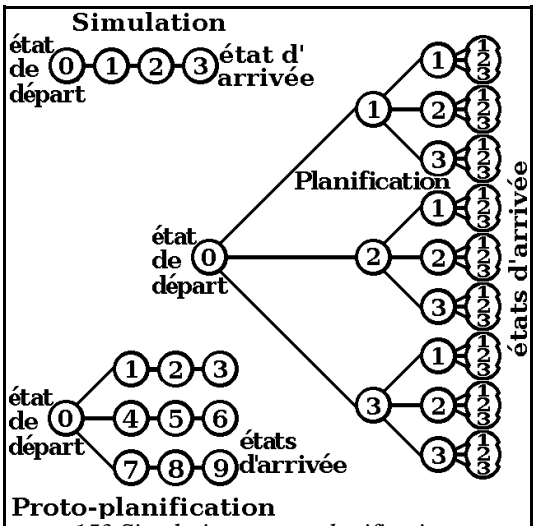
Le lecteur attentif remarquera comment, dans les deux implémentations, la première action déclenchée inhibe la suivante :

- Dans l'algorithme informatique, la répercussion se fait au travers des *si alors sinon* imbriqués.
- Dans le schéma électrique la répercussion de l'application de l'action se fait au moyen de la fonction *inverseur* qui bloque la fonction *et* suivante en lui fournissant un niveau 0 inhibiteur.

2) L'agent CD (*Croyances Désirs*)

2.A) Plus de possibilités chez un agent intentionnel

Maintenant passons à un être plus élaboré que le simple agent réactif : il est intentionnel. Il possède des croyances sur le monde, i.e. une représentation du support, du mobilier et donc des agents, i.e. en particulier de la façon dont le support répond aux actions des agents qui y agissent.



153 Simulation, proto-planification, planification

2.B) Vie artificielle et pression sélective

Quand nous plaçons des agents en survie sur un support, nous nous situons dans le cadre de la Vie Artificielle (VA). Alors la pression sélective augmente, et, pour des raisons d'efficacité, nous devons élaborer des agents plus adaptés. La solution dans cette direction est donc de ne plus évacuer le problème de priorité de déclenchement de règles, mais de s'attacher à le traiter. Il faut effectuer plus de tests afin d'agir plus causalement.

Ainsi cette problématique nous amène progressivement à des démarches de proto-planification et de planification.

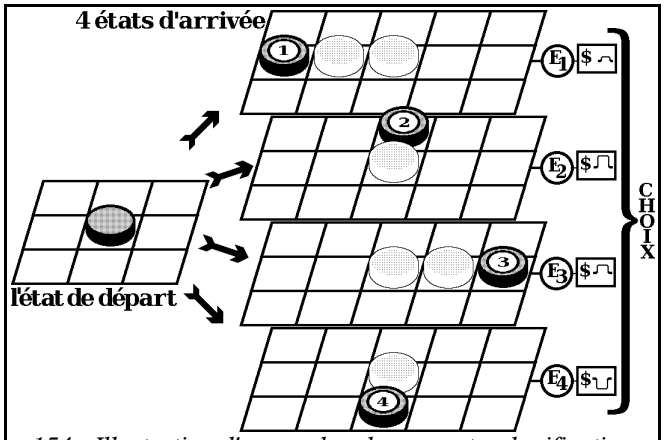
2.C) Solution partielle : l'agent Croyances Désirs utilisé en proto-planification

L'agent CD utilisé en proto-planification permet un traitement partiel du problème. Partant de l'état initial il recense toutes les instructions déclenchables, et il teste systématiquement toutes les simulations dans chacune des directions induites par de ces premières instructions. Voici cet algorithme plus détaillé :

Partant de Mt_0 : l'état initial du monde à t_0, il itère sur chacune des instructions déclenchables :
- Convenons d'appeler I_i chacune des instructions déclenchables au départ.
- Temps 1 : exécuter I_i, et nommer Mt_{i1} l'état du monde calculé à la date t+1,
- Temps 2 : en partant de ce premier résultat Mt_{i1}, effectuer une simulation à profondeur *n* pour obtenir $Mt_{i\,(n+1)}$.
- Temps 3 : évaluer l'intérêt de cette direction en comparaison avec les précédentes. Mémoriser le résultat.

Ces tests peuvent être effectués au moyen de différents critères. Le plus simple est de chercher si un des états de la séquence atteint l'état but. Mais nous pouvons envisager des réponses quantifiées (retourner le nombre d'instructions nécessaires pour atteindre le but, leur coût, le bénéfice qu'elles ont produit...).

Remarque : il faut noter que l'agent effectue seulement ce test partiellement. En effet, après la première instruction, il reprend ensuite sur la profondeur *n*, la méthode heuristique précédente des agents réactifs.



154 - Illustration d'une recherche en proto-planification effectuée selon 4 directions initiales

Par exemple, dans l'illustration de la page précédente, partant de l'état 0, il évalue bien les trois stratégies possibles (1, 4 ou 7), mais ces simulations (1-2-3, 4-5-6 et 7-8-9) sont effectués au moyen d'agents réactifs.

Par exemple, dans l'illustration ci-contre, le pion part depuis le centre de la grille. La recherche s'effectue en directions des quatre points cardinaux et évalue bien les quatre stratégies possibles (ouest, nord, est, sud). Le test dans la direction 2 rend le meilleur résultat. Finalement le choix de fixer le cap du pion au nord sera donc retenu.

2.D) Analyser cette solution partielle : l'agent CD utilisé en proto-planification

Agent *C (Croyances)*.

Évidemment, il nous faut des agents *Croyances*, car pour simuler, l'agent CD doit disposer d'une connaissance du monde.

Agent *D (Désirs)*.

Il nous faut des agents *Désirs* qui forment des buts, ces dernier constituent les états cibles que nous testons lors de la proto-planification.

Agent *I (Intentions)*.

Nous considérons que nous n'avons pas affaire à des agents *Intention* pour deux raisons. D'abord, tout au long d'une séquence, l'activité des agents est généré par un processus réactif, qui n'est donc pas intentionnel. Ensuite, juste au départ, la première action est générée par une commande fournie par les processus de choix. Elle se limite à une unique instruction exécutée un cycle de temps élémentaire. Elle ne possède pas la dimension d'une liste d'instructions, et donc encore moins celle d'un appel de sous-programme.

Agents munis du méta-regard.

Même pour les tests les plus simples, le processeur doit accéder aux données et aux règles : il doit posséder une aptitude au méta-regard.

En effet, il a besoin d'accéder aux données du traitement effectué, d'abord afin de pouvoir initialiser l'état de départ, et encore ensuite pour tester si l'état en cours calculé correspond à l'état but.

En simulation, quand il est arrivé à un état intermédiaire Mt_{ik}, puisqu'il a accès aux données, il le récupère. À ce moment il a besoin d'accéder aux règles pour pouvoir les composer ensemble, i.e. ici pour prendre une règle et l'appliquer au résultat intermédiaire et calculer l'état suivant...

2.E) Remarque conclusion

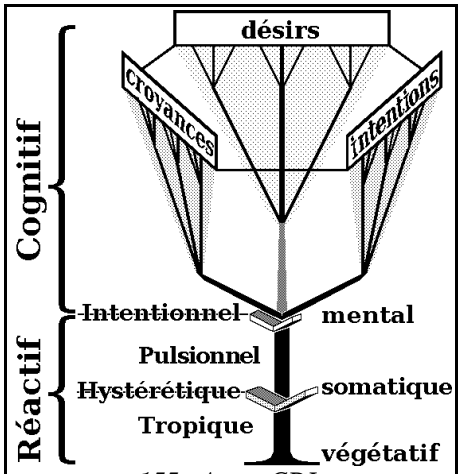
Dans notre progression pédagogique, nous avons introduit la proto-planification surtout car elle constitue une étape entre la simulation et la planification.

3) L'agent *Croyances Désirs Intentions (CDI)* utilisé en planification

3.A) Introduction

Nous continuons de placer les agents situés sur un support 2D où ils ont le choix de leurs actions. Nous nous situons dans une modélisation exigeante, par exemple dans le cadre de la VA où les êtres subissent la pression de sélection. Ainsi ils cherchent à générer un comportement très adapté : ils ne font pas de compromis et recherchent une stratégie de très bonne qualité.

Cette exigence se traduit par un nouvel algorithme : partant de l'état initial, à chaque étape, ils explorent toutes les possibilités d'action.



155 - Agent CDI
croyances + désirs + intentions

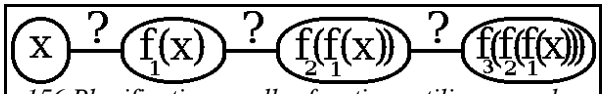
3.B) Décrivons la position du problème

Dans ce cadre, nous sommes confrontés à un travail de planification. Partant de l'état initial, le système cherche quelle séquence d'actions déclencher pour atteindre le but visé.

Dans ce fonctionnement, le système connaît l'état de départ, car c'est la donnée initiale du problème. Pour les mêmes raisons il connaît l'état du monde visé.

Par définition l'agent CDI, au moyen de ses croyances, connaît les actions que l'agent réactif modélisé peut exécuter : ce sont les règles qui constituent le comportement de l'agent.

Finalement, ce travail se ramène à un problème combinatoire : partant de l'état initial *x*, le système doit trouver la séquence d'actions $f_i(x)$ à composer afin d'accéder à l'état du monde cible.



156 Planification, quelles fonctions utiliser pour les f(i)?

3.C) Pour cette tâche nous devons disposer d'un Agent CDI

En conclusion, nous somme donc en présence d'un agent CDI :
- Croyances, car il possède un grand savoir sur le monde (support + mobilier/agent).
- Désirs, car il forme des désirs ; ici, en l'occurrence il s'agit du but visé.
- Intentions, car, à l'issue de la planification, il génère une liste d'actions à effectuer ensuite séquentiellement.

3.D) Détaillons un peu le calcul de la combinatoire d'une planification

Maintenant nous passons à une démarche quantitative, nous cherchons à dénombrer la combinatoire rencontrée dans un travail de planification.

Soit B : le facteur de branchement.

Si à chaque étape l'agent doit choisir entre *k* actions nous disons que son facteur de branchement B = k.

Soit P : la profondeur d'exploration.

Si partant de l'état initial l'agent cherche à identifier les *n* prochaines actions, nous disons que P la profondeur d'exploration à traiter vaut *n*.

Soit C : la combinatoire.

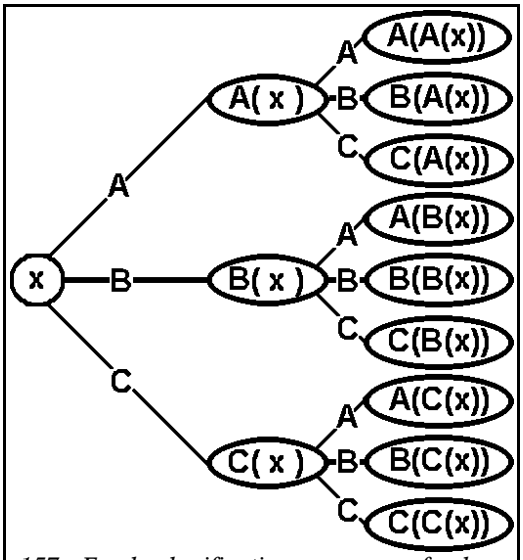
Finalement, si C est la combinatoire, nous obtenons : $C= B^P$.

Exemple 1.

La figure ci-contre illustre un exemple d'espace d'états à explorer, sur trois niveaux, depuis l'état initial x. Puisque trois choix se présentent à chaque nœud B=3. Ensuite, il faut composer deux fois les fonctions alors P=2. Finalement, la combinatoire B^P est de $3^2 = 9$.

Exemple 2.

La figure 1 au début du chapitre illustre un exemple d'espace d'états à explorer, sur quatre niveaux, depuis l'état initial 0, quand trois choix se présentent à chaque nœud. Puisque nous cherchons à composer trois fois les fonctions, la combinatoire est de $3^3 = 27$.



157 - Ex. de planification sur une profondeur de 2, avec un facteur de branchement de 3.

3.E) Exemple de planification : le problème du singe et de la banane

En matière de planification, l'exemple qu'on cite classiquement est *le problème du singe et de la banane*. Dans une salle se trouvent : un singe affamé, une table, une chaise, un bâton, et une banane accrochée au plafond par une ficelle. Le singe peut prendre, poser et déplacer les objets. Évidemment, son but est de manger la banane. Pour résoudre ce problème, il doit générer la séquence suivante :

- Placer la table sous la banane.
- Poser la chaise sur la table.
- Prendre le bâton.
- Monter sur la chaise (i.e. placer lui-même sur la chaise).
- Utiliser le bâton sur la banane pour décrocher la ficelle (placer le bâton sur la ficelle).

4) Implémenter la planification en IA symbolique classique

Le but de ce paragraphe est de montrer la lourdeur de la planification, la difficulté rencontrée pour l'implémenter (en programmation symbolique).

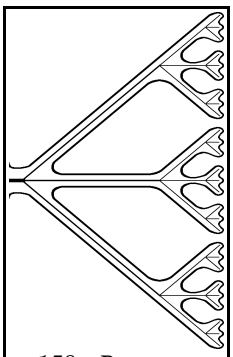
Nous sommes brutalement confrontés à des problèmes de combinatoire. Pour éviter l'explosion, nous sophistiquons le traitement qui devient complexe (à implémenter ou programmer).

4.A) Combinatoire d'une planification utilisant la force brute

Au départ, l'agent commence par le nœud initial situé à la racine de l' arbre. Pour trouver la solution, l'agent doit tester toutes les actions possibles, i.e. explorer chaque direction possible depuis ce niveau.

Puis ensuite, cette situation se représente récursivement à chaque nœud, à chaque étape de la séquence d'actions à trouver, à composer. Finalement, pour résoudre ce problème, il faut tester toutes les séquences possibles, toutes les combinaisons de B actions sur une profondeur P. Ainsi il faut traiter les combinaisons : 000, 001, 002, 010, 011, 012, 020, 021, 022, 100 ... 220, 221, 222.

Ainsi l'être doit parcourir un arbre selon une démarche en profondeur d'abord. le nombre de séquences à traiter croît exponentiellement : nous venons de chiffrer C la combinatoire : $C= B^P$.

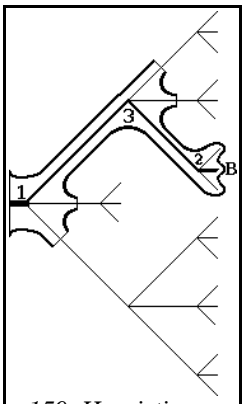


158 - Parcours d'un arbre avec B=3, et une profondeur à 3

Par exemple, dans l'illustration de droite, à chaque nœud, nous essayons 3 actions. Nous déduisons B=3. Nous y distinguons 4 niveaux, ce qui détermine 3 applications de règles depuis l'état de départ : Nous déduisons P=3. Ainsi nous pouvons constater combien la combinatoire est grande car rapidement la taille du parcours à effectuer grandit : $C= 3^3 = 27$.

L'implémentation de cette méthode est assez facile. Sa programmation est simple car elle est rustique. Cependant, ce que nous gagnons d'un côté, nous le perdons de l'autre : elle requiert la force brute de l'ordinateur, car sa combinatoire est gourmande en puissance machine.

4.B) Les heuristiques permettent d'élaguer dans les arbres



159 Heuristiques pour élaguer dans l'arbre

Parfois les domaines sont faciles à traiter. À l'issue d'un calcul le long d'une branche, nous concluons à l'impasse : il est donc inutile de poursuivre dans cette direction. Ainsi nous évitons de traiter les cas inutiles, ce qui permet d'élaguer dans l'arbre.

Parfois les domaines à traiter requièrent plus de doigté car nous connaissons seulement des résultats à priori : nous possédons des heuristiques. Ce sont des considérations empiriques permettant d'anticiper grossièrement le résultat de l'application d'une action sur le monde. Alors, quand nous nous trouvons au niveau d'une fourche, si une heuristique nous laisse entendre que presque certainement, il est inutile de l'explorer, cette considération nous permet encore d'élaguer dans l'arbre.

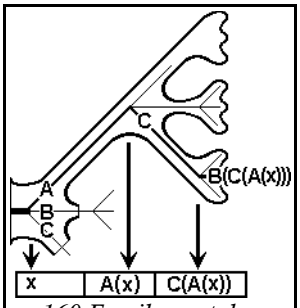
Par exemple, dans l'illustration ci-contre, au niveau de chaque fourche, des considérations heuristiques permettent d'éliminer deux directions sur trois. Il nous reste seulement une action à traiter. Ainsi nous pouvons constater combien la combinatoire est diminuée : le traitement à effectuer se réduit à l'unique parcours d'une branche principale.

Note : Il faut cependant préciser que parfois une heuristique peut nous faire couper la branche où se trouve la solution optimale. Donc l'utilisation des heuristiques nous fait quitter le domaine des tactiques (i.e. des traitements complets par définition), pour passer à celui des stratégies (i.e. des traitements incomplets par définition). Ainsi nous quittons le domaine de la démonstration mathématique (absolument parfaite), pour passer à celui de l'optimisation, qui est empirique.

a) La méthode du backtrack

La méthode du backtrack , i.e. de l'empilement des résultats intermédiaires.

Lors de l'exploration systématique d'un arbre, butter sur une impasse nous amène à redescendre d'une branche, et à repasser sur nos traces (étymologiquement : to come back on our track).



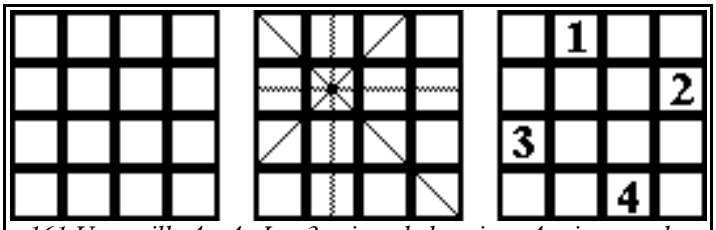
160 Empilement des résultats intermédiaires

Par exemple, dans l'arbre ci-contre, imaginons que nous venons de constater que A(C(A(x))) est une impasse. Dans ce cas, pour repartir en avant afin d'explorer une nouvelle branche B, nous devons connaître l'état du monde au nœud C(A(x)) où nous sommes. Mais pour ce faire, il est inutile de recalculer notre position depuis la racine de l'arbre, i.e. depuis l'état de départ du monde, et d'évaluer successivement : $x \rightarrow A(x) \rightarrow C(A(x))$. En effet, nous sommes déjà passés ici, et lors de la recherche précédente, nous l'avons déjà calculé.

Donc, la technique du backtrack consiste à parcourir un arbre en enregistrant les états intermédiaires, afin de les réutiliser. Puisque nous parcourons un arbre, dans un langage de programmation de bas niveau, ce travail s'effectue facilement au moyen d'une pile. Mais dans un langage plus macroscopique qui serait récursif, il peut s'effectuer encore plus facilement en mémorisant le résultat intermédiaire (l'état du monde) dans le contexte local de la fonction, ce qui consiste implicitement à utiliser la pile du langage et évite de créer la sienne. Pour ce faire, en C nous utiliserions une variable locale, et en Lisp nous nous servirions du *let*.

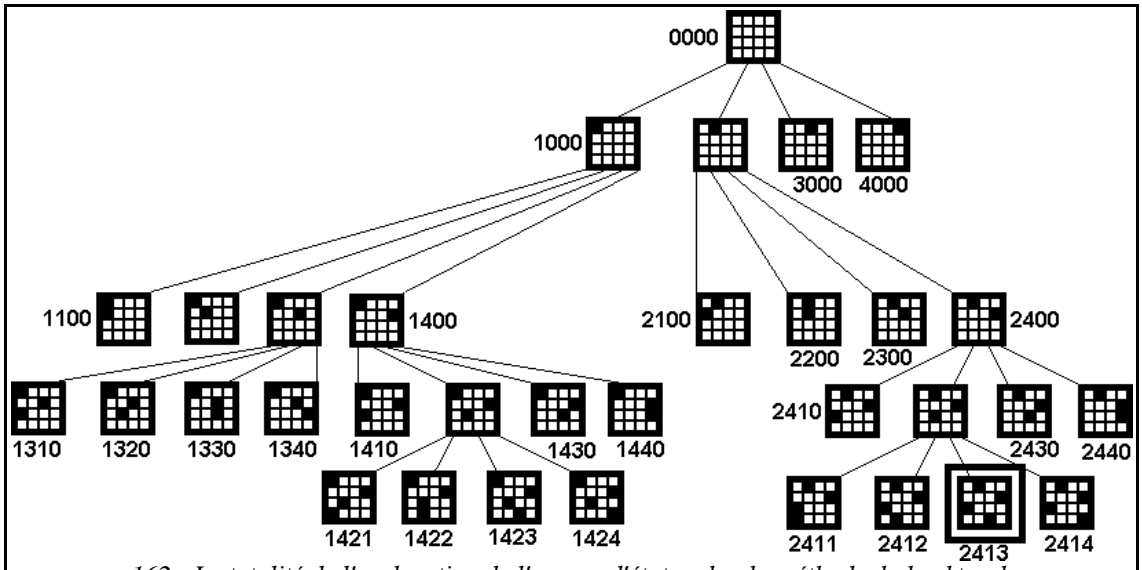
Exemple de backtrack : Les 8 reines sur un échiquier.

Maintenant nous illustrons un grand classique des cours à propos du backtrack : l'exemple des reines sur un échiquier. Pour simplifier le dessin nous prendrons seulement une grille 4x4 (ci-contre figure de gauche), mais le raisonnement est facilement généralisable.



161 Une grille 4 x 4 Les 3 prises de la reine 4 reines sur la grille

Il s'agit donc, selon les règles classiques des échecs, de disposer 4 reines sur un échiquier 4x4, de façon à ce qu'elles ne puissent pas se manger entre elles. Une des solutions est représentée, ci contre, dans la figure de droite). Rappelons qu'une reine prend à la fois comme



162 - La totalité de l'exploration de l'espace d'états selon la méthode du backtrack

un fou et une tour (représentation ci-dessus : figure du centre).

En utilisant la méthode du backtrack, nous parcourons cet espace d'états et finalement nous trouvons une des solutions : 2413. Les autres se déduisent par symétrie ou par rotation.

5) Une représentation sub-symbolique pour une planification bon marché

Dans le cadre des A6S, nous allons utiliser une représentation sub-symbolique pour implémenter facilement une planification.

5.A) Remarque à propos des êtres : agents CDI + méta-regard

Pour résoudre ce problème nous devons disposer d'agents CDI enrichis du méta-regard :
- Croyances : ils doivent avoir des connaissances sur le monde qui est ici un graphe de lieux.
- Désirs : ils ont le désir d'aller dans un lieu.
- Intentions : ils empilent les solutions intermédiaires sous la forme d'un trajet, qui correspond à un programme.

5.B) Synoptique d'un traitement pré-cognitif minimaliste

Nous décrivons ici l'implémentation minimaliste d'une démarche pré-cognitive constituée d'un planifieur facile à implémenter construit en sub-symbolique. Cette réalisation très épurée, fait intervenir plusieurs processus de base de l'IA : simulation, synthèse de comportement, méta-regard et donc planification.

a) Synthèse de comportement

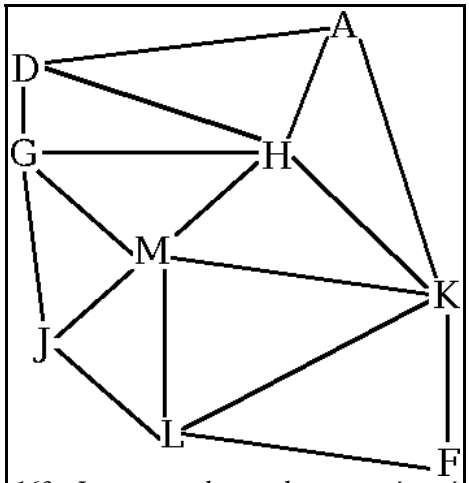
Le but de cette planification aisé à implémenter demeure la synthèse d'un plan permettant d'atteindre un but, et par là, de satisfaire un désir. Plus concrètement, disons par exemple que le besoin à satisfaire est la faim. Donc, ainsi notre tâche consiste à trouver un itinéraire jusqu'à un lieu cible : la réserve de nourriture.

b) Agent hystérétique

Nous souhaitons réaliser un agent pré-cognitif minimaliste, il ne faut pas s'étonner qu'il soit au moins hystérétique, i.e. doué de la faculté de mémorisation. Donc, pour rallier le lieu où se trouve la réserve de nourriture, nous utilisons un agent C*royances* qui possède une représentation du domaine où il vit.

Représentation symbolique.

Dans un premier temps, nous pouvons penser à une représentation de la connaissance sous forme d'un réseau de segments (voir l'exemple ci-contre). Puisqu'en partant d'un lieu, nous pouvons accéder à un ou plusieurs lieux d'arrivée, elle peut donc se coder dans une liste d'associations L qui soit de cette forme : L = (segment1 segment2 ... segmentN) ; avec chaque segment qui se décompose ainsi : (LieuDeDépart LieuDArrivée).



163 - Le support du monde est représenté par un réseau de lieux.

Finalement, nous obtienons L=
((A D) (A H) (A K)
(D A) (D G) (D H)
(F K) (F L)
(G D) (G H) (G J) (G M)
(H A) (H D) (H G) (H K) (H M)
(J G) (J L) (J M)
(K A) (K F) (K H) (K L) (K M)
(L F) (L J) (L K) (L M)
(M G) (M H) (M J) (M K) (M L))

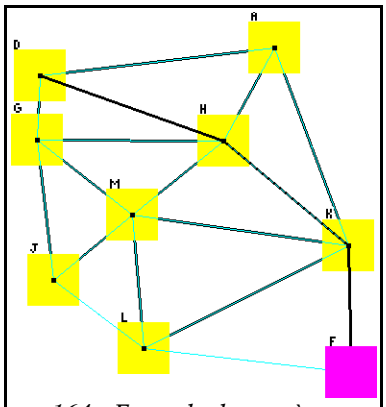
Mais cette représentation, structurée comme une liste de listes, est trop formelle, trop proche de l'IA symbolique.

Représentation sub-symbolique

Puisque nous souhaitons implémenter un traitement pré-cognitif, nous devons utiliser une représentation sub-symbolique, afin qu'elle soit plus proche des réseaux de neurones.

Soit N un nœud du réseau, et pN un pointeur sur ce nœud N. En utilisant ces conventions, nous pouvons représenter le réseau sous cette forme :

$\mathcal{R}$= {A, D, F, G, H, J, K, L, M}	Le réseau $\mathcal{R}$ est constitué d'un ensemble de nœuds
A= {pD, pH, pK}	Chacun de ces nœuds est constitué d'un ensemble
D= {pA, pG, pH}	de pointeurs sur un autre nœud.
F= {pK, pL}	
G= {pD, pH, pJ, pM}	
H={pA, pD, pG, pK, pM}	
J= {pG, pL, pM}	
K= {pA, pF, pH, pL, pM}	
L= {pF, pJ, pK, pM}	
M={pG, pH, pJ, pK, pL}	



164 - Exemple de succès : le planifieur trouve le but F

c) Planification

Notre but est de construire un planifieur facile à implémenter dans le cadre d'une représentation sub-symbolique. En conséquence, nous allons quitter le domaine de la tactique pour passer à celui de la stratégie. En effet, afin d'obtenir une circuiterie plus simple que l'algorithme du backtrack de l'IA symbolique, nous ne cherchons plus à réaliser un planifieur systématique, mais seulement un planifieur heuristique basé sur l'algorithme Monte-Carlo[52]. Le but est toujours que l'agent cherche un itinéraire, allant de son point de départ, vers un but qu'il a formé intérieurement (par exemple un lieu où se trouve de la nourriture ou de l'eau) ; mais maintenant nous ne garantissons plus une réponse parfaite en un temps donné : il peut arriver, avec une probabilité très faible, que la réponse tarde longtemps.

d) Simulation

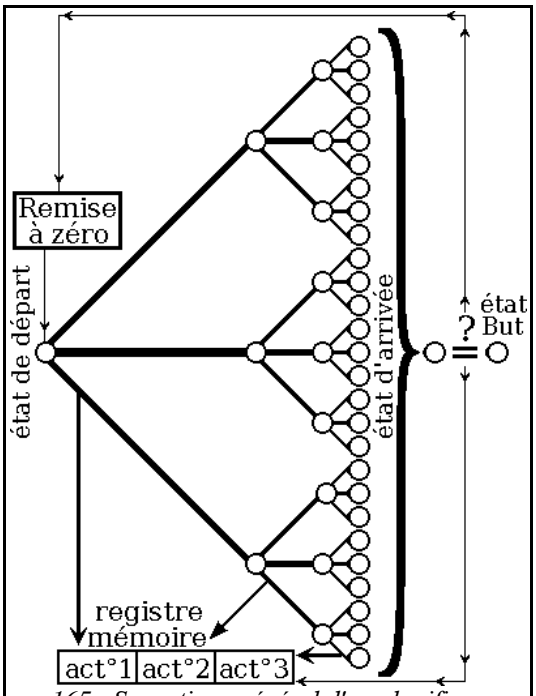
Partant d'un point de départ, l'être dispose donc d'une aptitude à la simulation pour composer N fois (dans l'illustration ci-contre N=3) l'action d'avancer le long d'un segment, afin d'arriver jusqu'à un but. La seule différence tient au fait que nous utilisons l'algorithme de Monte-Carlo : à un croisement donné, le choix du chemin emprunté se fait au hasard, et est mémorisé dans un simple registre.

e) Un méta-regard pour la condition d'arrêt

Doté d'une aptitude au méta-regard[53], l'agent est capable de s'arrêter seul. Doté d'imagination, depuis l'origine il simule un trajet de trois étapes choisies au hasard.

Si par hasard, il arrive au but, détecte cette réussite et passe dans un état *succès,* alors il préserve le registre mémoire, car ce dernier contient, indique l'itinéraire calculé.

Mais s'il erre sans trouver, au bout de 3 mouvements, l'angoisse surgit en lui : il cauchemarde, s'arrête en position *échec* et se réinitialise (remise à zéro), afin d'exécuter un nouveau cycle de recherche.



165 - Synoptique général d'un planifieur

52 La méthode Monte-Carlo : à Paris 8, plusieurs personnes ont travaillé sur ce sujet : Bernard Helmstetter, Tristan Cazenave (maintenant à Dauphine), et Jean Mehat.

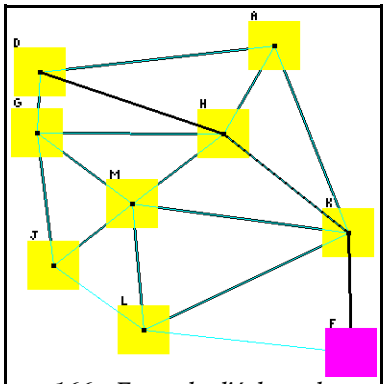
f) Remise à zéro - réinitialisation

Le système est doté d'un méta-regard qui gère et produit la réinitialisation. Il consiste en un mécanisme de remise à zéro chargé de provoquer un retour au point initial : le raisonnement est toujours capable de repartir depuis le lieu de départ.

g) Cet algorithme finit par trouver la réponse

Dans l'exemple ci-contre, à chaque cycle :
- P_S la probabilité de succès est de 1/27 = 0,037037...
- P_E la probabilité d'échec est de 1 - P_S = 0,963...

Au bout de 10 essais P_E = 0,686...
Au bout de 100 essais P_E = 0,023...
Au bout de 1000 essais P_E = 4,07 10^{-17}



166 - Exemple d'échec : le planifieur ne trouve pas le but F

5.C) Hypothèse forte ou faible

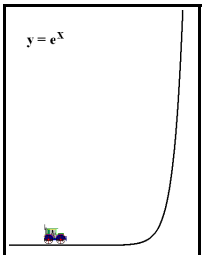
Cette modélisation d'un traitement pré-cognitif est-elle crédible, fait-elle des hypothèses fortes (est-elle tirée par les cheveux ?), ou bien est-elle plausible (fait-elle des hypothèses naturelles) ? En effet, nous pourrions critiquer le fait qu'elle utilise une mémorisation complexe où tout est remis à zéro en cas d'échec ; et surtout une fonction de génération de nombre aléatoire qui utilise la multiplication.

Pourtant, il faut bien voir ici que, pour un modèle connexionniste les hypothèses utilisées sont faibles :

- Le choix aléatoire dans les branches peut simplement être implémenté par le bruit de fond du système qui perturbe la décision effectuée par un comparateur.
- La remise à zéro du système en cas d'échec ressemble beaucoup à un être qui s'évanouit et, qui revient à lui en ayant tout oublié.

5.D) Conclusion :

Nous avons implémenté un traitement pré-cognitif très simple, au moyen d'une représentation sub-symbolique, mais nous avons indiqué que la combinatoire peut parfois être lourde.



167 - le mur de la combinatoire

53 Le méta-regard correspond à un positionnement, à un regard, un point de vue macroscopique, d'où nous regardons, auscultons et intervenons sur le processus. À ce sujet, il faut signaler l'énorme travail fait dans cette direction par Jacques Pitrat, chercheur au LIP6 (Paris6 Jussieu), auteur de *Masciste* et surtout d'un *Chercheur en IA*.

Ch. 28 La cognition chez l'agent rationnel

(Les traitements cognitifs effectués par les agents rationnels)

L'agent rationnel apparaît comme une couche méta ajoutée au-dessus de l'agent CDI (col 12)

type	*6) Pluralité : 156*	*7) formaliser l'AS*	*8) Exemples*	*9) de l'AT à l'AP*	*10)A Intensions*	*11)A Croyances*	*12) Agt. CDI*
nom	agent situé sur un support	idem	Ag. Tropique(AT)	Ag.Pulsionnel (AP)	Agents intention	Agt. Croyances	Agent CDI
exemple	2DCueilleSurUnRéseau	idem	2DOpportunistes	1DChevron	2DGroupCoop	Scouter	
Nature du cadre	indirectement orienté vers un but. Simplifie la représentation du support	modéliser l'agent situé sur un plan mathématique	Aspect tropique de l'agent situé tropique	Agent pulsionnel	Agents intention puis agent intentionnel	Agnt croyances pour simuler et anticiper	Agent CDI
transi° pr arriver ici	mettre des agents actifs ou passifs en parallèle	Généralise l'état avec variables	Utiliser l'agent de Braitenberg	Forçage somatique de mémoire	Forçage raisonné de mémoire	Apprentissage	croyances désirs intention
Nature du procès effectué	percept-décision-action = f(environnement) états & horloge externes agent tropiques indirects	Herbrand simplifie les règles/procès avec des relations agent-support avec variables	implémentation de l'aspect tropique (avec Braitenberg)	gérer plusieurs comportements	interprétation de programmes	mémorise des croyances simule et anticipe	CDI + une couche de méta = rationnel

1) Simuler un traitement rationnel cognitif sur des agents situés[54] sociaux

Dans ce chapitre nous traitons de la *cognition chez l'agent rationnel*. Pour illustrer ce propos, nous présentons un traitement rationnel cognitif effectué simplement en simulant un monde d' agent situés sociaux.

1.A) Tout d'abord définissons la cognition de l'agent rationnel

a) Présentation de la cognition classique[55]

En épistémologie, la définition de la cognition est imprécise et embrouillée :

> *La cognition est le terme scientifique pour désigner les mécanismes de la pensée. Historiquement, la cognition désignait la capacité de l'esprit humain à manipuler des concepts. Mais plus récemment, en sciences cognitives, le mot cognition est utilisé pour désigner non seulement les processus de traitement de l'information dits* de haut niveau *tels que le raisonnement, la mémoire, la prise de décision et les fonctions exécutives en général, mais aussi des processus plus élémentaires comme la perception, la motricité ainsi que les émotions alors même que traditionnellement, affectivité et intellect ont longtemps été vus comme des mécanismes opposés.*

Dans notre travail, nous ne pouvons pas utiliser cette définition fourre-tout : elle part dans toutes les directions et, pour nous, elle est inexploitable.

b) Définir la cognition rationnelle en se raccrochant au traitement de l'information

Afin de rédiger un exposé clair, nous continuons d'utiliser la méthode utilisée dans les premiers chapitres : pour nous la structure des agents est déterminante, et nous les classons selon leur architecture interne. Dans la première partie de notre exposé, nous avons déjà défini de cette façon, les agents réactifs (tropiques et pulsionnels) et les agent hystérétiques

54 Agents situés : Dans le cadre officiel de l'IAS (intelligence artificielle située) nous focaliserons sur les agents situés que personnellement nous nommons des A6S : Agents Somatiques Situés en Survie Sur un Support Simple. De plus, ici nous travaillons avec des agents sociaux.

55 Selon Wikipédia

(pulsionnels et/ou intentionnels). Maintenant, nous continuons dans cette direction et définissons le traitement rationnel cognitif par une structure.

c) Notre définition du traitement rationnel cognitif :

En nous inspirant d'une des définition de la cognition donnée par les psychologues : *an operation that affects mental contents*[56], voici la définition que nous donnons du traitement rationnel cognitif de l'information :

> *Pour modéliser un traitement rationnel cognitif facile à implémenter sans utiliser l'artillerie lourde de l'IA symbolique, nous utilisons simplement un agent CDI muni du méta-regard, et qui simule des agents réactifs.*
>
> *Dans sa structure, il implémente un système qui :*
> *- Intervient sur le jeu de règles générant le comportement des agents réactifs situés qu'il simule.*
> *- Après cette modification, évalue la pertinence du comportement de/des agent(s).*
> *- Il tire les conclusions au sujet de cette modification pour y effectuer les ajustements nécessaires.*

2) Projet : les agents situés pour effectuer un traitement rationnel cognitif

Nous souhaitons réaliser un traitement rationnel cognitif facile à implémenter sans mettre en oeuvre l'artillerie lourde de l'IA symbolique. Pour cela nous utilisons simplement un agent situé CDI, un agents situés muni du méta-regard, qui simule des agents réactifs.

Nous nous fixons ce but que nous pourrions qualifier de *traitement rationnel cognitif pauvre*, mais le travail demeure complexe. Aussi, dans les paragraphes suivants nous donnerons quelques pistes pour l'implémenter.

a) Nos limites en direction de la cognition

Ainsi nous nous limitons à cette structure et n'allons pas au-delà de ce but. Afin de réaliser une simulation très simple, nous délaissons plusieurs démarches classiques :

Pas de représentation formelle de l'IA symbolique

Nous utilisons avec parcimonie la représentation formelle de l'IA symbolique, ainsi que la machinerie informatique correspondante (registre, RAM, pile, file). En conséquence, nous ne progressons pas en direction du traitement rationnel cognitif obtenu au moyen de la représentation formelle (Agent BDI, SOAR).

Un seul niveau de méta-regard et pas d'espionnage

Nous choisissons de ne pas utiliser plusieurs niveaux méta empilés qui nous auraient permis de raisonner sur le jeu de règle (Jacques Pitrat, Sylvie Kornman). Nous n'utilisons pas la technique d'espionnage : nous ne modélisons pas un agent, qui soit capable de meta-regarder le déroulement de ses actions pilotées par son jeu de règles, afin d'analyser le résultat obtenu.

Conclusion : vers une synthèse entre l'IA formelle et le connexionisme

Non ! Toutes ces attitudes ne constituent pas notre quête. Ces voies ont été explorées par des générations de cogniticiens. Nous cherchons plutôt en direction du connexionisme muni de sa représentation sub-symbolique.

3) Situer le traitement rationnel cognitif sub-symbolique

Notre but est de présenter notre exemple de traitement rationnel cognitif effectué au moyen d'une représentation sub-symbolique. Nous souhaitons permettre au lecteur de mieux situer son positionnement original, au sein de l'IA. Mais comment allons-nous faire ?

D'un côté, nous ne pouvons pas passer sous silence les réalisations majeures de l'IA classique, mais de l'autre nous ne pouvons pas les détailler car ce n'est pas notre propos.

---

56 http://www.wordreference.com/definition/operation ; Définition 8 : process, cognitive process, mental process, operation, cognitive operation : [...] *an operation that affects mental contents* [...]

Donc, ici, nous choisissons seulement d'évoquer plusieurs réalisations cognitives classiques (basées sur une représentation symbolique formelle). Et ensuite par contraste, nous présentons notre travail.

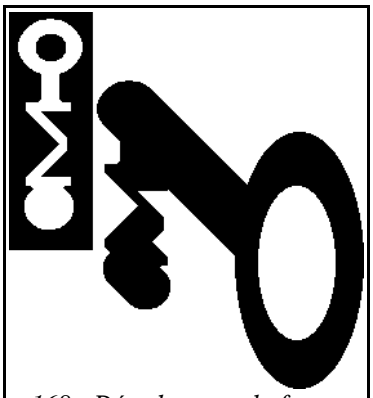
3.A) Deux orientations du traitement rationnel cognitif de l'IA symbolique

a) Lemme : préciser la définition de l'IA symbolique formelle

Avant de décrire ces deux orientations du traitement rationnel cognitif de l'IA symbolique formelle précisons tout d'abord ce qu'est cette dernière.

Elle utilise une représentation symbolique formelle des connaissances.

Ce type de représentation des connaissances (RC) est basé sur le symbole qui ensuite est assemblé sous forme d'arbre ou de liste. À l'instar de la clé qui est identifiée quand elle entre et tourne dans la serrure, l'identification des symboles est basée sur leur forme.



168 - Décodage par la forme

De plus, au sein de l'ordinateur, le symbole est indépendant du support qui le code. Il reçoit sa valeur de sa forme, mais pas des digits qui le portent. Avec cette propriété, nous débouchons sur la notion très pratique de mobilité des instances de symboles : au sein de l'ordinateur les instances de symboles sont mobiles. Par exemple, si le symbole *Paris* est dans l'adresse 5 de la mémoire, il peut très bien se dupliquer, et une de ses instances peut circuler le long du bus de données pour aller se mémoriser dans le registre accumulateur.

Conclusion : Un jeu de mécano assemblé à discrétion.

Cette RC est interconnectable sans détérioration et même sans atténuation du contenu manipulé. Cet outil de traitement de l'information peut donc être composé à discrétion par l'utilisateur. Ainsi, cette représentation est très pratique : elle procure un jeu de mécano si facile à mettre en oeuvre qu'il peut finir en usine à gaz. Par contre, elle se caractérise donc par une débauche de moyens matériels (RAM, registre, table, liste plate, liste récursive, pile, file, emboîtement de sous-programmes sur n niveaux). Pour s'en persuader il suffit de regarder la loi de Moore[57] qui pointe la croissance exponentielle de la puissance de traitement des micro-processeurs.

Dans ce cas, la simple action de tracer le fonctionnement d'une session, d'afficher le traitement qu'elle exécute produit facilement une logorrhée. En conséquence, et la chose est symptomatique, maintenant sont apparues en informatique, des commandes système qui modulent ce flux d'affichage. Par exemple sous Linux, nous pouvons citer la commande *quiet* qui s'oppose à *verbose*, et demande au noyau de limiter ses affichages.

b) Orientation 1 : la résolution de jeux digitaux en IA symbolique

Les journaux et périodiques publient des jeux (mots croisés/fléchés, casse-tête, message codés, énigmes) destinés à divertir et fidéliser leurs lecteurs. Beaucoup de ces divertissements appartiennent à la classe des jeux digitaux et se résolvent mathématiquement en cherchant la bijection adéquate entre deux ensembles. Parfois l'opération apparaît directement, par exemple, trouver la solution d'un anagramme rentre exactement dans cette classe. Mais souvent nous devons chercher une bijection entre un ensemble et une partie d'un autre. C'est par exemple le cas, au sein d'un jeu de mot croisés, du remplissage d'un mot vertical ou horizontal, à partir de sa définition.

Pour résoudre ce genre de problème, nous travaillons sur un domaine très régulier, au moyen d'une opération mathématique bien connue. Si, en informatique, nous cherchons à résoudre ce problème par la force brute de l'ordinateur, rapidement surgit le problème de la combinatoire du

57 Quand Moore à énoncé sa loi, il travaillait pour la société Untel.

traitement à effectuer. Et à ce moment il devient intéressant de le résoudre au moyen d'heuristiques et surtout en empilant plusieurs niveaux de méta-regard.

Nous tombons ici sur une informatique symbolique, qui fait des merveilles grâce à sa représentation symbolique et à la puissance de ses traitements effectués dans une mémoire de grande taille. Effectivement, elle résout des problèmes beaucoup mieux que ne le fait l'homme. Mais le lecteur voit bien ici que nous avons clairement décidé d'aller dans une autre direction car il est vain[58] de marcher sur les brisées de plusieurs générations de chercheurs. De plus nous pouvons reprocher à la résolution de problèmes de travailler sur un créneau ponctuel et de ne pas ratisser assez large car elle est déconnectée de la vie quotidienne. C'est pourquoi nous préférons porter notre attention sur les agents situés et le traitement des phénomènes langagiers.

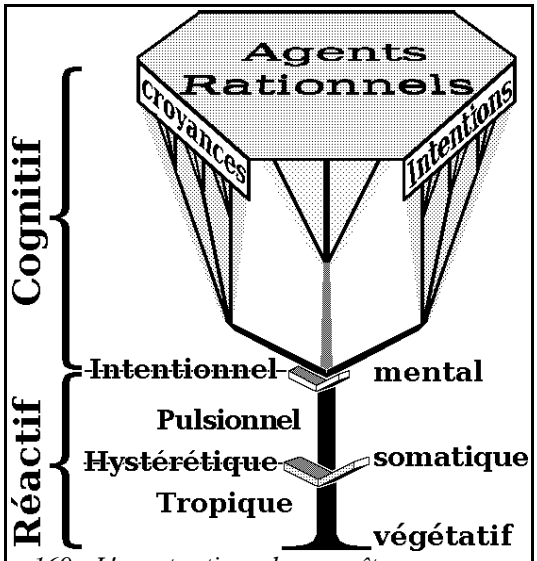
c) Orientation 2 : le projet *Soar*[59]

L'objectif principal du projet Soar est de permettre de gérer tout l'éventail des capacités d'un agent intelligent, de la routine la plus banale jusqu'aux problèmes non déterminés les plus difficiles. Pour y parvenir, selon la conception qui sous-tend Soar, il doit être capable de créer des représentations de la connaissance et d'utiliser des formes appropriées de la connaissance (procédurale, déclarative, épisodique, et éventuellement iconique). Soar accéderait alors à un ensemble de mécanismes de l'esprit.

Une autre idée sous-jacente à l'architecture Soar est qu'un système symbolique est nécessaire et suffisant pour l'intelligence générale. Cette hypothèse est connue sous le nom de système de symboles physiques *ou* système formel. *La conception de la cognition qui sous-tend Soar est liée à la théorie psychologique présentée dans l'ouvrage de Allen Newell,* Théories unifiées de la cognition.

Avec le projet *Soar*, nous sommes confrontés à un projet ambitieux, porté par toute une communauté. D'un côté, globalement, il constitue une usine à gaz *indébroussaillable*. De l'autre, les réalisations ponctuelles de l'industrie, basées sur des agents CDI sont souvent complexes et spécialisées.

Certes voici une démarche basée sur les agents, qui ratisse très large. Mais, d'abord pour tous les aspects précédents, et ensuite parce qu'elle est directement basée sur la représentation symbolique, nous ne pouvons la suivre car nous devons réaliser un document pédagogique, général, cohérent et autonome.



169 - L'agent rationnel apparaît comme une couche ajoutée au-dessus de l'agent CDI

3.B) Conclusion

Au moins, à l'occasion de ces présentations nous avons pu réaffirmer quelques unes de nos orientations : utiliser une représentation sub-symbolique, afin de proposer une réalisation autonome et cohérente, un travail simple et fédérateur qui puisse servir aux chercheurs en sciences humaines et/ou qui puisse être utilisé pour enseigner l'IA sub-symbolique à des étudiants.

58 Les idiomes *vain* et *vaniteux* possèdent la même origine étymologique.
59 D'après Wikipédia.

4) Représentation sub-symbolique pour un traitement rationnel cognitif

4.A) Notre définition du traitement rationnel cognitif

Nous voyons bien au travers de la mouvance *Soar*, que le traitement rationnel cognitif est un projet très vaste, basé sur une IA symbolique. Nous avons choisi d'éviter cette direction pour focaliser sur une représentation sub-symbolique. Dans ce cadre nous choisissons d'implémenter une cognition plus simple, que nous définissons par le traitement qu'elle effectue sur les connaissances.

Un traitement rationnel cognitif apparaît dès qu'un système raisonne sur son procès de l' information. Ceci peut se faire de façon simple, dans le cadre de la simulation : le système intervient sur les règles, initialise l'état du monde et simule son évolution ; ensuite il analyse le résultat obtenu pour recommencer à nouveau.

a) Un agent BDI, simulant des agents sociaux, pour un traitement rationnel cognitif

Nous souhaitions exhiber un exemple de traitement rationnel cognitif sub-symbolique simple. Pour le programmer, nous procédons selon une démarche éclectique, nous picorons tout azimut afin de récupérer les innovations intéressantes de plusieurs mouvances actuelles de l'IA : nous choisissons de simuler des agents situés[60] sociaux réactifs au sein d'un système, i.e. un agent CDI.

Agents réactifs.

Nous mettons au centre de notre travail un groupe d'*agents réactifs*, car c'est un outil simple : la prise de décision y est basée sur une table, une fonction logique combinatoire ou une base de règles. Toutes ces structures sont légères, et faciles à implémenter en neuronal.

Agents sociaux.

Nous incorporons dans notre travail le groupe, i.e. la mouvance importante de l'intelligence artificielle distribuée (IAD) : les systèmes multi-agents (SMA). Certes il faut implémenter un traitement parallèle des agents, mais nous récupérons l'effet de groupe et leur vie sociale.

Agents situés sur un support simple.

Nous utilisons les agent situés de l'IAS[61] : nous plaçons nos agents en situation sur un support où ils évoluent parmi les obstacles à la recherche de nourriture.

Agents en survie.

Dans le cadre général de la vie artificielle, les agents survivent sur la grille où ils évoluent. Leurs actions sont sous-tendues par le mécanisme de la sélection naturelle. Ainsi nous introduisons deux sémantiques dans le système :

- À un niveau microscopique, la sémantique de l'efficacité de l'action élémentaire réactive : l'agent mange l'énergie trouvée.
- À un niveau macroscopique, la sémantique de l'efficacité du comportement : l'agent devenu adulte se reproduit.

b) En toute rigueur, le système correspond à un agent CDI

Dans ce chapitre sur la cognition de l'agent rationnel, nous énonçons que *la cognition, c'est raisonner sur les règles.* Puisque nous décrivons un système qui effectue ce raisonnement, en toute rigueur nous concluons que la démarche cognitive obtenue est attachée tout entière à ce système.

Nous souhaitons exhiber un agents rationnel qui effectue un traitement cognitif. Donc nous devons encapsuler ce groupe d'agents réactifs au sein d'un agent CDI enrichi[62] qui simule leur

60 A6S : Agents Situés en Survie Sur un Support Simple.
61 IAS : Intelligence Artificielle Située.
62 Il est enrichi de méta-regard, et capable de méta-traitement : il modifie les règles des agents réactifs qu'il simule.

évolution au fil du temps. De cette façon, indirectement, il effectue un traitement rationnel cognitif en améliorant l'adaptation de ces agents à leur monde.

c) L'intérêt selon un point de vue informatique

L'utilisation des agents situés dans un langage de développement simplifie la tâche du programmeur.

Prenons un exemple : un agent reçoit de son capteur la perception *A*. Il dispose de deux actionneurs *B* et *C*. Comment doit-il se comporter à la vue de cette information ? Quelle règle d' action doit-il ajouter à sa base ?

Dans le cadre d'une IA symbolique classique, l'agent doit effectuer un apprentissage de l' action à exécuter. Quand apparaît cette perception *A*, il doit faire des statistiques, i.e. mémoriser l'évaluation du résultat obtenu selon qu'il exécute l'action *B* ou *C*. Finalement, si au terme de plusieurs traitements, une des solutions s'avère meilleure, il peut construire une nouvelle règle de conduite qu'il ajoute à son comportement. En conclusion, il s'agit ici d'un traitement lourd.

Mais, dans le cadre d'une IA sub-symbolique, le programmeur simplifie son travail de développement en utilisant des agents. Quand l'un d'eux, perçoit le résultat *A*, et doit choisir entre les actions *B* et *C*, alors, il lui suffit de créer au hasard, un des deux types d'agents : celui qui se comporte selon la règle *si A alors B,* ou l'autre qui applique *si nonA alors C*. Puis il le place dans le monde et il laisse l'action se dérouler...

En effet, la simulation est autonome et exécute la suite du traitement elle-même : test, sélection, disparition ou reproduction. Les agents mal programmés disparaissent, et ceux qui s'avèrent adaptés survivent, se reproduisent en propageant leur culture (et donc cette nouvelle règle qu'ils viennent d'apprendre).

4.B) La cognition chez l'agent rationnel

a) Rappel de notre définition du traitement rationnel cognitif

La cognition, c'est raisonner sur les règles. Voyons maintenant cet aspect plus en détail.

b) Raisonnement sur les règles : intervenir, appliquer, évaluer, analyser et corriger

Voici en détail du synoptique du raisonnement sur les règles que nous utilisons : intervenir, appliquer, évaluer, analyser et corriger

Intervention sur les règles :	Changer leur ordre ou leur coefficients, en ajouter ou en ôter.
Appliquer les nouvelles règles :	Les implémenter au sein d'agents que nous mettons en situation dans le monde.
Évaluation du résultat :	Évaluer indirectement l'efficacité de ces agents au moyen de la sélection naturelle.
Analyse et prise de décision :	Les mutants non viables sont supprimées par la sélection naturelle.

c) Vu la complexité des *interventions sur les règles* nous avons 2 qualités de cognition

Un traitement rationnel cognitif aisé à implémenter mais passable

D'abord, avec des interventions simples, nous obtenons une cognition facile à implémenter, mais qui est de bas niveau.

Si nous manipulons des règles au format (TêteDeRègle–QueueDeRègle), nous pouvons nous contenter de les ranger par ordre hiérarchique et de les déclencher en prospectant la liste séquentiellement. C'est la structure de l'agent *fourrageur*. Le traitement se fait facilement au moyen de gestion de tableaux ou de listes, mais nous obtenons une cognition passable.

Le fait de rajouter à ce traitement passable, une fonction pour changer l'ordre de la liste ou ôter des règles, augmente un peu la qualité de la prestation obtenue.

Un traitement rationnel cognitif de qualité acceptable

Par contre, si nous mettons plusieurs symboles dans la tête et la queue de chaque règle, si nous leur ajoutons des coefficients de déclenchement avec la gestion correspondante, si nous en fabriquons d'autres têtes ou queue de règle par panachage, le travail se complique, mais le résultat s'améliore en proportion. Nous obtenons une cognition plus acceptable.

4.C) Implémenter un exemple de ce traitement rationnel cognitif bon marché

a) Présentation d'un exemple de traitement rationnel cognitif simple

Dans le chapitre sur les *agents Croyances*, nous avons déjà présenté le programme *TransmetCulture*, utilisant ces agents qui implémentent une communication culturelle : ils y transfèrent des règles de conduite. Puis, en prolongeant ce travail, nous avons réalisé la modélisation *MemeticLamarck,* un exemple de traitement rationnel cognitif simple implémenté pour cadrer avec notre définition de ce concept.

b) Survoler rapidement le principe général de cette simulation

L'idée de base est d'utiliser une réplique des algorithmes génétiques pour mélanger non pas des gènes, mais plutôt des mèmes, i.e. des règles épigénétiques, des instructions culturelles, des consignes comportementales.

Des agents réactifs sont en situation sur un support où il cherchent de la nourriture. Quand ils se reproduisent, leurs enfants naissent immatures, car ils ignorent comment se déplacer : ils doivent donc effectuer cet apprentissage en assimilant les instructions correspondantes. Cependant, leur éducation n'est pas forcément faite par les parents : elle ressortit de la communauté tout entière.

Initialement, sur la grille, nous choisissons de placer différentes *ethnies* d'agents porteuses de règles comportementales dissemblables. Au fil du déroulement de la simulation les cultures en présence s'hybrident en mélangeant leurs instructions culturelles. Les êtres au comportement incohérent disparaissent ; mais, par contre, la pression de sélection fait apparaître de nouveaux agents au comportement plus adapté : le processus d'évolution a sélectionné chez eux un jeu de règles amélioré.

Selon un point de vue global, nous concluons avoir bien implémenté une cognition telle que nous la définissons car :
- À l'instar des algorithmes génétiques, nous avons modifié un jeu de règles.
- Au moyen de la quête de nourriture nous avons indirectement évalué en situation l'adaptation de l'agent obtenu.
- Au moyen de la vie artificielle nous avons analysé le résultat obtenu, et pris en compte cette évaluation.

c) Pour plus de précisions, voir les polycopiés donnés en annexe

Pour des raisons de clarté, nous ne développons pas ce thème ici. Le lecteur est invité à se reporter à deux polycopiés de la partie pratique, où ce thème est développé in extenso : TransmetCulture et MemeticLamarck.

4.D) Conclusions sur cet exemple de traitement rationnel cognitif bon marché ?

a) Les algorithmes génétiques (AG) constituent un domaine bien empirique

Fin de la démonstration rigoureuse.

Évidemment, quand nous travaillons avec cet exemple de cognition, nous quittons le domaine mathématique rigoureux de la démonstration. Bien sûr les AG ne constituent pas une méthode systématique puisque les fusions de patrimoine génétique se font de façon aléatoire.

Fin de l'optimisation.

Notre exemple de traitement rationnel cognitif n'est même pas de l'optimisation car nous serions bien en mal de démontrer que nous avons atteint un maximum, même local. ici, nous ne sommes même pas dans le domaine de l'optimisation.

Évolution.

Le seul résultat de cet exemple de traitement rationnel cognitif que nous pouvons garantir est que, au fil du temps, la qualité du traitement effectué par les agents évolue, progresse[63]. Donc, nous sommes dans le domaine de l'émergence et de l'évolution.

b) Les AG permettent une prospection imaginative en largeur et en profondeur

Avec cet exemple de traitement rationnel cognitif, nous sommes en présence d'une stratégie d'exploration mixte qui se développe dans deux directions : en *largeur* et en *profondeur*.

Si nous considérons l'arbre d'un être initial :

Si nous focalisons sur un agent présent à la fin du traitement (à la fin de la simulation) et regardons son ascendance, nous obtenons une structure d'arbre qui se ramifie à chaque accouplement, à mesure que nous remontons dans le temps.

En effet, à chaque reproduction, le nouvel agent intègre dans son patrimoine génétique une information provenant de ses deux parents. Puisqu'il a survécu, notre être a conservé les règles les plus adaptées, tout en se délestant des moins significatives. Ainsi, à chaque reproduction il délaisse quelques méthodes peu efficaces pour les remplacer par le même nombre de comportements plus riches.

Nous sommes en présence d'une stratégie d'exploration mixte alliant *largeur* et *profondeur*.
- Largeur car chaque méiose fusionne le patrimoine des ascendants de chaque agent, et sélectionne les méthodes efficaces.
- Profondeur, car, à chaque génération, à chaque nœud de l'arbre, les ancêtres de l'agent ont enrichit leur comportement.

Si à chaque génération, nous regardons l'ensemble des agents encore vivants.

Et si nous dessinons les fusions entre les agents, progressivement nous obtenons un dessin qui présente une structure de graphe.

Si nous regardons la simulation globalement.

Nous pouvons considérer qu'à chaque génération, le système entier, utilise le principe de réalité pour jauger les agents. Ainsi, il fait disparaître les moins adaptés et conserve les meilleurs. Mais ici, nous comprenons bien que, dans ce processus, le hasard intervient en positif et négatif. Alors nous concluons que sommes en présence d'une évaluation heuristique.

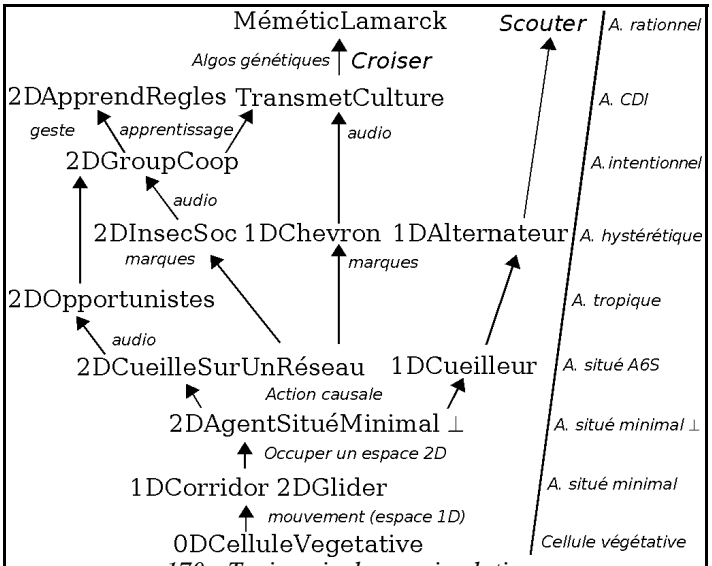
c) Conclusion

Finalement nous avons construit une traitement rationnel cognitif qui est certes facile à implémenter, mais au moins, par ce biais, nous pouvons exhiber, à l'intention des étudiants, un exemple simple de traitement rationnel cognitif bien intéressant à disséquer.

63 Et encore, il faut le dire vite. Parfois il peut arriver que les générations se mettent à cycler : A est dominée par B, B est dominée par C. Mais il s'avère que A domine C ! Alors le système boucle selon ce cycle sans spécialement progresser.

Ch. 29 Classifier des simulations d'agents situés sub-symboliques

Fournir une classification de simulations de groupes d'agents situés représentant et traitant une information sub-symbolique.



170 - Taxinomie de nos simulations

1) Problématique :

1.A) Quel est notre but ?

Notre but est de fournir aux chercheurs en IA et sciences humaines une progression de simulations basées sur des groupes d'agents situés représentant et traitant une information sub-symbolique (ci-contre).

1.B) Fournir une progression d'outils classés selon une complexité croissante

Nous fournissons aux utilisateurs plusieurs simulations. Les premières sont classées dans une progression selon une complexité croissante de leur structure et de leur programmation. Ensuite, la classification se ramifie et devient arborescente : nous obtenons une taxinomie.

Baliser par des étapes, une taxinomie qui va de la cellule végétative à la cognition.

Notre but est d'exhiber une progression qui ratisse large. Elle part de la cellule initiale pour aller jusqu'aux portes de la cognition. Puisque nous présentons seulement une douzaine d'étapes, cette évolution ne peut s'effectuer continûment, et doit donc progresser par sauts qualitatifs, que nous balisons par autant de modèles.

Une taxinomie pour palier la difficulté de la dispersion dans des directions différentes ?

Maintenant nous étudions, couche après couche, la complexité croissante du traitement de l' information chez un agent. Mais voici d'abord qu'il s'organise avec ses congénères afin de former un groupe ; ensuite ils mettent en place des démarches de communication et de transmission culturelles : à ce stade, la progression séquentielle initiale se ramifie : elle part dans des trois directions séparées. Ainsi la structure qui classifie ces étapes devient une taxinomie.

Mais quand vient le moment de communiquer ou de rédiger, nous avons besoin d'une liste de simulations successives. Pour cela, au moyen d'un *parcours en largeur d'abord* de son arbre, nous la projetons sur une droite. Ainsi, à la place de l'arborescence, nous disposons maintenant d'une progression séquentielle.

Diffuser ces modélisations, ces outils de l'IA vers une large gamme d'utilisateurs.

Finalement, ces programmes sont documentés pour fournir une boîte d'outils de l'IA, une progressions de simulations, à une large gamme d'utilisateurs qui les utilisent pour l' enseignement et la recherche : sciences humaines, sciences sociales, sciences cognitives, et aussi pour l'IA.

2) État de l'art

2.A) Notre progression de simulations s'inscrit dans l'état de l'art de l'IA

La progression de simulations que nous fournissons reprend les outils classiques de l'IA

Étape 0 : l'informatique générale.
Les premiers êtres (séquenceurs et automates) sont élémentaires : ils appartiennent plutôt à l'informatique générale. Leur code qui traite l'information perçue est centralisé dans une unique table.

Étape 1 : l'IA traite l'information perçue afin de décider de l'action à effectuer.
Les premiers modèles de cette progression sont d'abord des outils de l'informatique ou de l' IA. Ils comportent un traitement élémentaire de l'information (perception décision action) : l'être perçoit son environnement, il traite conséquemment ses percepts, afin de *décider* son action.

Étape 2 : l'IAD distribue code et donnée au sein d'objets décentralisés.
Puis viennent les outils de l'intelligence artificielle distribuée (IAD), qui distribuent le code et les données dans des entités autonomes : les objets.

Étape 3 : l'IAS s'inscrit dans l'espace et fait interagir des agents situés.
Puis ces outils se matérialisent en se situant dans l'espace : la mouvance des agents situés[64], fait interagir des êtres sur un support : elle les place en situation de cueillette dans un domaine (un graphe de lieux), où la nourriture est distribuée aléatoirement au milieu des obstacles...

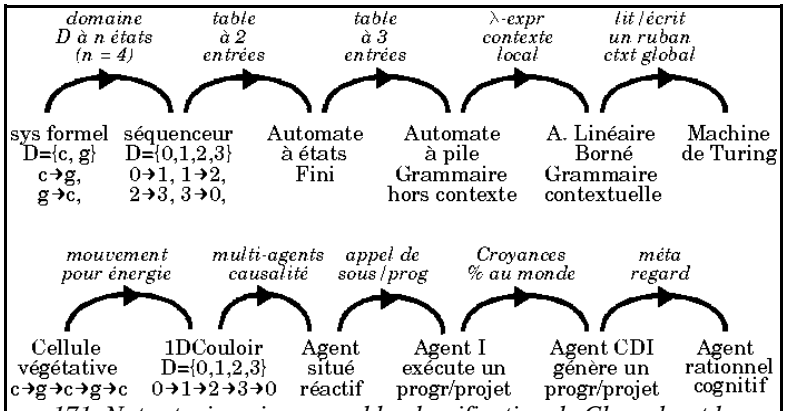
Étape 4 : introduction de la vie artificielle (VA).
Puis, cet outil met en situation des agents dont les actions sont sous-tendues par des critères de la vie artificielle. La justesse de leur comportement, de leur adaptation au milieu, est évaluée au moyen de trois sanctions différées mais abruptes :
- Critère 1 : Si un être ne trouve pas de nourriture, il jeûne ;
- Critère 2 : S'il est à la diète trop longtemps alors il meurt ;
- Critère 3 : S'il est trop faible, il ne se reproduit pas.

Étape 5 : le groupe, la communication et la cognition.
Puis, avec la reproduction des agents apparaît le groupe. Afin d'être plus efficaces les agents communiquent. Leurs représentation et traitement de l'information se complexifient : chez eux apparaît la cognition.

2.B) Notre taxinomie reprend celle de Chomsky et la prolonge vers les agents

Type 4 : le séquenceur
À la classification de Chomsky nous rajoutons le type 4 correspondant au séquenceur : son traitement de l'information est effectué par une table à une entrée.

Type 3 : l'automate fini
Ici nous reprenons l'automate à états fini de la classification de Chomsky, et nous focalisons sur sa structure : son traitement de l'information est effectué par une table à deux entrées.



171 Notre taxinomie reprend la classification de Chomsky et la prolonge

64 C'est un terme consacré, en anglais on le traduit par *situated agents*.

Aller de l'automate fini jusqu'à l'agent situé.

La partie 1 de notre travail consiste principalement à montrer comment transiter depuis l' automate fini jusqu'à l'agent situé réactif.

Montée en puissance vers l'agent CDI et son traitement cognitif.

La partie 2 de notre travail consiste principalement à rajouter des croyances, désirs et intentions à l'agent, afin de le faire monter en puissance vers l'agent CDI et modéliser un traitement rationnel cognitif de l'information.

3) Les concepts mis en avant

Concept : Une concentration verticale qui ratisse large.

Cette méthodologie induit une simplification drastique du travail et permet une concentration verticale des modèles. Ceci nous permet de décrire et classifier une progression de modélisations de l'IA, partant de la cellule végétative pour aller jusqu'à un modèle de traitement cognitif.

Concept : L'agent situé, ce nouvel outil informatique prolonge le classique automate fini.

Les architectures des agents ainsi obtenus apparaissent comme la transcription selon le registre de l'IA, des outils informatiques classiques : l'automate fini, automate à pile, recherche par backtrack, etc. En procédant ainsi, nous obtenons des êtres simples, basiques, proches de la pureté mathématique, c'est pourquoi, ces simulations pourraient constituer des modèles élémentaires, fondateurs et généraux d'une nouvelle mouvance basée sur l'IAD, les agents situés, la VA et le connexionisme[65].

4) Critère de la progressivité pour évaluer la qualité de notre classification

Nous proposons à l'utilisateur une taxinomie. Elle est linéaire dans sa première partie, ensuite, elle devient arborescente. Afin d'évaluer sa qualité nous lui imposons d'être progressive.

4.A) Chercher un critère pour quantifier la progressivité de notre classification

Une faible distance entre chaque modèle.

Dans l'idéal, pour fournir une progression de qualité, nous devrions exhiber des modèles équidistants. Mais la chose ne se commande pas, et nous veillons seulement à présenter des simulation assez proches les uns des autres.

Pas de *missing link* (pas de fossé infranchissable).

Donc, il est acquis que la distance entre les modèles n'est pas forcément constante. Cependant le critère le plus important est de *ne pas présenter de fossé infranchissable* (pas de *missing link*). Ainsi, même si nous réussissons la plupart du temps à trouver des modèles proches, la qualité de la progression serait totalement altérée si à un moment apparaissait un grand trou, un grand *gap*, que nous ne saurions combler.

4.B) Traduction mathématique de critère de progressivité

Si nous disposions d'un critère pour mesurer la complexité d'une simulation, il nous faciliterait l'évaluation de la qualité de notre progression.

Pour une progression de N+1 modèles, et une distance d_i entre chaque modèle, afin de calculer un critère de qualité *C,* nous pouvons utiliser la forme suivante, i.e. sommer le carré des distances entre chaque intervalle : $C = \Sigma_{i=1}^{n} d_i^2$.

65 Le connexionnisme à cause de sa représentation sub-symbolique.

De plus, notre but est de le minimiser ce critère. Ainsi, C_{min} la valeur minimale de C est atteinte quand l'évaluation des simulations se répartissent de manière équidistante. Dans ce cas on calcule $C_{min} = \Sigma_{i=1}^{n} [(D_n-d_1)/n]^2$.

5) Quelles démarches pour fournir une classification de simulations

Notre démarche consiste à utiliser des critères de la vie artificielle (VA), pour sélectionner des outils de l'intelligence artificielle distribuée (IAD), i.e. programmes, simulations, modélisations basées sur des agents. Ensuite nous les simplifions et les classons, afin d'obtenir une progression de modèles très dépouillés et pratiques, que nous proposons aux utilisateurs.

5.A) Simplifier, pour épurer la structure, trouver de nouveaux outils universels

Nous prospectons ce domaine à la recherche de quelque simulation typique. Nous révisons chaque simulation intéressante afin de la simplifier : MISS (make it simple, stupid). Nous travaillons à épurer le modèle en y ôtant le superflu, afin de n'y conserver que la quintessence : la structure caractéristique.

5.B) Une représentation et un traitement sub-symboliques de l'information

Éclairer les travaux de nos aînés à la lumière des agents et du sub-symbolique.

Dans le cadre de ce travail concernant le langage, la vie artificielle, l'informatique, l'IA et l'IAD, nous nous positionnons en regard des travaux de nos aînés, qui apparaissent très dispersés. Notre démarche consisterait à reconnecter toutes ces mouvances en utilisant le fil conducteur des agents situés, qui représentent et traitent une information sub-symbolique, puis ensuite communiquent au sein de leur groupe.

a) Méthode pragmatique : nous choisissons une démarche concrète et opportuniste.

Notre méthode autorise l'utilisation de cas particuliers à condition qu'ils fonctionnent. Seul l'être qui survit a raison, les absents ont tort.

Dans un démarche opportuniste, nous construisons pièce à pièce des cas particuliers de domaines, pour que l'action y soit facilitée, afin que les agents situés puisse y mener une vie artificielle. Par exemple, dans *Chevron*, nous construisons un support assez sophistiqué afin que les êtres qui y vivent soient très simples.

b) Critère important : chaque agent doit utiliser une représentation sub-symbolique

Éclater les grandes tables des automates vers des agents munis de règles sub-symboliques

La contrainte de la combinatoire nous amène à décentraliser le code global vers des agents locaux. Quand nous plaçons l'agent dans un monde 2D, la taille de son domaine, i.e. T le nombre de lieux de son support, augmente rapidement : il est de l'ordre de $T = C^2$, où C est la taille du côté de la grille. En conséquence l'automate initial se complexifie encore plus car la taille de sa table est $TxT = C^2 \times C^2 = C^4$, le carré de T, cardinal de son domaine.

Le travail d'implémentation de l'agent doit utiliser un domaine digital et régulier. Ainsi, le programmeur qui place un agent en situation sur le support, peut confondre les lieux et son traitement de l'information peut être généralisé.

Ainsi, il peut décentraliser le code, i.e. décomposer l'automate complexe du système global, en simples agents délocalisés. Les grandes tables à 2N ou 3N entrées des automates doivent êtres éclatées vers N agents indépendants à 2 ou 3 entrées.

Le programmeur doit utiliser une représentation sub-symbolique pour simuler le domaine

- Le support du domaine se réduit à un graphe de lieux. Chaque lieu ne reçoit pas de nom et pointe sur ses voisins.
- Le mobilier du domaine se réduit à une, ou quelques sortes de matières (sol, eau, nourriture, énergie, obstacle).

- Une unique relation *lieu* ↔ *matière* : chaque lieu est vide, ou contient au maximum une seule unité d'une seule matière.

L'agent doit utiliser une info sub-symbolique pour représenter et traiter ses perceptions

Dans chaque simulation, l'agent, perçoit son environnement, au moyen de ses capteurs, i.e. les cases du domaine et leur contenu. Pour effectuer cela, il doit seulement utiliser une représentation et un traitement sub-symbolique de sa perception.

Modularité des agents pulsionnels

De plus le programmeur doit repérer chaque être au fonctionnement modulaire, et décomposer son synoptique en modules. Ensuite, il l'implémente au moyen d'un agent pulsionnel, i.e. d'un agent capable de satisfaire différents buts qui apparaîtraient l'un après l'autre.

Afin de réaliser ce fonctionnement, d'abord il monte en parallèle plusieurs agents tropiques, capables chacun de satisfaire un but dédié. Ensuite, il les chapeaute par une hiérarchie qui au moment voulu, active le module qui s'avère nécessaire : il utilise un câblage hiérarchique qui valide le fonctionnement de l'agent tropique adéquat.

5.C) Méthodes de la VA pour évaluer une organisation mise en situation

Avant d'introduire une simulation dans la progression, il faut encore évaluer l'être, le groupe ou l'organisation du groupe mis en situation dans la modélisation candidate. Mais la question surgit immédiatement : dans le cadre d'une représentation sub-symbolique, comment effectuer cette évaluation, quand nous disposons seulement du schéma de son organisation, ou du listing de ses règles de fonctionnement ?

Ici, la solution vient de la bionique : nous imitons les méthodes de la vie artificielle. Selon le principe biblique : *C'est aux fruits qu'on reconnaît l'arbre*, il suffit de procéder à une évaluation différée, i.e. de ne pas s'intéresser directement à l'action, mais plutôt à ses résultats.

Pour cela, nous implémentons dans la simulation, les trois critères centraux de la VA : réussir à trouver de l'énergie, ne pas mourir de faim, et être le plus fort pour se reproduire. Ensuite, nous prolongeons la démarche, nous continuos à l'abstraire pour la décliner selon de multiples registres :

- Pour favoriser la culture, nous disons que, la réserve de nourriture est dans tel lieu, et que l'être doit savoir y aller.
- Afin d'encourager la planification, nous demandons à l'agent de générer un itinéraire jusqu'à un lieu éloigné.

En conclusion, simuler ces lois de la vie artificielle fournit directement au programmeur sub-symbolique, un outil simple pour évaluer l'adaptation d'une organisation à son milieu. De plus, pendant le déroulement de la simulation, il est facile d'afficher l'évolution de ces paramètres au fil des cycles de vie.

5.D) Construire en couches concentriques pour obtenir une classification progressive

Puisque nous cherchons à construire notre progression en couches concentriques, nous utilisons deux démarches :

a) Construire à la frontière

D'abord nous cherchons à placer de nouvelles simulations à la frontière, i.e. en position de feuille, au bout d'une branche de notre arborescence (communication, apprentissage, cognition). Dans ce cas, les nouveaux agents mis en situation héritent de l'architecture de leur prédécesseur et reçoivent une unique fonction supplémentaire.

b) Sinon, baliser le chemin en direction des novateurs

Mais il arrive que nous trouvions une simulation intéressante qui se classe bien au-delà de notre taxinomie et crée un *gap*. Dans ce cas, les nouveaux agents mis en situation diffèrent assez de l'architecture de leur prédécesseur car ils présentent plusieurs fonctions supplémentaires.

Alors, afin de pouvoir la rajouter dans notre arborescence, nous lui cherchons des simulations intermédiaires, afin de baliser le chemin entre elle et une feuille de notre arbre.

6) Quels résultats avons-nous obtenus ?

6.A) Le premier résultat consiste à fournir une classification arborescente

Nous fournissons à l'utilisateur une taxinomie, un classement arborescent de simulations modélisant des phénomènes cognitifs, de communication, et langagiers.

6.B) La taxinomie présente un tronc linéaire et une triple fourche

a) Le tronc linéaire

D'abord, dans un premier temps, depuis notre base initiale, la cellule végétative, nous partons en parallèle dans plusieurs directions : le temps, une dimension, deux dimensions, la nourriture, les obstacles... Mais, au moyen d'un parcours en largeur d'abord, nous construisons notre progression, en incorporant, étape après étape plusieurs grandeurs physiques : ainsi nous faisons apparaître le tronc linéaire : une progression, en couches concentriques, jusqu'à l'agent situé.

b) Une triple fourche

Dans un second temps, depuis l'agent situé, nous partons en parallèle dans trois directions : l'apprentissage, la communication et le traitement rationnel... et nous développons ces branches selon ces trois directions qui s'avèrent plus ou moins indépendantes.

7) Nous obtenons une *émergence déterministe*

Construisant progressivement l'agent situé nous obtenons une émergence déterministe

Quand nous construisons la base de cette taxinomie, i.e. le bas de son tronc, la structure du monde détermine la construction des agent, et paradoxalement, nous obtenons une construction, une émergence déterministe de leur structure.

À la racine de l'arbre, la construction de l'agent situé est déterministe

Nous partons de la cellule végétative en relation de lieu avec son support unique : *sur(Cellule, Support)*.

Dès que dans la modélisation, nous introduisons l'espace sur la forme d'un ou plusieurs autres lieux, nous introduisons implicitement les premiers verbes d'action : changer de place, avancer, reculer...

Alors que nous sommes dans un processus d'émergence qui devrait être aléatoire, l'introduction de ces verbes devient un nécessité : elle est donc déterministe.

Au niveau de la fourche, l'introduction progressive rend déterministe la construction

Au niveau de la fourche, nous introduisons progressivement les composantes physiques. Cette démarche rend déterministe le parcours en largeur d'abord.

D'abord nous commençons dans le cadre très contraint de l'agent situé. Ensuite, nous programmons des exemples concrets et réels, donc nous sommes en plus dans un cadre instancié, i.e. plus déterministe.

Nous introduisons très progressivement les briques qui servent à construire le monde (mobilier+support). Ainsi, nous introduisons progressivement l'espace : une dimension sans obstacle, puis avec obstacles ; deux dimensions sans obstacle, puis avec obstacles ; sans changement de direction, puis avec changement...

En introduisant progressivement les composants : capteurs, actionneur et fonctions de traitement de l'information aux agents, nous obtenons une croissance très progressive, que la

stratégie de parcours en largeur d'abord, nous permet de construire par étape successive, sans gap, sans trop de sauts qualitatifs.

Ainsi nous réussissons à parcourir l'arbre en largeur d'abord, en introduisant quelques étapes sur chacune des directions.

Dans ce cas nous obtenons une solution, une émergence progressive, une construction presque *déterministe*.

7.A) Résultats sur le plan concret

a) Nous fournissons un outil, un cadre

Plus concrètement, à l'intention de l'utilisateur, du développeur qui veut lui aussi effectuer le même genre de recherche, susciter la même émergence, nous fournissons un outil adapté, un simulateur très léger, écrit en C, écriture objet, très léger.

b) Nous fournissons des exemples simples et probants

Finalement, nous fournissons à l'utilisateur une taxinomie d'exemples simples et probants de phénomènes cognitifs ou langagiers. Elle prend la forme d'une classification que commence par une progression en couches concentriques, et finit par devenir arborescente.

Voici donc une liste d'exemples simples et probants de phénomènes cognitifs ou langagiers que nous allons présenter en détail :

1DCorridor, 2DGridRandFood, 2DOpportunist, Glider, MemeticLamarck, scouter, 1DCueilleur, 2DGroupCoopArt, 2DSocialInsects, LearnTraj6PPV, MemeticTransmet, 1DAlternateur, 2DCueilleSurNet, 2DGroupCoopNat, Croiser, LearnTrajet RandGrid.

0DCelluleVegetative	La cellule végétative de base (modèle théorique)
1DCorridor	Agent réactif, en mouvement dans un monde 1D, un corridor
1DCueilleur	Agent réactif, cueilleur dans un monde 1D, un corridor
2DCueilleSurNet	Agent réactif, cueilleur dans un monde 2D, sur une grille
1DAlternateur	Transition du réactif vers l'intentionnel, agent bistable dans un corridor.
2DOpportunistes	Agents opportunistes, introduction d'une communication globale passive.
2DInsecSoc	Communication locale active : l'agent utilise une hormone de stress et sort d'un cycle vicieux
2DGroupCoop	Communication globale active : recherche de nourriture par un groupe d'agents coopératifs
2DApprendRegle	Communication gestuelle : transmission de règles au moyen d'une mémoire sub-symbolique
TransmetCulture	Communication langagière : simulation de transmission mémétique dans un groupe d'agents
Anticipation	Agents hystérétiques : anticiper, vu comme projeter à profondeur unaire (modèle théorique)
ImaginePlanifie	Agents hystérétiques : planification vue comme une projection à profondeur N-aire
SystèmeCognitif	Traitement cognitif : simulation de darwinisme des idées dans un groupe d'agents
Croisement	Jalons pour le croisement de patrimoine entre deux populations d'agents
Morphogénèse	(prospective) Vers une morphogénèse d'un être multicellulaire depuis une cellule élémentaire

AFD	Automate Fini	Perte de la causalité
VegCel	Cellule végétative	Séquenceur à 1 temps, à la racine de l'arbre
Bloc	Bloc du jeu de la vie	Séquenceur à 1 temps
Clignotant	Clignotant	Séquenceur à 2 temps
Cellule-graine	Clignotant	Séquenceur à 2 temps
Séquenceur	Séquenceur	Séquenceur à N temps
		introduction du mouvement
1DCorridor	Un agent dans un couloir	mouvement+var→proto-agent + couloir n
Glider	Un agent dans un couloir	mouvement+var→proto-agent + couloir infini
		introduction de la causalité
1DCueilleur	ASitué en survie ds couloir 1D	Agent causal (fait le geste de manger)
2DGridRandFood	ASitué en survie sr grid 2D	Agent causal +(gestion du déplacement)
2DCueilleSurNet	ASitué en survie sr net	Agent causal + gère déplacement + réseau
2DOpportunist	Auditeur converge vers appeleur	Agent tropique+communication
LearnTrajet	Agent tropique+communica°	Agent tropique+communica°+ApprendIA sub symbolique
LearnTraj6PPV	Agent tropique+communica°	Agent tropique+communica°+ApprendIA sub symbolique
1DAlternateur	agent pulsionnel	Agent pulsionnel
Chevron	agent pulsionnel	Agent pulsionnel
2DGroupCoopNat	Groupe coopère ds monde naturel	Agent force drapeau+communica°
2DGroupCoopArt	Groupe coopère ds monde artificiel	Agent force drapeau+communication
MemeticTransmet	Transmission mémétique	Agent BDI+ Communication
2DSocialInsects	Communica° avec phéromones	Agent BDI + Communication→Anticipation
scouter	proto-cognition planification	Agent BDI+Proto-cogni° → planification
MemeticLamarck	Cognition de bas niveau	Agent BDI + Cognition de bas niveau
Croiser	dégrossir la cognition	Agent BDI + Communication + Cognition

8) Évaluer ce résultat : vérifier la qualité de cette progression

Une progression basée sur des outils mathématiques connus

Elle part du séquenceur et de l'automate fini qui sont des classiques en mathématiques pour l'informatique.

Une progression cohérente

Nous présentons une classification faite sur mesure pour l'utilisateur. Elle s'appuie sur un socle d'outils mathématiques connus, ainsi elle est cohérente car elle se développe en couches concentriques à partir de cette base.

9) Conclusion :

Des résultats à ne pas négliger sous prétexte qu'ils sont simples, bien au contraire

Nous invitons le lecteur à ne pas se tromper sur notre travail et à bien comprendre l'intérêt des simulations clé (TransmetCulture, 2DApprendRègles, MéméticLamarck, 2DGroupCoop, Scouter). Bien sûr, certaines simulations sont placées dans la taxinomie pour faire du remplissage afin de lisser sa progression, mais les simulations clés qui pourraient paraître simplistes, sont en fait intéressantes justement à cause de cette simplicité.

10) Perspectives

10.A) Fournir à l'utilisateur des résultats sous forme de courbes graphiques ?

Évaluer la validité de la simulation d'un groupe d'agents.

Dans un paragraphe précédent nous montrons que simuler certaines lois de la vie artificielle fournit au programmeur un outil pour évaluer l'adaptation d'un groupe d'agents à son milieu. Ainsi, pendant le déroulement de la simulation, nous pouvons afficher l'évolution de ces paramètres, au fil des cycles de vie que nous simulons.

Mais ce critère est seulement absolu, il permet d'évaluer si un être, un groupe ou une organisation est valide. Cependant, il nous faut plutôt un critère relatif, capable, au sein de la progression, de comparer entre eux, générations, couches ou outils.

Comparer l'adaptation de deux générations successives.

Afin de fournir à notre utilisateur la courbe de l'adaptation des générations successives, bien sûr nous pensons à utiliser la méthode de la vie artificielle. À l'instar de la théorie de Darwin, l'idée serait carrément d'utiliser *la lutte des espèces* ! Mais, d'abord ce genre de démarche est peu ragoûtant car élitiste ; de plus, à ces difficultés étiques vient se rajouter le problème de la quantification.

En effet, quand nous rajoutons une fonctionnalité à une famille d'agents aussi élémentaires, si ensuite, dans une simulation nous mettons en compétition ces deux générations, alors apparaît un saut qualitatif : c'est le pot de fer contre le pot de terre, c'est l'épée en fer contre l'épée de bronze. À l'issue du combat, l'armée traditionnelle est décimée.

L'effet de levier est trop fort. À l'issue de ce genre de confrontation, parfois nous obtenons un résultat abrupte, de la forme : $0 < K$. Et, avec cette méthode, tout au plus nous pouvons obtenir ce diagnostic : *Adaptation (générationN) < Adaptation (générationN+1)*. Certes nous déduisons que la nouvelle génération est mieux adaptée que la précédente, mais c'est un peu court : ces résultats qualitatifs sont inexploitables car trop contrastés.

Il nous faudra donc chercher la solution du côté d'une méthode différentielle : nous regardons la vitesse avec laquelle une nouvelle génération submerge l'ancienne. C'est là un critère, une mesure bien aléatoire qui doit être aménagée.

10.B) Évaluer la complexité d'une simulation

Au sein de la progression de simulations de groupes d'agents situés, nous recherchons un critère d'évaluation relatif et quantitatif, i.e. qui nous permette d'afficher à l'écran une comparaison quantifiée entre deux générations.

a) Quelques pistes pour évaluer le coût total d'implémentation d'une simulation

Évaluer une simulation en chiffrant son coût est difficile pour plusieurs raisons

Évaluer la complexité d'une simulation en chiffrant son coût total d'implémentation nous fournirait un critère pour positionner chaque outil au sein d'un graphisme global de la progression. Cependant, cette évaluation est difficile pour plusieurs raisons, mais voici quelques solutions à ces différents problèmes.

Évaluer la complexité de la structure du soma des agents.

D'abord, il faudrait quantifier la complexité du matériel : capteurs, actionneurs, structure porteuse... Et ensuite il faudrait trouver comment composer ce résultat avec celui mesurant la complexité du programme. Mais ces deux tâches paraissent bien difficiles. Alors une solution simple et radicale est de considérer que la structure porteuse matérielle évolue peu, et ainsi de délaisser sa contribution à la complexité totale de l'édifice.

La difficulté à évaluer la complexité du programme de décision chez l'agent.

Il faut quantifier la complexité du programme, i.e. celle du centre de décision avec laquelle l'agent traite l'information qu'il perçoit. Puisque nous imposons aux simulations de la représenter sous forme sub-symbolique, alors au sein de la progression, les agents réactifs prennent leur décision, d'abord au moyen de fonctions logiques câblées, puis ensuite en se servant de règles hiérarchiques. La solution est d'évaluer à combien de circuits logiques élémentaires (nor ou nand) ce câblage se réduit. Finalement, pour en chiffrer la complexité, de façon pratique, nous convenons d'en prendre le logarithme.

Il reste cependant une difficulté : afin de vérifier si globalement la simulation effectue ce que nous attendons d'elle, l'implémentation de cette circuiterie est faite par programmation, et la couche sub-symbolique de décision n'est pas directement simulée sous forme de composants électrique. Son fonctionnement est seulement émulé dans un programme en code C. Dans ces conditions, séparer la couche sub-symbolique de la simulation de la structure s'avère difficile.

La difficulté à évaluer parallèlement la complexité du groupe d'agents.

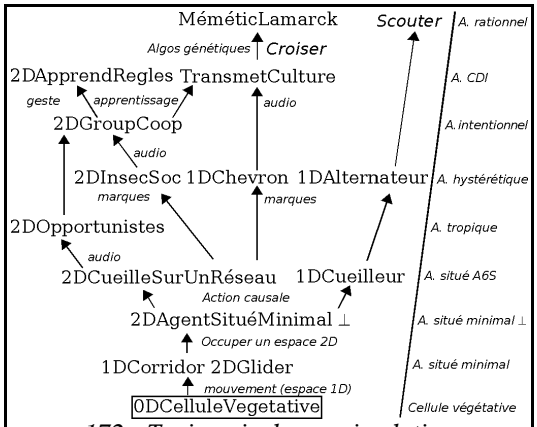
Après avoir chiffré la complexité *soma+psyché* de l'agent, comment quantifier en plus le fait que les agents s'organisent selon un groupe de N personnes ? Empiriquement, il apparaît possible de multiplier la complexité d'un agent par N. La solution peut venir des logarithmes : pourquoi ne pas ajouter la complexité du groupe à celle de l'agent en lui ajoutant le logarithme de N.

Évaluer la complexité de la mémoire de la même façon que pour celle du groupe

Quand nous sommes en présence d'agents Croyances, nous devons en plus évaluer la complexité de la mémoire qu'il utilisent pour enregistrer leurs savoirs. Soit le cas d'une mémoire de M éléments, nous pouvons procéder comme pour celle du groupe, et additionner au résultat obtenu le logarithme de M.

Ch. 30 *0DCelluleVégétative* - La cellule végétative (CV)

1) Introduction de la cellule végétative



172 - Taxinomie de nos simulation

Le modèle de départ de notre taxinomie.

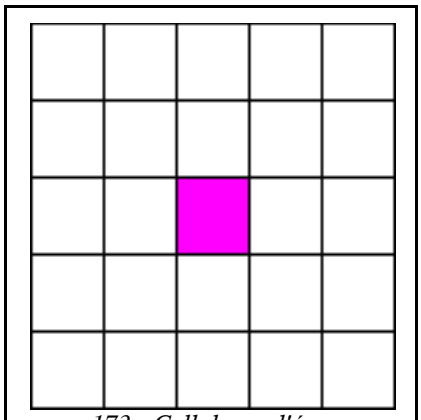
La première étape de notre classification est constituée d'une entité un peu à part. Selon le point de vue du langage commun, elle se nomme *la cellule végétative initiale*. Selon un point de vue informatique, elle correspond à un être auto-réplicateur vivant dans un espace 0D.

Situer cette étape dans notre taxinomie.

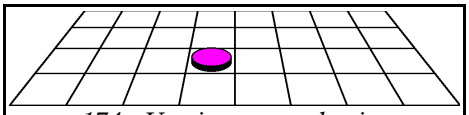
Cette cellule végétative constitue le point d'ancrage de notre taxinomie. Ainsi elle est positionnée à la racine de notre classification (Fig 1). Elle nous permet de poser les fondations de notre travail, c'est pourquoi nous la positionnons en pierre angulaire de notre classement.

Plus concrètement, il faut préciser que la cellule végétative n'est pas directement simulée au sein d'un modèle de notre progression. Par contre, elle constitue le cœur de tous les agents au sein de chacune des simulations.

2) Outils, méthodes pour implémenter cette CV


173 - Cellule sur l'écran

Modélisation informatique : un pixel.


174 - Un pion sur un damier

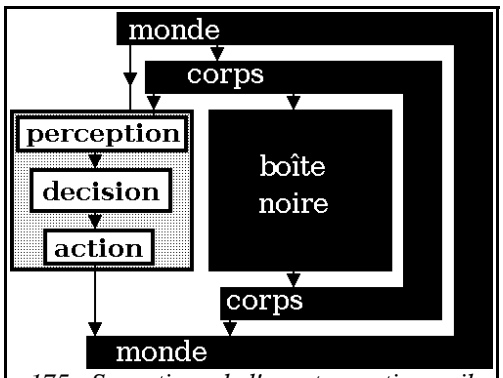
Sur l'écran de l'ordinateur, cette cellule est modélisée par un pixel, car c'est le composant qui se rapproche le plus du point de la géométrie euclidienne[66].

Modélisation par un jeu digital : un pion ou un jeton.

Concrètement, dans le cadre d'un jeu digital, cette cellule est implémentée comme un jeton sur une case, un pion sur un damier (ci-contre à gauche).

Implémentation informatique : une boîte noire.

Par la suite, dans chacune des simulations de notre progression, chaque agent est modélisé par un pion ou un jeton posé sur une case. Et, cette cellule est implémentée comme une boîte noire au sein de ce dernier.



175 - Synoptique de l'agent somatique : il porte une boîte noire en son sein

66 Selon nos hypothèses, la cellule végétative est de dimension 0, ce qui correspond, selon la théorie mathématique, à un point dont la surface est nulle. Mais cette singularité la rend difficile à modéliser en informatique, alors pour l'implémenter concrètement, nous prenons le pixel, i.e. le composant qui en est le plus proche car il correspond à un *point* sur l'écran.

L'illustration ci-contre présente globalement un agent, que nous détaillerons plus tard. Cependant, nous pouvons déjà observer en son sein, la boîte noire correspondant à la cellule végétative, qui se trouve encapsulée au cœur de l'agent (ci-contre à droite).

La cellule entretient une relation avec l'énergie.

Dans notre modélisation, la cellule végétative requiert un peu d'énergie afin de vivre. Comme sa consommation est faible, elle n'a pas besoin de se déplacer et capture celle qui est diffusée dans son environnement.

Cette relation peut s'illustrer par la plante qui capte l'énergie solaire et le CO_2 au moyen de ses feuilles, ou par la cellule, dont la membrane laisse diffuser les matières nutritives en provenance du milieu aqueux dans lequel elle est plongée.

3) programmes, méthodes et implémentation

3.A) Outils informatiques utilisés pour développer les .exe de cette progression

Tous les exécutables présentés dans notre taxinomie sont développés sous Linux au moyen de la même gamme d'outils : la programmation pure est réalisée au moyen de *gcc*, *make* et *gdb*. L'affichage graphique et la gestion des interfaces sont effectués au moyen de *SDL (Simple DirectMedia Layer)*. Cette utilisation de la même gamme d'outils informatiques apporte une unité à l'ensemble de notre travail.

3.B) Programmation objet

Les bonnes habitudes demeurent.

Initialement, tous ces programmes étaient développés en langage objet (soit *c++*, soit *Java*), ainsi certaines habitudes de programmation subsistent, notamment la décentralisation du code et des données au sein d'objets. Par contre, l'héritage des structures, du code et des données n'est pas implémenté et a été simplement effectué *à la main*.

Programmation par envois de messages.

Certaines habitudes de programmation des langages objets demeurent. Ainsi, l'exécution d'un programme par envoi de message à un agent qui était de la forme :

pAgent-> FaisCeciVuCeIntEtCepCase(5, pCase);

devient :

Toi_pAgentFaisCeciVuCeIntEtCepCase(pAgent, 5, pCase);

En fait, dans ce programme en langage C, nous convenons de passer *pAgent* le pointeur sur l'agent en premier paramètre, ce qui nous permet de simuler une programmation par envois de messages, et de conserver les habitudes de programmation qui en découlent, notamment la décentralisation des données dans les objets.

3.C) Le programme d'initialisation : main.c

```
int main(int argc, char *argv[]) {
...
 {                                                          // début de screen
  Uint32 videoflags = SDL_HWSURFACE;         // Uint32 videoflags = SDL_HWSURFACE | SDL_DOUBLEBUF;
  Uint8  video_bpp= 0;
                                                            //Video mode activa° on obtient en
  SDL_Surface* screen= SDL_SetVideoMode(Width, Height, video_bpp, videoflags);//retour une surface

  if (!screen) {
    fprintf(stderr, "I can not activate video mode: %dx%d: %s\n", Width, Height, SDL_GetError());
    exit(2);
   }

  {                                                         // Init des variables globales
```

```
    InitLesCouleursRGB(screen);
                                                                    // Efface l'écran avant de
    Draw_FillRect(screen, 0, 0, Width-0, Height-0, c_white);        // dessiner la Cell de départ

    {                                                               // les initialisations
     InitGridTabCase ();
     GridTabCase_ChaineTesCases ();

     DessineLesCasesDeLaGridTabCase (screen);

     InitSwarmTabAgent ();
     DessineLesAgentsDeLaSwarmTabAgent (screen);
    }

    boucle(screen);                                                 // Appel à boucle

   }                                                                // fin des variables globales
  }                                                                 // fin de screen
  return 0;
 }                                                                  // fin de main
```

Table 5 : La partie principale du programme main.c d'initialisation et de lancement

La boucle centrale du programme :

```
void boucle(SDL_Surface * screen) { // Début de boucle
...
 {
  int i;

  for (i=0; i<9999; i++)
   {

    SDL_UpdateRect(screen, 0, 0, 0, 0);          // Post dessin, Met à jour l'écran to get l'affichage on screen

    {
     int done= 0;
     int CptKeyDown= 0;                                        // int CptDefault= 0;
     while (!done) {
       SDL_Event event;                       // Une structure pour recevoir la description de l'évènement à traiter
       while (SDL_PollEvent(&event)) {
         char KeyName[50];
         strncpy(KeyName, SDL_GetKeyName(event.key.keysym.sym), 50);
         switch (event.type) {
           case SDL_KEYDOWN :
            fprintf(stderr, "\nDEBUT CASE SDL_KEYDOWN %s\n", KeyName);
            switch (*KeyName) {
              case 'q' :                                     // Traite la clé 'q' pour 'quit'
               i= 99999;                                     // Pr sortir immédiatement de la boucle for
               break;
              case 'u' :                                     // Traite la clé 'n' pour 'next'
               TraiteUnCycleACetteDateCumCeScreen(i, screen);
               break;

              case 'd' :                                     // Traite la clé 'd' pour dix 10 ¹
               { int j;
                for (j=0; j<10; j++, i++) {
                 TraiteUnCycleACetteDateCumCeScreen(i, screen);
                }
               }
               Print_SurStderr_LesSlotsDesAgentsDeLaSwarmTabAgent();

               break;

              case 'c' :                                     // Traite la clé 'c' pour cent 100 = 10²
               { int j;
                for (j=0; j<100; j++, i++) {
                 TraiteUnCycleACetteDateCumCeScreen(i, screen);
                }
               }
               Print_SurStderr_LesSlotsDesAgentsDeLaSwarmTabAgent();

               break;

              case 'a' :                                       // aide
              case 'A' :                                       // aide
              case 'h' :                                       // aide
              case 'H' :                                       // aide
              case '?' :                                       // aide
              default:
               fprintf(stderr,"\nUsage : main \n");
               fprintf(stderr,"'a, A, h, H, ?' Pour afficher cette aide et sortir\n");
               fprintf(stderr,"'q' Pour quitter \n");
               fprintf(stderr,"'u' Pour avancer une fois \n");
               fprintf(stderr,"'d' Pour avancer dix fois \n");
               fprintf(stderr,"'c' Pour avancer 100 fois \n");
               fprintf(stderr,"Toute autre touche affiche cette aide et sort\n");
               i= 99999;                                 // Pour sortir immédiatement de la boucle du for
               break;
             }
            done= 1;                                     // Pour force la sortie du " while (!done) "
            break;                                       // fin de case SDL_KEYDOWN
           case SDL_QUIT:                                // ici Il faut quitter
            fprintf(stderr, " case SDL_QUIT : %d\n", CptKeyDown);
```

```
            i= 9999;                                    // Pr sortir immédiatement de la boucle des for
            done = 1;
            break;
           default:
            //fprintf(stderr, " Defaut%d ", CptDefault++);    // Par exemple, les mouvements de la tortue
            break;
          } // fin de switch
        } // fin de while SDL_PollEvent(&event)
      } // fin de while (!done)
    } // fin de int done
   } // fin du for i=0
  } // fin de i
 } // fin du s/programme
```

Table 6 : La partie principale de boucle.c qui interface (clavier/souris) le programme, et fournit l'interface graphique

3.D) Modélisation informatique des relations que CV entretient avec l'énergie

a) Modélisation informatique de la consommation instantanée d'énergie

Dans toute simulation, l'énergie consommée à chaque intervalle de temps par la cellule, est modélisée facilement en utilisant la base de temps discret des modèles.

```
void Agent_Agis(Agent * pAgent) { // En dernier le dispatcheur
...
 pAgent->EnergyLevel-= ActionEnergyCost;//Chaque action de l'agent lui fait consommer de l'énergie
...
}
```

Table 7 : Implémentation de la consommation instantanée d'énergie par l'agent

Au début du code correspondant à l'action de chaque agent au cours d'un grain de temps discret, nous trouvons l'instruction : *pAgent->EnergyLevel-= ActionEnergyCost;* qui enlève à son niveau d'énergie le coût forfaitaire de l'instant d'existence accordé par la vie.

b) Modélisation de la sanction létale pour défaut de ressource d'énergie

Dans chaque simulation, après avoir ôté la consommation d'énergie des réserves de l'agent, le programme teste si le niveau létal est dépassé et si c'est le cas l'agent disparaît purement et simplement de la case où il était.

Dans le code ci-dessous, la simulation est plus fine, à chaque intervalle de temps, le système impute à l'agent une consommation d'eau et d'énergie.

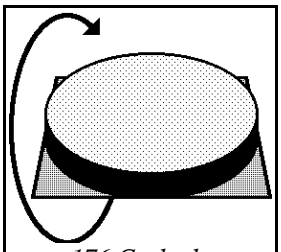
```
void Agent_AgisA7Date(Agent * pAgent, int date) { //
...
 pAgent->EnergyLevel-= ActionEnergyCost;        // each action de l'agnt lui fait consommer de l'énergie
 pAgent->EauLevel   -= ActionEauCost;           // Chaque action de l'agent, lui fait consommer de l'eau
 pAgent->Poid_SigmaFoodEau+= (pAgent->EnergyLevel+pAgent->EauLevel)/2; // each instant l'agent grandit
...
 if ((pAgent->EnergyLevel < LethalLevel) || (pAgent->EauLevel < LethalLevel)) {
  Case * pCaseOuIAm= pAgent->pCaseOuIAm;
  pCaseOuIAm->pAgent=  NULL;                    // La case ne contient plus d'agent
  fprintf(stderr, "mort\n");                    // Traitement de la mort des agents par épuisement
  return;                                       // Et on choisit de s'arrêter ici
}
```

Table 8 : Implémentation du test de l'énergie restante chez l'agent

c) Modélisation informatique de la reproduction de la CV (et de son agent)

Dans son milieu, la cellule végétative est lentement mais sûrement approvisionnée en énergie, qu'elle accumule progressivement. Ensuite, passé un certain seuil, elle meurt en produisant une graine, qui grandit sur place, et produit une nouvelle cellule : le cycle est bouclé.

Ensuite, dans le cadre de la progression de modèles informatiques, quand la cellule végétative se trouve au cœur d'un agent, elle fonctionne un peu comme une cellule souche : elle possède à elle seule tout le patrimoine génétique de l'agent et est capable de le répliquer lors de l'opération de morphogénèse.



176 Cycle de reproduction de la cellule végétative

d) Modélisation mathématique de la reproduction de la cellule (et de son agent)

En utilisant les systèmes formels et les graphes de transition, nous pouvons facilement décrire des règles de production nécessaires, et modéliser le cycle obtenu.

Dans une première approche, très simplifiée :

1) En utilisant la seule règle :
C → C | La cellule C se réplique en une cellule C,
2) En utilisant l'état de départ suivant :
C |Au départ nous disposons de la cellule C
3) nous obtenons le graphe suivant :
C → C → C → C → C ... | Ce graphe de transition est infini.

Seconde approche, plus élaborée : nous faisons apparaître une graine intermédiaire *c* :

1) Il faut utiliser deux règles :
C → c | La cellule C se réplique en une graine c,
c → C | La graine c se réplique en une cellule C,
2) En utilisant l'état de départ suivant :
C |Au départ nous disposons de la cellule C
3) nous obtenons le graphe suivant :
C → c → C → c → C ... | Ce graphe de transition est infini.

e) Implémentation informatique de la reproduction de la cellule (et de son agent)

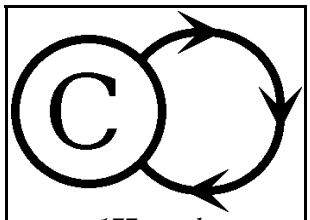
En lisant le code du programme, nous retrouvons toutes les explications précédentes :

```
..
if (pAgent->Poid_SigmaFoodEau>MaxPoidAvantEnfanter) {//si agent est costaud alors il enfante
 CreelAgentALImageDeCeluiCiEtPousseLeDansSwarmTabAgent(pAgent);
 pAgent->Poid_SigmaFoodEau= 0;
 fprintf(stderr, "CreelAgentALImageDeCelui\n");
}
...
```

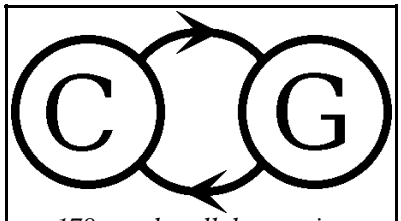
Table 9 : Implémentation de la reproduction d'un agent

4) Pas de résultat brut à présenter

Au sein de notre progression de simulations, la modélisation informatique de la première étape de la *cellule végétative* sert à introduire les bases, c'est pourquoi elle est simpliste.



177 - cycle cellule→cellule



178 - cycle cellule→graine

Nous n'avons donc pas de résultat brut, de courbe ou d'histogramme à présenter. Les seuls éléments dont nous disposons, et qui sont présentés ici, consistent en deux graphes de transition qui modélisent les deux formes de reproduction.

5) Conclusion

Pour fournir à l'utilisateur une taxinomie de qualité, il nous fallait introduire une large gamme de simulations, qui partirait de la cellule végétative, pour aller jusqu'au traitement rationnel de la cognition. À ce stade, nous venons donc d'introduire le premier élément de cette taxinomie.

En présentant la cellule végétative, nous avons décrit le fonctionnement interne de sa boîte noire, et nous avons introduit sa relation à l'énergie : comment elle mange, se reproduit et meurt. Par la suite, dans chaque simulation de notre taxinomie, nous retrouverons ce fonctionnement somatique, au sein de chaque agent modélisé.

6) Transition

Après avoir introduit cette chose abstraite qu'est la cellule végétative, quittons maintenant l'univers mathématique pour aborder des agents en situation dans le monde concret, et les communications qui en découlent.

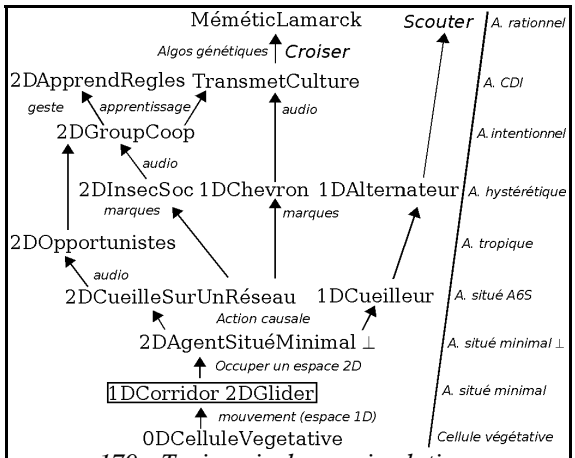
7) Perspectives

Maintenant nous allons poser cette structure végétative amorphe sur un support, une structure porteuse et la faire bouger, afin qu'elle parte à la recherche d'une nourriture plus abondante.

Ch. 31 *1DCorridor* : un domaine où l'agent agit, avance

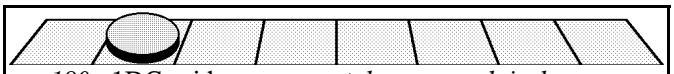
1) Situer l'étape dans la taxinomie

Après avoir décrit la cellule végétative comme base de notre progression, maintenant nous étudions *1DCorridor*, la première simulation de cette taxinomie : à cette étape, la cellule se fait plus discrète, elle reçoit un support et s'entoure d'accessoires pour s'y déplacer : elle devient l'*ASM* (*agent situé minimal*).



179 - Taxinomie de nos simulations

2) Introduire une représentation simple pour simuler


180 - 1DCorridor *: un agent dans un couloir de cases*

Ici nous introduisons une représentation simple que nous utiliserons ensuite dans nos simulations.

2.A) Représenter connaissances et actions dans les simulations de la taxinomie

De façon générale et à long terme, dans les futures simulations de notre taxinomie, le programmeur devra successivement représenter :

- Le support du mobilier. Il est fait d'un ensemble de constantes (de lieux).
- Le mobilier qui subit l'action. Dans les premières simulations, il est fait d'un ensemble de constantes (d'objets).
- Les relations entre le mobilier et son support (l'objet est sur le support, dans un lieu donné).
- L'action, comme une variation de la relation entre le mobilier et son support (l'objet change de lieu).
- L'agent qui fait l'action (l'agent déplace l'objet : il le change de lieu).

2.B) *1DCorridor* : le choix d'une représentation simple

De façon générale ce programme modélise un ASM[67] qui assimile sa nourriture en se déplaçant sur les lieux du support. Étudions la représentation de la connaissance qui permet d'implémenter facilement cette simulation.

Le support de l'action : un graphe de lieux.

En utilisant une représentation de la connaissance très simple, nous pouvons représenter le support au moyen d'un graphe de lieux. Avec des hypothèses plus contraintes (plus *digitales*), nous pouvons nous servir d'un tableau de cases.

Le mobilier qui subit l'action.

Dans cette simulation le mobilier est réduit à sa plus simple expression. Nous supposons que la nourriture est distribuée uniformément sur lui, au point qu'il ne soit pas nécessaire de la modéliser. Alors le mobilier se réduit à un unique agent.

67 ASM : agent situé minimal (le plus simple des agents situés).

Les relations entre le mobilier et son support.

La seule relation qui lie l'agent à son support est sa position, qui précise sur/dans quel lieu il est.

L'action vue comme une variation de la relation entre le mobilier et son support.

Chaque procès ou action est décrit comme une variation de la relation entre le mobilier et le support, à savoir comme le changement du lien entre l'agent et deux lieux : l'agent se déplace d'un lieu à l'autre.

L'agent qui effectue l'action.

L'agent est un être posé sur un lieu (sur un support constitué de lieux). Pour agir, il est muni d'un actionneur de mouvement qui lui permet d'aller de lieu en lieu.

L'agent est indirectement finaliste : il a été conçu par un architecte qui l'a muni d'une fonction de mouvement perpétuel, afin qu'il puisse à chaque instant assimiler l'énergie qui lui est nécessaire afin de toujours survivre. Ainsi nous sommes autorisés à dire que son procès constitue aussi une action.

2.C) Théorie : construire le domaine, l'interpréter, diverses conventions

a) Construction mathématique du domaine

Construire le domaine comme un ensemble de symboles

Selon un point de vue mathématique, d'abord nous introduisons les symboles, puis nous construisons le domaine comme l'ensemble qui les contient. Exemple 1 :

- Soit deux symboles	: A, B.	Nous introduisons deux symboles
- Soit l'ensemble D qui les contient	: D = {A, B}	Nous construisons le domaine D

Hypothèse d'une relation $\mathcal{R}$

- $\mathcal{R}$ est binaire	: $\mathcal{R}(A, B)$ encore notée $A\mathcal{R}B$	Le symbole A est relié au symbole B
- $\mathcal{R}$ est symétrique	: $\mathcal{R}(A, B) \Rightarrow \mathcal{R}(B, A)$	Si A est relié à B, alors B est relié à A
- $\mathcal{R}$ est transitive	: $\mathcal{R}(A, B) \wedge \mathcal{R}(B, C) \Rightarrow \mathcal{R}_i(A, C)$	Si A est lié à B et B à C alors A est lié indirectement à C
	: que nous notons encore $A\ \mathcal{R}_i\ C$	(il s'agit seulement d'utiliser ici une notation dite infixée)

Hypothèse d'un domaine relié

Dans chaque modélisation mathématique, nous disposons d'un domaine D, un ensemble de symboles. Puis nous ajoutons l'hypothèse que ces symboles sont tous en relation entre eux. Exemple 2 :

(H_1) chaque symbole de D, est en relation directe ou indirecte avec tous les autres symboles du domaine.

Soit le domaine D,	: $D = \{S_1, S_2 \dots S_i\}$	D est un ensemble de symboles
Convenons de notez	: $\mathcal{R}_i$	La relation indirecte entre 2 symboles.
Par exemple	: $A\ \mathcal{R}_i\ B$	A est en relation indirecte avec B
Soit c, un couple du domaine	: $\forall\ c = (S_i, S_j),\ c \in D \times D$	Avec deux symboles du domaine, nous fabriquons un couple
Nous excluons les couples réflexifs: avec $i \neq j$		Nous imposons que *i* soit différent de *j*
Nous supposons D connecté	: $S_i\mathcal{R}S_j$ ou $S_i\mathcal{R}_iS_j$	Tous les couples sont en relation directe ou indirecte

De ceci nous déduisons :

(Th 0) Les symboles sont connectés en un graphe unique.

Démonstration par l'absurde : si les symboles se répartissent en plusieurs graphes distincts, les symboles d'un graphe ne sont plus en relation avec ceux du ou des autres graphe(s) ; ce qui est contraire à H_1.

(Th1) Chaque symbole est au moins lié à un autre symbole.

Démonstration : par définition, selon *(H_1)*, chaque symbole S du domaine est soit en relation directe (*cas a*), soit indirecte avec un autre symbole S' (*cas b*).

- Dans le *cas a*, la démonstration est immédiate.

- Dans le *cas b*, S est lié indirectement à S', donc il existe S" relié directement ou indirectement à S', qui est aussi relié à S.
- En conclusion, dans les deux cas, S est relié à un autre symbole (CQFD).

Hypothèse de la restriction du domaine à une unique dimension spatiale

Puis, nous ajoutons H_2 une hypothèse restrictive, qui nous permet de modéliser D^1, un domaine à une dimension spatiale :

(H_2) Chacun des symboles utilisés dans D^1, est au plus en relation avec deux autres symboles.

Soit A,B et C éléments de D^1 :	$A \in D^1$, $B \in D^1$, $C \in D^1$	Nous prenons trois symboles d'un domaine à une dimension
Si B est lié à A et à C	: si $A \mathcal{R} B \wedge B \mathcal{R} C$	Si B est déjà lié à A et à C
alors A et C sont uniques	: alors $\neg\exists E' \in D^1 \mid E' \mathcal{R} B$: avec $E' \neq A$, $E' \neq C$,	Alors B ne peut être lié à un troisième symbole

b) Transposition : pour concrétiser le domaine, nous le plongeons dans la réalité

Ensuite nous concrétisons le domaine en le plongeant dans l'espace :
- Le traitement mathématique en cours deviens une simulation.
- Le domaine devient un support spatial.
- Les symboles deviennent des lieux.
- La relation $\mathcal{R}$ devient une relation d'accès
- Le graphe de symboles mathématiques devient un réseau des lieux physiques.

Muni de la relation $\mathcal{R}$ entre ses lieux le support spatial se représente ss la forme de graphes

Nous munissons domaine de la relation $\mathcal{R}$ entre ses lieux :

- Une relation binaire $\mathcal{R}$ entre eux: $\mathcal{R}(A, B)$ encore noté $A \mathcal{R} B$	Le lieu A permet d'accéder au lieu B : A mène à B
- La relation $\mathcal{R}$ est symétrique : $\mathcal{R}(A, B) \Rightarrow \mathcal{R}(B, A)$	Si A mène à B, B mène à A
- La relation $\mathcal{R}$ est transitive : $\mathcal{R}(A, B) \wedge \mathcal{R}(B, C) \Rightarrow \mathcal{R}(A, C)$	Si A mène à B, et B mène à C, alors A mène à C

Ainsi construit, le support spatial se représente sous la forme d'un ou de plusieurs graphes.

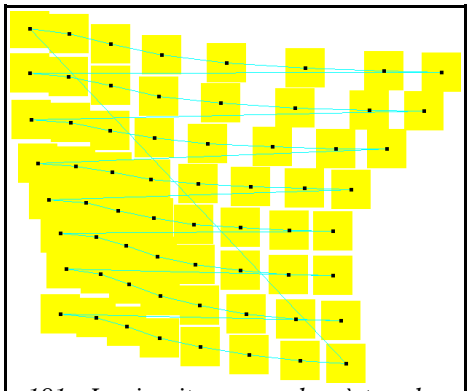
Hypothèse d'un domaine parcourable :

Nous formons l'hypothèse d'un domaine parcourable : le support spatial peut se représenter sous la forme d'un unique graphe.

Dans chaque simulation, nous disposons d'un support spatial D, un ensemble de lieux. Puis nous ajoutons une hypothèse : ces lieux sont en relation entre eux, et tous sont accessibles, i.e. chaque lieu, par transitivité, est en relation plus ou moins directe avec tous ceux du domaine. Ainsi construit, le support spatial se représente sous la forme d'un unique graphe.

1DCorridor : hypothèse d'un support restreint à une unique dimension spatiale

Puis, à ce support de lieux, nous ajoutons H_2 une hypothèse restrictive, afin de modéliser son domaine D^1 comme un domaine à une dimension spatiale. Nous modélisons *1DCorridor*, en spécifiant que chacun des lieux utilisés, est au plus en relation avec deux autres. Ainsi construit, le support spatial se représente sous la forme d'un unique graphe 1D.



181 - Le circuit : un graphe où tous les lieux sont doublement connectés

c) Introduction de diverses conventions

Convention de représentation des lieux à l'écran.

Dans les simulations de notre taxinomie, nous convenons de représenter les lieux par des cases[68] ;

68 Dans toutes les modélisations elles sont carrées, sauf dans certaines versions de la simulation 2DAprendRègles où elles sont hexagonales.

c'est pourquoi, à l'écran, dans *1DCorridor*, un lieu est représenté par un pixel ou un groupe carré de pixels, i.e. par une case.

Convention de parcours des graphes 1D

Dans ce chapitre nous construisons des graphes au moyen de la relation $\mathcal{R}$ est qui est symétrique : $\mathcal{R}(A, B) \Rightarrow \mathcal{R}(B, A)$. En effet, si A mène à B, alors B mène à A. Ainsi les graphes obtenus ne sont pas orientés et se parcourent dans les deux sens. Cependant, pour des raisons de clarté, convenons que ceux représentés dans ce chapitre sont orientés dans le sens de la lecture, i.e. de gauche à droite, et ensuite de haut en bas.

Alors, selon ces conventions, dans l'illustration ci-contre, les traits verts visualisent le chaînage entre les cases (lieux). Nous voyons bien, en fin de ligne, le passage à la suivante ; et en fin de page, le retour au début, i.e. à la première case de la première ligne.

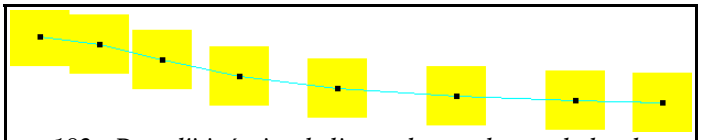
2.D) Application pratique des graphes 1D

a) Nous focalisons sur deux configurations particulières :

Dans un domaine à une dimension (hypothèse *H1*), qui soit parcourable (hypothèse *H2*), nous construisons deux configurations particulières de graphes. En fonction de nos résultat précédents, nous pouvons déjà déduire que chacune présente un graphe unique. Maintenant regardons-les de plus près :

La configuration en circuit : une boucle pour simuler un tableau infini

Dans cette topologie, les lieux sont connectés selon un graphe en forme de boucle (ci-dessus à droite).



182 - Dans l'itinéraire de lieux, chacun des symboles de départ et celui d'arrivée possède un unique voisin

Dans les modélisations à venir (*1DCueilleur*...), nous utilisons cette configuration pour simuler un espace 1D infini, au moyen d'un couloir de cases, ou d'un itinéraire de lieux, qui se reboucle en circuit fermé.

La configuration en itinéraire de *1DCorridor*

Ici les lieux sont connectés en un graphe ouvert aux deux extrémités (ci-contre à droite). Aux bouts, chacun des deux symboles (lieu de départ et lieu d'arrivée) est connecté à un unique voisin.

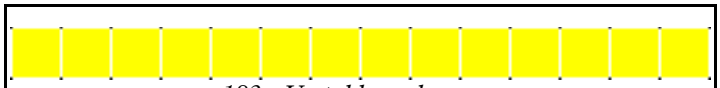
b) Les domaines particuliers utilisés pour les jeux digitaux

Pour les jeux digitaux : une hypothèse supplémentaire de régularité du support.

Si, dans la modélisation du support, nous souhaitons remplacer le réseau de lieux (cases) par le plateau d'un jeu digital (damier, échiquier...), nous pouvons rajouter, à notre modélisation précédente, l'hypothèse forte que les cases sont équidistantes et alignées.

Représenter *1DCorridor*, dans le cadre des jeux digitaux.

Si nous souhaitons convertir la configuration en itinéraire de *1DCorridor* vers le cadre des jeux digitaux, nous rajoutons des hypothèses d'alignement et de pas entre les nœuds : nous demandons que les cases utilisées soient carrées, de côté *r*, alignées horizontalement et équidistantes de *r*. Ainsi nous obtenons un tableau à une dimension (ci-dessus à droite).



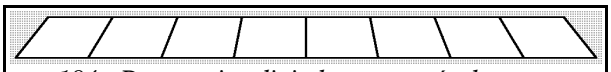
183 - Un tableau de cases

Afin d'être plus concrets, reprenons la terminologie des jeux digitaux. Quand les cases sont disposées ainsi, nous disons qu'elle sont organisées en couloir, en corridor, ou en rangée.

Remarque à propos de l'hypothèse supplémentaire de régularité introduite pour les jeux digitaux.

Nous venons de rajouter à notre modélisation initiale, une hypothèse d'alignement des nœuds et de régularité de leur pas.

Cependant nous précisons clairement que cet ajout est seulement effectué afin d'obtenir un support digital semblable à celui des jeux de plateaux.



184 - Dans un jeu digital, une rangée de cases

En effet, il n'est pas nécessaire à la mise en place de toute notre démarche, qui n'impose pas une distance ou un alignement des nœuds, i.e. qui ne requiert pas la régularité d'une grille. Elle se contente seulement d'un graphe au maillage *relativement régulier*. Ainsi, les simulations de notre taxinomie utilisent des nœuds possédant 2, 4, 6 ou n voisins.

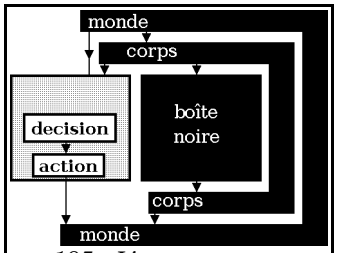
2.E) Le mobilier du monde se réduit à un ASM (agent situé minimal)

a) Modéliser la cellule en position de système végétatif (*boîte noire= cerveau reptilien*)

Le programme *1DCorridor*, modélise un être qui se déplace : une structure porteuse encapsule la cellule végétative au sein d'une *boîte noire*, et la déplace au moyen d'un actionneur de mouvement.

Introduction d'une structure porteuse pour le mouvement.

En passant de l'étape1 à *1DCorridor*, la cellule végétative se dote d'un organe dédié au mouvement, qui pousse ou tire sur une structure porteuse (membrane, carapace, squelette). Ceci induit deux conséquences :



185 - L'agent avance toujours

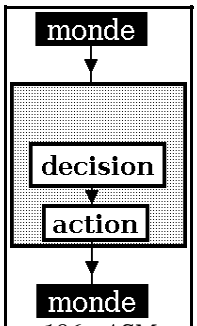
D'abord la CV se trouve encapsulée au sein d'un agent.

Elle se retrouve en position centrale au sein de cette structure porteuse qui la soutient et la protège (carapace). De plus elle conserve le fonctionnement initial que nous avons déjà détaillé au chapitre précédent. C'est ce qu'exprime le dessin de la *boîte noire*, dans le schéma de l'agent (ci-contre à droite).

Ensuite elle est dotée d'un actionneur de mouvement.

La cellule végétative, se trouve encapsulée au sein de la structure porteuse, qui est munie d'un organe capable de pousser ou de tirer, afin de la déplacer. Cet équipement correspond à l'actionneur des agents.

Comme elle baigne au milieu d'une substance nutritive qu'elle absorbe, nous convenons qu'elle avance continuellement afin d'assimiler perpétuellement de l'énergie.

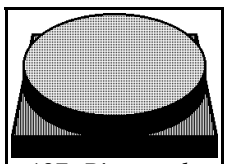


186 - ASM

b) Introduction de l'ASM (agent situé minimal)

Dans ce premier modèle, la cellule végétative, portée par sa structure, avance à chaque instant élémentaire. En conséquence, dans le synoptique de câblage de cet agent, le centre de décision se réduit au minimum : une unique règle inconditionnelle, qui active en permanence l'actionneur.

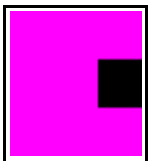
c) Représenter l'agent



187- Pion sur la case

Si l'agent se tient sur un des lieux du support. Dessiné en perspective, il peut être représenté par un pion sur une case (exemple ci-contre à gauche).

Si l'agent se tient sur un graphe de lieux (ou sur un tableau à une dimension), puisque nous représentons les lieux par des cases, à l'écran il prendra lui aussi la forme d'une case magenta ou rouge. Le petit carré noir précise la direction vers laquelle il se dirige. Dans l'exemple ci-contre à droite, il indique que l'agent marche à l'est.



188 - Agent

2.F) Représenter les relations du mobilier avec son support

Introduction.

Dans *1DCorridor*, le mobilier se réduit à l'agent, et le support à un espace 1D de lieux. La seule relation qui lie l'agent à son support est sa position, qui précise dans quel lieu il est. En représentation mathématique nous obtenons cette notation : *lieu(Agent, CaseN)*.

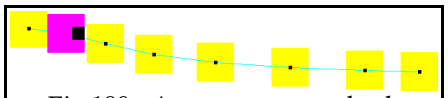


Fig 189 - Agent sur un graphe de lieux

Implémentation d'un agent dans un graphe de lieux.

Si nous restons dans l'hypothèse faible, que ces lieux sont consécutifs, mais pas forcément alignés et équidistants, alors nous pouvons représenter à l'écran le support sous forme d'un graphe de lieux et l'agent couvre une case entière.

Implémentation d'un agent dans un tableau (hypothèse du jeu digital).

Si nous faisons l'hypothèse forte des jeux digitaux, i.e. qu'en plus ces lieux sont alignés et équidistants, alors, à l'écran, nous pouvons représenter le support sous forme d'un tableau.

Si nous utilisons une représentation en perspective, nous dessinons un pion en relief sur une rangée de cases (ci-contre).

190 - Un agent dans un couloir (jeu digital en relief)

Si nous représentons l'agent à l'écran, en reprenant les conventions utilisées précédemment, nous obtenons le dessin ci-contre :

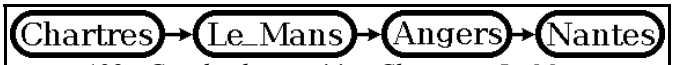
191 - Agent dans un couloir (jeu digital à l'écran)

2.G) Représenter les actions de l'agent sur le mobilier

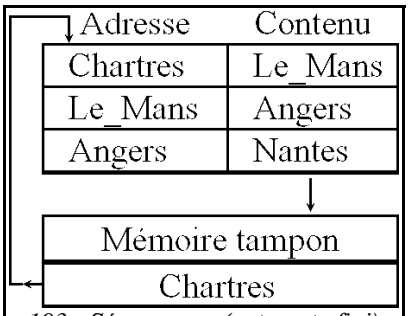
Maintenant nous allons expliquer comment, dans le cadre de *1DCorridor*, sont représentées et implémentées les actions de l'agent sur le mobilier. Puisque la nourriture est distribuée uniformément dans toutes les cases, alors nous l'oublions et même nous l'occultons : le mobilier se réduit à l'agent. Ainsi les actions à traiter se réduisent principalement aux mouvements de l'agent.

a) Implémenter le mouvement comme variation de la relation *agent* ↔ *support*

Une démarche classique en simulation consiste à représenter un déplacement par une transition d'état au sein d'un graphe de lieux.



192 - Graphe de transition Chartres – Le Mans



193 - Séquenceur (automate fini)

Cette simulation s'implémente facilement au moyen d'un automate à états finis sans ruban, i.e. au moyen d'un séquenceur (ci-contre à gauche).

Finalement, sur l'écran du programme, nous obtenons la figure de droite. L'agent est posé sur une des cases du graphe de lieux. À chaque cycle, i.e. pour chaque appui sur la touche *u*, l'agent avance d'un cran, jusqu'à la case suivante (selon les conventions du sens de la lecture).

Ainsi le mouvement est représenté comme une variation de la relation entre l'agent et le support, i.e. les lieux où il transite (ci-contre, à droite).

194 - À l'écran : l'agent dans un couloir

b) Modélisation informatique du déplacement de l'agent d'un lieu à un autre

Sans rentrer de suite dans les détails, présentons le principe général de ce déplacement. Il est implémenté au moyen de pointeurs : l'agent en possède un qui pointe sur la case où il est posé, laquelle possède un pointeur sur l'agent qu'elle contient.

Gestion des pointeurs côté agent.
Pour simuler le déplacement, il suffit de faire pointer ce pointeur de l'agent sur la case suivante, i.e. celle qui est devant lui.

Gestion des pointeurs côté case support.
Après le déplacement, la case où était l'agent, ne pointe plus sur lui mais, en général sur du vide. Mais la case qui était devant l'agent et qui vient de le recevoir pointe sur lui.

3) Détailler l'implémentation informatique de la simulation

3.A) Détailler l'implémentation informatique du domaine (le support)

a) Implémentation d'une case dans *1DCorridor*

Les déclarations de base de la case.
Selon un point de vue mathématique abstrait nous devrions seulement manipuler les états d'un graphe de transition, qui sont représentés par une constante. Mais, dans le modèle, nous introduisons un support spatial, alors ils se représentent comme des lieux. Puisque sur le plateau d'un jeu digital, un lieu est représenté par une case, alors dans *1DCorridor*, à l'écran, un lieu est représenté par un pixel ou un carré de pixel, i.e. encore une case. Voyons maintenant comme elle est implémentée selon un point de vue informatique.

```
// L'agent peut marcher sur le contenu de la case. Pour savoir si la case en contient un il faut
regarder dans sa facette pAgent

enum Nature {sol, food, arbre, eau, mur}; // Les différentes natures des cases

struct Case_ {                  // Je mets le _ ici pour ne plus avoir à le trimballer ensuite
  int CenterX;                      // Position en pixel du centre X de la case
  int CenterY;                      // Position en pixel du centre Y de la case
  int InxJ;                         // Index J de la case dans le Tableau de Cases
  int InxI;                         // Index I de la case dans le Tableau de Cases
  int CoinHauGauX;                  // Position en pixel du coin haut gauche en X de la case
  int CoinHauGauY;                  // Position en pixel du coin haut gauche en Y de la case
  int CoinBasDroX;                  // Position en pixel du coin bas droit en X de la case
  int CoinBasDroY;                  // Position en pixel du coin bas droit en Y de la case
  Uint32 Couleur;                   // Couleur de la case (en fonction de son contenu)
  int Nature;                       // Nature, contenu de la case (non utilisé ici)

  struct Agent_ * pAgent;           // Quand l'agnt est chez lui ptr sr l'agnt sinon ptr NULL
};
```

Table 10 : Implémentation d'une case (sans la partie décrivant le chaînage des cases entre elles).

Affichage d'une case.

Une case s'affiche comme un carré

```
void VuLesPosDeTesCoins_DessineTonCadre(Case * pMyCase, SDL_Surface * screen){
// Affiche un carré à partir du coin haut gauche et de la taille de son côté
 Uint32 CaseColor;                      //Crée 1 var loc pr traiter cas où il y a 1 agent
 if (pMyCase->pAgent == NULL)           // S'il y a un agent sur la case
  CaseColor= pMyCase->Couleur;          // alors affiche la case en foncti° de sa couleur
  else                                  // sinon
  CaseColor= pMyCase->pAgent->Couleur; // Affiche la couleur de l'agent sur la case


 Draw_FillRect(screen,
  pMyCase->CoinHauGauX,
  pMyCase->CoinHauGauY,
  CaseSideSize,
  CaseSideSize,
  CaseColor );
}
```

Table 11 : Affichage d'une case à l'écran.

b) Implémentation du tableau des cases, dans *1DCorridor*

Dans les simulation qui se passent sur un support de cases, elles sont centralisées dans un tableau à deux dimensions : GridTabCase.

```
#define CaseVertNbr 8 //  12 //  20 //  8       //   52
#define CaseHoriNbr 8 //  12 //  20 //  8       //   52
#define CaseNbr    64 // 144 // 400 // 64       // 2604
                                                 // Déclare un tableau et alloue
Case * GridTabCase [CaseVertNbr] [CaseHoriNbr] ; //  la place mémoire pour lui
```

Table 12 : Implémentation du tableau des cases du support.

c) Représenter le support sous forme régulière (couloir ou grille)

Le support est représenté sous forme d'un tableau 2D de cases. Quand nous faisons l'hypothèse forte qu'à l'écran, ses cases sont affichées régulièrement. Dans ce cas la position à laquelle elles sont affichées à l'écran se calculent ainsi :

```
Case * pCase_VuInxJYInxICreeYInit1Case(int IndexJ, int IndexI) {
 ...
 int posX= IndexI * CaseSideSizePlus1;     // Car à l'affichage sur l'écran,
 int posY= IndexJ * CaseSideSizePlus1;     // je sépare les cases par un espace

 pMyCase->CoinHauGauX= posX;
 pMyCase->CoinHauGauY= posY;
 ...
}
```

Table 13 : Initialisation de la position d'affichage à l'écran de la case.

Finalement, à l'affichage, nous obtenons le dessin d'un damier, d'une grille.

d) Représenter le support sous une forme souple : un itinéraire, un graphe de lieux

De façon profonde, le support est représenté sous forme d'un tableau 2D de cases. Mais, pour fonctionner *1DCorridor* peut se contenter d'hypothèses plus faibles et ne nécessite pas que ses cases soient forcément équidistantes entre elles. D'une façon moins contrainte, nous pouvons nous contenter de l'hypothèse que les cases du support, i.e. que les lieux de l'espace utilisés dans *1DCorridor*, soient seulement distribués consécutivement dans la même direction. Disons que, selon cette disposition, ils sont organisés en graphe ou en réseau à 4PPV (Chaque nœud possède quatre voisins).

Dans ce cas la position à laquelle elles sont affichées à l'écran se calcule au moyen d'une distorsion locale des coordonnées initiales au moyen d'un vecteur perturbant :

```
// Le graphe vallonné des lieux est obtenu par un déplacement de la position des cases, Ce tableau sert à
// tamponner des valeur d'atténuation d'influence des déformations, en fonction de la distance à l'épicentre.
// Pour éviter de calculer des racines de la distance entre la case et l'épicentre, la table est f (distance²)
#define AttenuationTabSize ((CaseVertNbr*CaseVertNbr) + (CaseHoriNbr*CaseHoriNbr))// Car la distance max entre
                                                 // deux cases = CaseVertNbr²+CaseHoriNbr²
int AttenuationTab [AttenuationTabSize];         // Déclare un tableau d'atténuation
#define FacteurDAttenuation1024 900              // amortit assez fort, car dans 1DCorridor, il y a peu de cases
//#define FacteurDAttenuation1024 1000           // Déclare un facteur d'atténuation sur une échelle de 1024
```

14 : Atténuer de la distorsion des coordonnées en fonction de la distance à l'épicentre de la perturbation

```
void DistordLaGrille_VuCePointJI_VuCeVecteurJI(int PointJ, int PointI, int VecteurJ, int VecteurI) {
 int j, i;
 for (j=0; j<CaseVertNbr; j++) {                // La distorsion s'opère sur toutes les cases J, I, de la grille
  for (i=0; i<CaseHoriNbr; i++) {               // et elle s'atténue en fonction de la distance à la perturbation
   Case * pCase = GridTabCase[j][i];            // Pour éviter de calculer des racines carrées, l'atténuation
   int CarreDistPointEtIci= carre(PointJ-j) + carre(PointI-i); // est f(distance)²
   int posX= pCase->CoinHauGauX + VecteurI* AttenuationTab[CarreDistPointEtIci]/1024; // Recalcule localement les
   int posY= pCase->CoinHauGauY + VecteurJ* AttenuationTab[CarreDistPointEtIci]/1024; // nouvelles posX et posY
                                                // Répercute le déplacement au sein de la case
   pCase->CoinHauGauX= posX;                    // actualise la position des composante de la case en fonction
   pCase->CoinHauGauY= posY;                    // de la nouvelle position après perturbation
   pCase->CenterX= posX + CaseSideSizeSur2;
   pCase->CenterY= posY + CaseSideSizeSur2;
   pCase->CoinBasDroX= posX + CaseSideSize;
   pCase->CoinBasDroY= posY + CaseSideSize;

   //fprintf(stderr, " Dist%d %d_%d %d.%d\n", DistPointEtIci, j, i, pCase->CenterX, pCase->CenterY);
  }
  //fprintf(stderr, "\n");
 }
}
```

Tab 15 : Calcul des nouvelles coordonnées de la case, vue la distance à l'épicentre de la perturbation.

e) Une boucle pour simuler un tableau infini

Plus précisément, le programme est capable de simuler un espace 1D infini au moyen d'un couloir de cases, ou d'un itinéraire de lieux qui se reboucle en circuit fermé. Concrètement ceci s'implémente en chaînant les cases entre elles.

```
struct Case_ {                      // Je mets le _ ici pour ne plus avoir à le trimballer ensuite
 ...
  struct Case_ * pNxtCaseNor;       // ptr sur la case voisine au nord
  struct Case_ * pNxtCaseSud;       // ptr sur la case voisine au sud
  struct Case_ * pNxtCaseEst;       // ptr sur la case voisine à l'est
  struct Case_ * pNxtCaseWst;       // ptr sur la case voisine à l'ouest
 ...
};
```

Table 16 : Implémentation d'une case (partie chaînage des cases entre elles).

```
void GridTabCase_ChaineTesCases () {                    // Chainage des cases en vecteur 1D pseudo infini
 int j, i;
                                                        // traitement de pNxtCaseEst : E S T
 for (j=0; j<CaseVertNbr; j++) {                        // Itère sur tt les cases,
  for (i=0; i<CaseHoriNbr-1; i++) {                     // mais pour i, arrête un cran avant la fin
   GridTabCase[j][i]->pNxtCaseEst= GridTabCase [j] [i+1];// Fait pointer sur la case suivante (selon i)
  }
 }

 i= CaseHoriNbr-1;                                     // Mais pour i, un cran avant la fin
 for (j=0; j<CaseVertNbr-1; j++) {                     // Alors itère sur tt les j, je fais aller à la ligne
  GridTabCase[j][i]->pNxtCaseEst= GridTabCase [j+1] [0];// Fait pointer sur la case initiale (0) (selon i)
 }

 j=CaseVertNbr-1;                                      // Mais pour i et j, un cran avant la fin
 GridTabCase[j][i]->pNxtCaseEst= GridTabCase [0] [0];  // Fait pointer sur la case initiale (0,0) (selon j, i)
}
```

Table 17 : Implémentation de la consommation instantanée d'énergie par l'agent

Nous savons que les graphes représentés dans le cadre de ce document sont orientés dans le sens de la lecture, i.e. de gauche à droite, et ensuite de haut en bas. Alors, selon ces conventions, le code ci-dessus chaîne les cases entre elles, en affectant les *pNxtCaseEst*. Nous voyons bien, en fin de ligne, le passage à la suivante ; et en fin de page, le retour à la première case de la première ligne.

3.B) Détailler l'implémentation informatique du mobilier

Déclaration et réservation du tableau d'agents.

Le mobilier, c'est par définition ce qui est posé sur les lieux (cases). Dans *1DCorridor* le mobilier se réduit à un agent. Il est contenu dans *SwarmTabAgent* le tableau des agents (de l'essaim).

```
#define AgentNbr 1                      // Déclare le nombre d'agent, ici un unique agent
Agent * SwarmTabAgent[AgentNbr] ;       // Déclare un tableau et alloue la place mémoire pour lui
```

Table 18 : Déclaration et réservation du tableau d'agents (de pointeurs sur un agent)

Initialisation du tableau d'agents.

Ensuite, le code ci-dessous initialise le premier élément du tableau en créant un agent et en mettant un pointeur sur lui dans *SwarmTabAgent* le tableau des agents.

```
void InitSwarmTabAgent () {
 int i;
 for (i=0; i<AgentNbr; i++) { // Fait pointer les pointeurs du tableau sur le ptr NULL
   SwarmTabAgent[i] = (Agent *) NULL;
 }
                //pAgent_VuInxJ_InxI_Direction_SocPos_CreeYInitlAgent(IndexJ, IndexI, Direction, SocPos)
 SwarmTabAgent[0]= pAgent_VuInxJ_InxI_Direction_SocPos_CreeYInitlAgent(     1,      1,       est, leader);
}
```

Table 19 : Initialisation du tableau d'agents (de pointeurs sur un agent)

3.C) Détailler l'implémentation des relations du mobilier ↔ support

Introduction.

Dans *1DCorridor*, le support se réduit à un espace de cases (de lieux) assemblées selon un chaînage 1D, qui se reboucle sur lui-même. La seule relation qui lie l'agent à son support est écrite de façon redondante dans les facettes de la case et dans celles de l'agent.

Détailler l'implémentation de la relation agent → case (lieu).

Au sein des facettes de l'agent, nous trouvons un pointeur sur la case.

```
struct Agent_ {
...
  struct Case_ * pCaseOuIAm;                    // ptr sur la case où est l'agent
...
};
```

Table 20 : Implémentation d'une case (partie décrivant le chaînage vers l'agent)

Détailler l'implémentation de la relation case → agent.

Au sein des facettes de la case, nous trouvons un pointeur sur l'agent.

```
struct Case_ {
...
 struct Agent_ * pAgent; // Si l'agent est chez lui ptr sur l'agent sinon ptr sur NULL
};
```

Table 21 : Implémentation d'une case (sans la partie décrivant le chaînage des cases entre elles).

3.D) Détailler l'implémentation des actions de l'agent sur le mobilier

Introduction.

De façon classique la représentation des actions est obtenue en faisant varier des relations. Dans *1DCorridor*, la seule relation qui lie l'agent à son support est sa position, faire varier cette position consiste à le déplacer.

Gestion des pointeurs côté agent.

Afin de simuler le déplacement de l'agent, il suffit de faire pointer son pointeur, vers la case suivante, i.e. celle qui est devant lui. Elle est obtenue en utilisant le chaînage des cases, i.e. les pointeurs sur les cases voisines.

En effet, au début du chapitre nous définissons une relation $\mathcal{R}$ entre deux lieux A et B, notée $\mathcal{R}$(A, B) ou encore A$\mathcal{R}$B. Après discussion, au moyen de *(Th1)* et de *(H_2)* nous concluons que dans ce domaine 1D : *chaque symbole est en relation $\mathcal{R}$ avec au moins un autre symbole, et au plus avec deux.*

Au niveau informatique, l'implémentation de la relation A$\mathcal{R}$B est réalisée au moyen de deux pointeurs : le lieu A reçoit un pointeur sur le lieu B, et réciproquement, alors il suffit de tirer sur ces pointeurs pour faire pointer le lieu de l'agent vers la case suivante.

Remarque : dans *1DCorridor* nous définissons un sens de parcours en utilisant celui de la lecture, et nous n'utilisons que lui. Ainsi l'agent ne se déplace que dans un sens, donc nous

avons seulement implémenté la moitié des pointeurs : ceux qui vont dans le sens A→B, et délaissé ceux qui vont dans le sens inverse (A←B).

Gestion des pointeurs côté case support.

Après le déplacement, la case où était l'agent, ne pointe plus sur lui mais, en général sur du vide. De même la case qui était devant l'agent, et qui vient de le recevoir, pointe sur lui.

```
void Avance_pAgent(Agent * pAgent) {    // pour avancer l'agent jusqu'à la case suivante
 Case * pCaseIci= pAgent->pCaseOuIAm;   // Utilisation d'un tampon pour plus de lisibilité
 Case * pNxtCase= pCase_QuelleEstLaCaseDevantToi_pAgent(pAgent);//récupère la case suivante:qui est devant

 fprintf(stderr,"Avance pCaseIci pNxtCase ");//Imprime sur la console pour tracer le déroulemt du traitemt
 PrintTesSlotsSurStderr_pCase(pCaseIci);
 PrintTesSlotsSurStderr_pCase(pNxtCase);
                                        // Actualisation des ptr dans les Cases
 pCaseIci->pAgent= NULL;                // L'ancienne Case ne pointe plus sur l'agent
 pNxtCase->pAgent= pAgent;              // La case suivante pointe sur l'agent
                                        // Actualisation du ptr dans l'agent
 pAgent->pCaseOuIAm= pNxtCase;          // L'agnt n'est plus ds l'ancienne Case mais dans la case suivante
}
```

Table 22 : Implémentation du déplacement de l'agent depuis la case où il est, jusqu'à la case suivante

4) Présenter les résultats (sans les interpréter)

4.A) Présenter le programme à l'utilisateur

195 - Écran au lancement du programme

Lancement du programme.
Il se lance au moyen de la commande *main*.

Présentation de l'écran au lancement du programme.
Au lancement, puisque initialement nous posons l'agent dans la ligne 1 et la colonne 1, à l'écran, nous obtenons la présentation ci-contre[69].

L'écran reprend les conventions d'affichage du graphisme (par opposition à celles des mathématiques) : l'origine de l'axe des y ou des J (ordonnées) est situé en haut, et son sens est dirigé vers le bas.

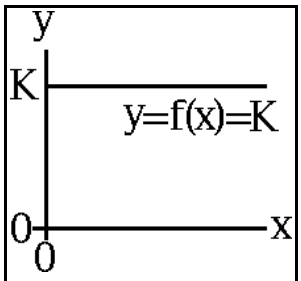
Utilisation du programme.
Une fois l'exécutable lancé, nous obtenons un écran graphique, sans invite de commande. Quand nous tapons la lettre *a* au clavier (a pour aide) nous obtenons, sur la console, cet affichage :

```
Usage : main
'a, A, h, H, ?' Pour afficher cette aide et sortir
'q' Pour quitter
'u' Pour avancer une fois
'd' Pour avancer dix fois
'c' Pour avancer 100 fois
Toute autre touche affiche cette aide et sort
```

Table 23 : Aide en ligne affichée à la console

La touche *u* fait avancer l'agent d'un pas ; la touche *d*, de 10. En fin de ligne l'agent retourne à la ligne suivante ; en fin de trame, il remonte en position 0, 0. Le cycle continue indéfiniment, car le but est de circuler selon un vecteur 1D infini.

4.B) Pas de résultat brut à présenter



196 - fonction constante

Au sein de notre progression de simulations, *1DCorridor* la modélisation informatique présente un être qui assimile la nourriture qui l'entoure, ce qui explique pourquoi il doit se déplacer. Mais nous imaginons qu'il le fait dans un milieu riche et homogène : le mouvement et l'assimilation se font sans à coup, et demeurent donc constants.

De plus, cette simulation dépend de plusieurs paramètres libres, i.e. la valeur calorique de la nourriture, la densité de sa diffusion ou le coût énergétique de l'action élémentaire. Pour le programmeur, ils sont tous aussi ajustables les uns que les autres, donc la quantité de points de fonctionnement de ce système est infini. Il serait donc sans intérêt d'en présenter un plutôt qu'un autre. Alors, à cette étape, nous disposons encore seulement de résultats constants et sans intérêt à présenter (figure de droite).

69 Ne pas oublier qu'en C la numérotation des tableaux commence à 0.

5) Conclusion

Afin de fournir à l'utilisateur une taxinomie de qualité, nous devons fournir une large gamme de simulations, allant de la cellule végétative, au traitement cognitif de l'information. À ce stade, avec *1DCorridor*, nous venons donc d'introduire le mouvement.

En soutenant que la conception de l'être *avanceur permanent*[70] était indirectement finaliste nous avons pu dire que le procès qu'il effectue constitue une action. Nous avons pu conclure que cet être était un agent.

La fonction de décision de notre agent se réduit à une règle *avance*, activée en permanence : nous disposons d'un être très simple, *l'agent situé minimal*.

6) Perspectives

Introduction de la reproduction et du parallélisme.

La cellule végétative, portée par sa structure se déplace. Elle trouve une nourriture abondante. Quand l'ancien cycle de reproduction de la cellule végétative $C \rightarrow c \rightarrow C \rightarrow c \rightarrow C$... est remplacé par un cycle comportant une règle expansive :

$C_i \rightarrow c_j\, c_k$	(i et j entiers positifs)	Une cellule peut produire deux graines
$c_i \rightarrow C_i$		Une graine produit une cellule

Alors les cellules trouvent aussi de l'espace pour se reproduire.

Introduction de la causalité au moyen de la bionique.

Le cycle de reproduction s'ouvre, alors une multitude d'êtres apparaissent et envahissent l'espace. Afin d'introduire la causalité dans notre modèle, inspirons-nous de l'évolution d'un groupe d'êtres vivants, lâché en pleine nature :

Ils se reproduisent exponentiellement, mais leur domaine physique est fini. Il se trouve donc sillonné d'êtres de la même espèce, qui avancent droit devant et creusent un sillon rectiligne, en assimilant la nourriture sur leur passage.

L'énergie qui était omniprésente et constante, se cantonne maintenant à certains lieux épargnés, et qui se font de plus en plus rares. Chaque agent ne peut assimiler en permanence une nourriture qui apparaît aléatoirement. Il s'adapte et s'ouvre au monde afin de percevoir son environnement : il fait le geste de manger seulement en présence de nourriture. C'est l'apparition de la causalité : l'être devient un agent réactif.

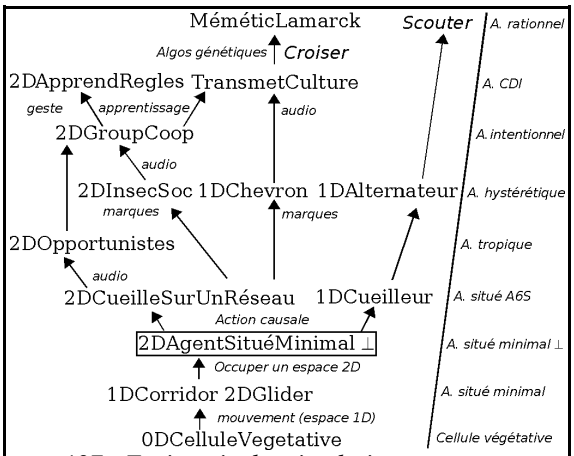
70 En effet, dans *1DCorridor*, l'agent réactif active en permanence son actionneur pour avancer : il va toujours de l'avant.

Ch. 32 2D ASM⊥ : 2DAgentSituéMinimalOrthogonal

1) Introduction

1.A) Distribuer aléatoirement l'énergie

Cette simulation est écrite pour modéliser une nouvelle répartition de l'énergie sur le support. Celle-ci n'est pas distribuée uniformément sur le support, mais elle est concentrée sur quelques groupes de cases, distribués aléatoirement.



197 - Taxinomie des simulations que nous présentons

1.B) Situer l'étape dans la taxinomie

Après *1D ASM*.

Jusqu'ici, l'agent situé minimal (*1D ASM*) se déplace en permanence sur le support 1D, i.e. dans un monde à une dimension. Sa cellule végétative, portée par sa structure, y trouve une énergie diffuse distribuée uniformément.

Nous pouvons aussi voir *1D ASM* comme un agent qui, dans un monde à 2D ou 3D, ne possède pas d'actionneur pour tourner : le résultat est équivalent, il ne peut pas changer de direction et se déplace en ligne droite.

Avant l'apparition de la causalité chez l'agent situé.

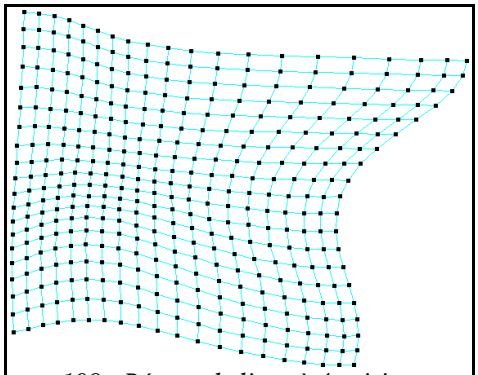
Dans cette simulation, nous modifions le support pour qu'y apparaissent des lots d'énergie distribués aléatoirement. Notre but est d'initier l'étape suivante de la progression, où les agents situés sur le support, percevront leur environnement et agiront causalement.

2) Quels outils informatiques pour modéliser cette étape ?

2.A) Modélisation du monde (support + mobilier)

a) Représentation du support

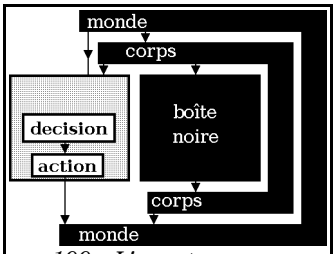
Pour modéliser le monde de cette simulation, nous réutilisons les mêmes outils que pour *1DCorridor*, où nous disposions déjà d'un réseau 2D de lieux potentiellement munis d'un pointeur sur chacun de leurs quatre voisins. Nous continuons de l'utiliser, mais cette fois, au lieu de seulement utiliser le pointeur *est*, nous renseignons aussi les pointeurs *nord*, *sud* et *ouest*, que nous avions délaissés quand nous travaillions en 1D.


198 - Réseau de lieux à 4 voisins

Notez que pour cette simulation nous avons besoin d'un monde varié, C'est pourquoi, afin d'ajouter

de la diversité, les lieux situés sur le bord ne sont pas rebouclés, ce qui aurait donné au support une forme de boule ou de tore. Nous choisissons plutôt une organisation déstructurée, i.e. nous les rebouclons un peu n'importe où, afin de rendre relativement erratiques les trajets de l'agent.

Notez aussi, que pour des raisons de clarté, les rebouclages du réseau aux extrémités ne sont pas représentés.



199 - L'agent avance toujours droit devant lui

b) Représenter le mobilier du domaine (agent et énergie)

Représenter l'agent : le *2D ASM⊥*.

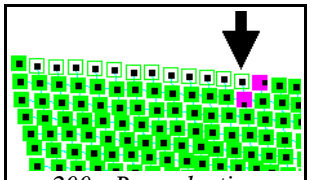
Pour représenter l'agent qui se déplace sur notre support, nous reprenons l'agent situé minimal, et le plaçons sur le support 2D où il continue à se mouvoir de la même manière, i.e. droit devant lui.

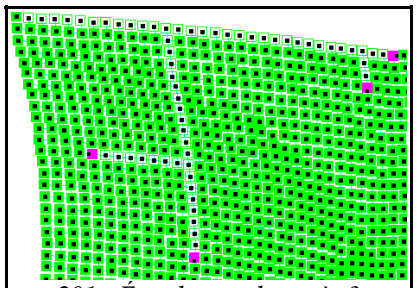
Représenter l'énergie sur les cases du support.

Au départ de l'action, l'énergie est distribuée régulièrement dans le monde : chaque lieu du support reçoit une pièce de nourriture (voir les illustrations ci-dessous).

La reproduction orthogonale de *2D ASM⊥*.

L'ASM⊥ se reproduit dans un monde 2D en posant son enfant orthogonalement à son axe de déplacement.


200 - Reproduction orthogonale de 2D ASM⊥


201 - État du monde après 3 reproductions de 2D ASM⊥

Dans l'illustration ci-contre à droite, d'abord l'ASM part en haut à gauche du point (0, 0) et se dirige vers l'est.

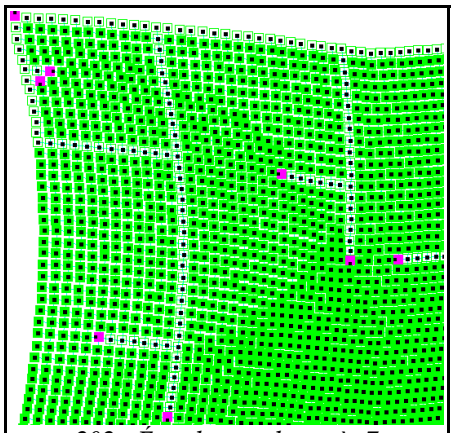
Arrivé au point (0, 12) marqué par la flèche noire, il se reproduit en posant son enfant tourné vers le sud.

Finalement, un cycle plus tard, nous obtenons l'état du monde tel qu'il est affiché.

Introduction du parallélisme.

Si nous continuons le processus, après trois reproductions de nos agents nous obtenons l'état du monde ci-dessus à gauche. Et, après sept reproductions nous obtenons l'état du monde ci-contre à droite. Finalement sa descendance se répartit rapidement sur tout le support.

Selon un point de vue informatique, nous devons donc implémenter un système parallèle pour gérer simultanément le groupe d'agents répartis sur le support.


202 - État du monde après 7 reproductions de 2D ASM⊥

2.B) Représenter le domaine (le support)

La simulation *2D ASM⊥* reprend la représentation des lieux décrite dans les simulations précédentes. Cependant, la représentation du domaine sous forme d'un graphe de lieux est privilégiée. Dans ce mode, il faut juste signaler une petite différence de visualisation :

- Les cases pleines sont représentées en vert (couleur de la chlorophylle).
- Les cases qui viennent d'être mangées ne sont représentées que par un petit carré noir.

2.C) Présentation de la mise en oeuvre du programme par l'utilisateur

Lancement du programme.
Il se lance au moyen de la commande *main*.

Utilisation du programme.
Une fois l'exécutable lancé, nous obtenons un écran graphique, sans invite de commande. Quand nous tapons la lettre *a* au clavier (a pour aide) nous obtenons, sur la console, cet affichage :

```
Usage : main
'a, A, h, H, ?' Pour afficher cette aide et sortir
'q' Pour quitter
'u' Pour avancer une fois
'd' Pour avancer dix fois
'c' Pour avancer 100 fois
Toute autre touche affiche cette aide et sort
```

Table 24.: Aide en ligne affichée à la console

La touche *u* fait avancer l'agent d'un pas ; la touche *d*, de 10 ; la touche *c*, de 100.

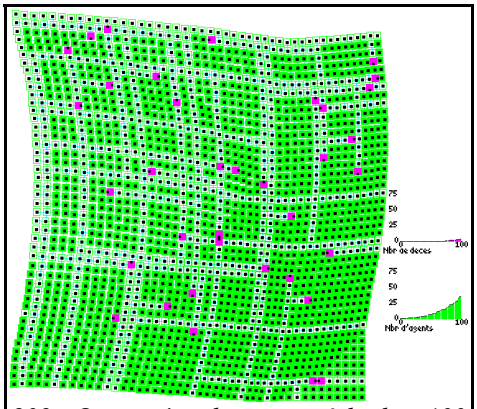
3) Résultats

3.A) Présenter les résultats

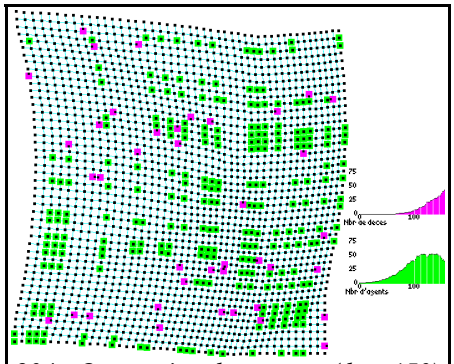
a) Introduction

Sur un réseau de 46x46 lieux nous plaçons en (0, 0)[71] un *ASM*⊥ initial orienté à l'est.

Au fil de la simulation, nous traçons deux histogrammes : En haut, le nombre des décès d'agents (couleur magenta). En bas, le nombre des agents vivants sur le support (couleur vert clair).



203 - Occupation du support à la date 100



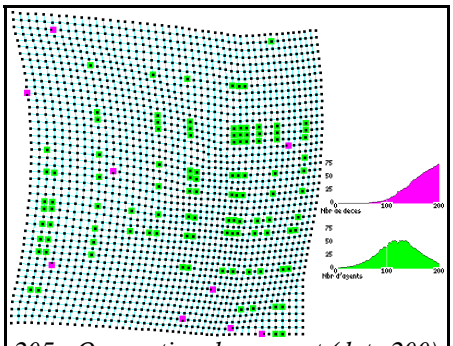
204 - Occupation du support (date 150)

b) Date 100

À la date 100 (ci-contre à droite) l'agent initial s'est rapidement reproduit selon une courbe à l'allure exponentielle. Nous ne déplorons encore que quelques morts (4). La première survient à la date 67.

c) Date 150

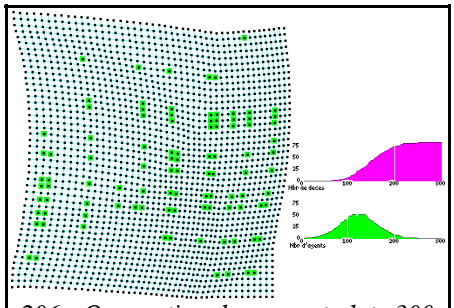
À la date 150 (ci-contre, à gauche), déjà les agents saturent le support. À la date 110, nous avons déjà 52 agents. La courbe sature à 53 aux dates 129 et 137 puis s'infléchit vers le bas.



205 - Occupation du support (date 200)

71 Nous utilisons les conventions de repérage en (x, y) de l'informatique, où l'axe des y se trouve tout en haut de l'image et descend en s'accroissant.

À cause de cette saturation du support nous déplorons un nombre de morts grandissant. À la date 150, il est de 33.



206 - Occupation du support, date 300

d) Date 200

À la date 200 (ci-contre, à droite), les agents ont presque épuisé la ressource. Seuls 8 d'entre eux survivent et leur nombre décroît régulièrement.

À cause de cette saturation du support nous déplorons un nombre de morts grandissant. À la date 200, il est de 75.

e) Date 300

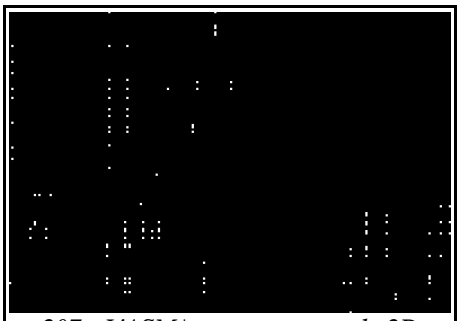
À la date 300 (ci-contre, à gauche), les agents ont presque épuisé la ressource, mais passé la date 250, aucun n'a survécu : les 84 sont morts.

3.B) Interpréter les résultats

La descendance de l'ASM⊥ sature le support.

L'agent situé minimal orthogonal se reproduit dans un monde 2D en posant son enfant orthogonalement à son axe de déplacement. Ainsi, rapidement sa descendance se répartit sur le support. Les agents sillonnent le support en mangeant.

Les traces de leur passage s'entrecroisent au hasard et la nourriture disparaît aléatoirement des cases. La surpopulation induit une crise des ressources, il ne reste plus que quelques rares îlots de nourriture : l'énergie qui était omniprésente et distribuée régulièrement sur le support, se retrouve cantonnée à quelques rares lieux disséminés aléatoirement (figures ci-dessus à gauche et ci-contre à droite).



207 - L'ASM⊥ sature un monde 2D

Analyse des résultats : la répartition de l'énergie.

À cause de cette nouvelle distribution de l'énergie, l'agent ne peut plus chercher à assimiler en permanence une nourriture qui maintenant apparaît aléatoirement.

L'agent va devoir devenir réactif.

Il doit s'adapter. Cet être devra s'ouvrir au monde afin de percevoir son environnement, il se dotera d'un actionneur afin de faire le geste de manger seulement en présence de nourriture. Ceci induira l'apparition de la causalité. L'être deviendra un agent réactif.

4) Conclusion

Nous fournissons à l'utilisateur une taxinomie composée d'une large gamme de simulations, allant de la cellule végétative, au traitement cognitif de l'information. À ce stade, avec *2D ASM⊥*, nous disposons d'un modèle où la nourriture est distribuée par lots répartis aléatoirement sur le support.

5) Perspectives

Vers l'agent situé.

Dans les étapes suivante, cette situation va induire des changements chez l'agent, apparition de capteur pour percevoir le monde et d'actionneur pour manger. Ainsi dans notre progression, nous débouchons sur les agents cueilleurs, en situation s dans un monde 1D ou 2D.

Dans une simulation de l'agent réactif le programmeur doit successivement :

- Modéliser un monde où la nourriture est distribuée aléatoirement.
- Ouvrir la perception de l'agent au monde, au moyen de capteurs.
- Rajouter un actionneur pour manger.
- Utiliser une table ou une fonction afin de programmer le module *décision* de l'agent, et commander ce nouvel actionneur.

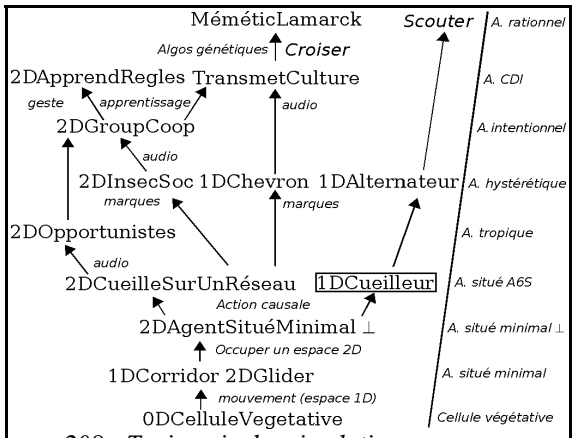
Ch. 33 *1DCueilleur*, l'agent situé fait le geste de cueillir sa nourriture

1) Situer l'étape dans la taxinomie

À cette étape, l'énergie qui était omniprésente et constante sur le support, se cantonne maintenant, à certains lieux.

L'agent ne peut en permanence essayer d'assimiler une nourriture, qui apparaît aléatoirement. Il doit s'adapter, alors il s'ouvre au monde afin de percevoir son environnement pour faire le geste de manger seulement en présence de nourriture.

C'est l'apparition de la causalité : l'être devient un agent réactif, i.e. un vrai agent situé.



208 - Taxinomie des simulations que nous proposons

2) Introduire une représentation simple dans la simulation

2.A) Dans une simulation de l'agent réactif, le programmeur doit :

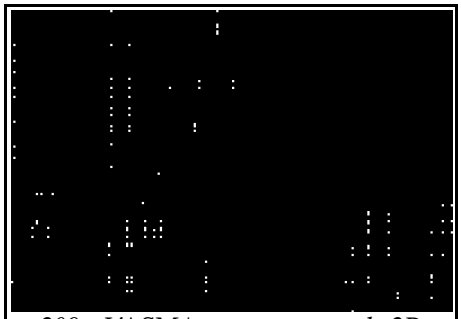
- Modéliser un monde où la nourriture est distribuée aléatoirement.
- Rajouter un actionneur pour manger.
- Ouvrir la perception de l'agent au monde, au moyen de capteurs.
- Utiliser une table ou une fonction afin de programmer le module *décision* de l'agent, qui commande ses actions.

2.B) *1DCueilleur* : un être causal qui constitue un vrai agent situé

L'agent situé minimal orthogonal (*ASM*⊥) donne naissance à l'agent situé.

Dans le chapitre précédent nous avons montré comment l'*ASM*⊥ se reproduit dans un monde 2D en posant son enfant orthogonalement à son axe de déplacement.

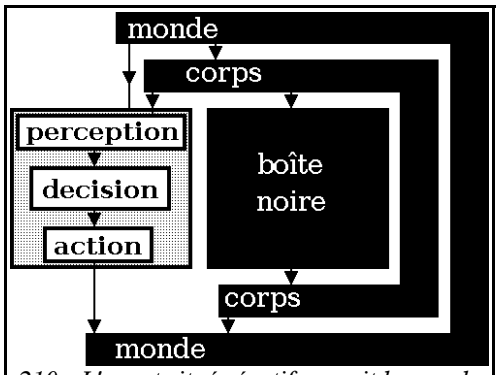
Ainsi, rapidement sa descendance se répartit sur le support, envahit et sature le monde. Cette surpopulation induit une crise des ressources, il n'y reste plus que quelques rares îlots de nourriture.


*209 - L'*ASM⊥ *sature un monde 2D*

Alors apparaissent les agents cueilleurs : ces êtres débrouillards récoltent la nourriture et l'emportent dans une poche où il la maintiennent au contact de leur cellule végétative, ce qui facilite son assimilation. Ainsi ils y trouvent un grand avantage sélectif, qui est rapidement retenu par le mécanisme d'évolution des espèces.

Ajouter un actionneur pour manger.

Dans cette simulation *1DCueilleur*, la nourriture n'est plus distribuée universellement et de façon constante. Donc l'agent ne peut plus l'assimiler régulièrement au cours de son déplacement. Quand il en trouve, il doit mettre les bouchées doubles, i.e. disposer d'un actionneur qui facilite et accélère l'assimilation cette énergie, par la cellule végétative. Une nouvelle action[72] apparaît : manger la nourriture qui se tient sur la case juste devant l'agent.



210 - L'agent situé réactif perçoit le monde

De façon générale cette simulation modélise un agent qui fait le geste de cueillir sa nourriture, tout en se déplaçant sur son support. Étudions la représentation de la connaissance qui permet d'implémenter très simplement ce programme.

Modéliser un monde où la nourriture est distribuée aléatoirement.

Les autres agents sillonnent le monde en mangeant. Les traces de leur passage s'entrecroisent au hasard et la nourriture apparaît aléatoirement dans les cases.

S'ouvrir au monde au moyen de capteurs.

L'agent quitte son solipsisme, il s'ouvre au monde et commence à percevoir son environnement, pour cela il est doté d'un capteur qui détecte si de la nourriture est présente sur la case devant lui.

2.C) Représenter le domaine (le support)

La simulation *1DCueilleur* reprend la représentation des lieux décrite dans les simulations précédentes. Cependant, la représentation du domaine sous forme d'un graphe de lieux est privilégiée. Dans ce mode, il faut juste signaler une petite différence de visualisation :

- Les cases vides ne sont représentées que par un petit carré noir.
- Les cases pleines sont représentées en vert (couleur de la chlorophylle).
- Les cases qui viennent d'être mangées et n'ont pas repoussé sont en blanc, entourées d'un carré vert.

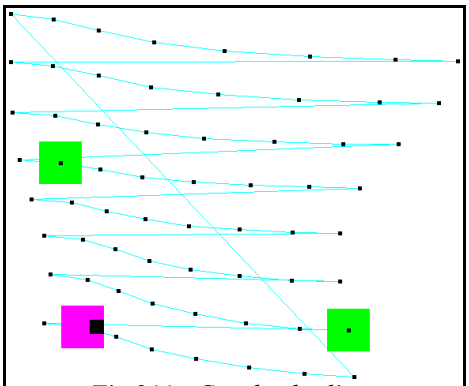


Fig 211 : Graphe des lieux

2.D) Représenter le mobilier du domaine

a) Modéliser un agent réactif

Le programme *1DCueilleur*, modélise un agent situé réactif (perception décision action) capable d'effectuer un traitement combinatoire de l' information, qu'il perçoit au moyen de son capteur de nourriture.

L'être conserve sa structure qui porte sa cellule végétative.

En passant à l'étape *1DCueilleur*, l'être conserve son actionneur de mouvement qui déplace sa structure porteuse. Cette dernière porte sa cellule végétative qui assimile de la nourriture.

72 Au travers de ce mémoire, nous distinguons rarement l'action et son procès, ce choix est déjà expliqué dans la partie théorique, et dans le chapitre sur *1DCorridor*, puis à la fin du chapitre sur 2DCueilleur.

Maintenant l'être se dote d'un capteur pour percevoir la nourriture.

L'agent s'ouvre au monde et commence à percevoir son environnement, pour cela il est doté d'un capteur d'énergie. Quand la nourriture est présente sur la case devant lui, la sortie de ce capteur fait monter un signal électrique dédié qui exprime que, *oui*, il perçoit quelque chose.

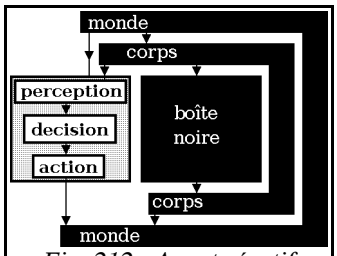


Fig. 212 - Agent réactif

Maintenant l'être se dote d'un actionneur pour manger.

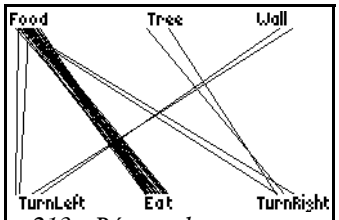
En passant de *1DCorridor* à cette étape, l'être se dote d'un actionneur pour manger. Nous pouvons le voir comme un organe de préhension. Il saisit la nourriture et l'enfourne rapidement dans une poche qui la transportera. Ici, elle se retrouve au contact de la cellule végétative qui l'assimile lentement.

Description formelle de l'action *manger.*

L'action est décrite comme une variation de la relation entre le mobilier et le support, à savoir comme le changement du lien entre la pièce de nourriture et son support. Suivant la sophistication de la simulation, nous trouvons différentes modélisations. Dans les cas les plus simples, le fait de manger met pas fin à la relation de contenant entre la case et la nourriture, et elle augmente le niveau d'énergie de l'agent. Dans cette simulation, la cueillette de la nourriture remet à zéro le niveau d'énergie de la case cueillie, ainsi, au fil du temps, elle peut se reconstituer : l'herbe se remet à pousser.

Implémenter la prise de décision de commander l'action au moyen d'une fonction câblée.

Nous devons implémenter le fonctionnement suivant : quand l'agent perçoit de la nourriture sur la case devant lui, il doit faire le geste de la cueillir et de la mettre dans son estomac.



213 - Réseau de neurones avec liaison food→eat *renforcée*

Plus concrètement, précisons le synoptique à implémenter. Il faut que quand la nourriture apparaît sur la case devant l'agent, son capteur d'énergie soit activé. D'abord, pour implémenter cette fonction, nous pouvons utiliser la table d'un automate fini. Sinon, dans un fonctionnement sub-symbolique que nous détaillerons plus tard, le codage de la décision est simple, il se réduit à un lien connectant le capteur *nourriture(food)* à l'actionneur *mange(eat)*.

Illustration ci-contre à droite : par exemple, dans le réseau de neurones ci-contre, pour déclencher l'action de manger en présence de nourriture, la liaison food→eat (i.e. nourriture→mange) est renforcée[73].

b) Représenter la nourriture, qui est distribuée aléatoirement sur le support

Auparavant, dans *1DCorridor* la nourriture était diffusée uniformément sur tout le domaine, au point que nous négligions de la représenter. Par contre, quand nous passons à *1DCueilleur*, la nourriture est plus dense et elle apparaît aléatoirement dans les cases, et donc, pour la première fois, nous devons la modéliser.

2.E) Représenter les relations du mobilier avec son support

Implémentation d'un agent sur son support.

Dans *1DCueilleur*, cette implémentation demeure inchangée.

Expliquer comment une case peut porter de la nourriture.

Dans *1DCueilleur*, la nourriture apparaît aléatoirement dans les cases. Dans les paragraphes suivants, nous détaillons comment une case qui porte de l'énergie, est implémentée au niveau informatique.

73 Pour plus de précisions, le lecteur peut se reporter au chapitre sur la simulation *2DApprendRègles*.

2.F) Représenter les actions de l'agent sur le mobilier

Modélisation informatique du déplacement de l'agent d'un lieu à un autre.
Elle est implémentée comme précédemment.

Modélisation informatique des repas de l'agent.
Pour la première fois nous représentons la nourriture et sa consommation. Ainsi, quand l'agent entre dans un lieu qui en contient, il la mange, elle disparaît. Elle ne repousse qu'au bout d'un délai. Ce comportement doit être modélisé dans le programme.

3) Détailler l'implémentation informatique de la simulation

3.A) Détailler l'implémentation informatique du domaine (le support)

a) Implémentation d'une case dans *1DCueilleur*

Les adjonctions à la déclarations de base de la case.
La simulation *1DCueilleur* modélise le fait que maintenant l'agent mange la nourriture. Ensuite, il faut juste signaler une petite différence de visualisation : les cases vides sont représentées par un petit carré noir. Les pleines sont représentées en vert et celles qui viennent d'être mangées sont en blanc, entourées d'un carré vert.

Voyons maintenant, selon un point de vue informatique, comment l'ajout de deux champs à sa structure initiale permet son implémentation.

```
struct Case_ {
...
  int ReBirthDate;      // Date à laquelle la case mangée retrouve son contenu initial.
...                     // Si -1, elle n'a pas été consommée.
  int Nature;           // Nature, contenu de la case
};
```

Table 25 : Ajout d'un champ à l'implémentation de la case initiale.

Modification de l'affichage initial de la case.
Il nous faut modifier l'affichage initial de la case pour prendre en compte un nouveau champ introduit.

```
void DrawTonContenuSiDifferentDeSolEtPasMange_A7DateYDansCeScreen(Case * pMyCase, int date, SDL_Surface * screen)
{
 if
  (pMyCase->Nature != sol) {      // Ce n'est pas une couche d'apprêt, ne dessine que si nécessaire
  if(date >= pMyCase->ReBirthDate)// Si la date actuelle dépasse la date de renaissance
   Draw_FillRect(screen,          //  alors on affiche son contenu
    pMyCase->CoinHauGauX,
    pMyCase->CoinHauGauY,
    CaseSideSize,
    CaseSideSize,
    pMyCase->Couleur );
   else                           // sinon on l'entoure seulement d'un cadre de sa couleur
   Draw_Rect(screen,
    pMyCase->CoinHauGauX,
    pMyCase->CoinHauGauY,
    CaseSideSize,
    CaseSideSize,
    pMyCase->Couleur );
  }
}
```

Tab 26.: Affichage à l'écran, d'une case contenant de la nourriture, en fonction de sa date de repousse

3.B) Détailler l'implémentation informatique du mobilier du domaine

a) Modéliser un agent réactif (perception décision action)

Le programme *1DCueilleur* modélise un agent réactif capable d'effectuer un traitement combinatoire de l'information qui perçoit au moyen de son capteur de nourriture.

Maintenant l'être se dote d'un capteur pour percevoir la nourriture.

L'agent s'ouvre au monde et commence à percevoir son environnement, pour cela il est doté d'un capteur de nourriture. Quand elle est présente sur la case devant lui, la sortie de ce capteur fait monter un signal électrique qui exprime que, oui, il voit quelque chose.

Maintenant regardons comment ce fonctionnement est simulé par informatique.

```
Case * pCase_QuelleEstLaCaseDevantToi_pAgent(Agent * pAgent) { // retourne un ptr sur la case devant l'agent

 Case * pCaseIci= pAgent->pCaseOuIAm;                    // Avec le ptr sur l'agent, retrouve la case où il est
 Case * pNxtCase;                                        // Tampon : Un ptr sur la case qui est devant

 switch (pAgent->Direction) {                            // En fonction de la direction de l'agent
  case nor : pNxtCase= pCaseIci->pNxtCaseNor; break;     // met le ptr sur la case qui est devant ds le tampon
  case sud : pNxtCase= pCaseIci->pNxtCaseSud; break;
  case est : pNxtCase= pCaseIci->pNxtCaseEst; break;
  case wst : pNxtCase= pCaseIci->pNxtCaseWst; break;
  default  : fprintf(stderr, "Erreur dans Avance_pAgent\n");
 }

 return pNxtCase;                                        // retourne le tampon
}
```

Table 27 : Comment l'agent peut savoir la nature de la case qui est devant lui

Implémenter la prise de décision de commander l'action.

La relation de causalité simulée s'énonce ainsi : si la case devant contient de la nourriture, alors l'agent avance et la mange.

```
void Agent_Agis_A7Date(Agent * pAgent, int date) {

 Case * pNxtCase= pCase_QuelleEstLaCaseDevantToi_pAgent(pAgent);
 int ChoseDevantNature = pNxtCase->Nature;

 if     (ChoseDevantNature == food)     Mange_pAgent_A7Date     (pAgent, date);
 else if(ChoseDevantNature == sol)      Avance_pAgent           (pAgent);
 else fprintf(stderr, "Erreur dans Agent_Agis\n");
}
```

Table 28 : Comment l'agent prend une décision en fonction de la nature de la case qui est devant lui.

Maintenant l'être se dote d'un actionneur pour manger.

L'action est décrite comme une variation de la relation entre le mobilier et le support, à savoir comme le changement du lien entre l'agent et deux lieux : le fait de manger ne met pas fin à la relation de contenant entre la case et la nourriture ; elle augmente seulement le niveau d'énergie de l'agent.

```
void Mange_pAgent_A7Date(Agent * pAgent, int date) {

 Case * pCaseIci= pAgent->pCaseOuIAm;     // Un tampon pour la case où se trouve l'agent
 Case * pNxtCase= pCase_QuelleEstLaCaseDevantToi_pAgent(pAgent);// Un tampon pour la case devant l'agent
                                          // Actualisation des ptr dans les Cases
 pCaseIci->pAgent= NULL;                  // L'ancienne Case ne pointe plus sur l'agent
 pNxtCase->pAgent= pAgent;                // La case suivante pointe sur l'agent
 pNxtCase->ReBirthDate= date+ReGrowingDelay;// Puisque l'agent mange, la nourriture, elle disparaît un moment

                                          // Actualisation du ptr dans l'agent
 pAgent->pCaseOuIAm= pNxtCase;            // L'agent n'est plus dans l'ancienne Case mais dans la case suivante


 pAgent->EnergyLevel+= FoodEnergyVal;     // L'ingestion de la nourriture augmente le niveau d'énergie de l'agent
}
```

Table 29 : Implémentation de l'action de manger effectuée par l'actionneur de l'agent

b) Modéliser une case du graphe de lieux (ou du couloir) contenant de la nourriture

Dans *1DCueilleur*, une case est modélisée comme à l'étape précédente (*1DCorridor*), mais il faut implémenter en plus le fait que l'herbe qui vient d'être mangée, repousse et est ensuite comestible au bout d'un certain laps de temps. Toute cette implémentation est décrite dans le paragraphe précédent *Implémentation d'une case dans 1DCueilleur*.

3.C) Détailler l'implémentation des relations du mobilier ↔ support

Implémentation d'un agent sur son support.

Dans *1DCueilleur*, elle demeure inchangée.

Modéliser un monde où la nourriture est distribuée aléatoirement.

Initialement dans *1DCorridor*, la nourriture est diffusée uniformément dans tout le domaine ; mais quand nous passons à *1DCueilleur*, la nourriture apparaît aléatoirement dans les cases. Au niveau informatique, ceci est implémenté au moyen d'un générateur aléatoire programmé autour de l'instruction *rand*.

```
void InitGridTabCase () {
...
 {                                          // Deux tampons destinés à être remplis
  int RandI, RandJ;                         //  par des valeurs aléatoires
  for (j=0; j<CaseNbr/20; j++) {            // La quantité de nourriture à distribuer est
   RandI= random() % CaseHoriNbr;           // paramétrée en fonction de la taille de la grille
   RandJ= random() % CaseVertNbr;

   Case * pCase = GridTabCase[RandJ][RandI];  // Tampon pr aller chercher la case correspondante
   InitLanatureDe7CaseA7Nature(pCase, food);  // dépôt de la nourriture
  }
 }
}
```

Table 30 : Implémentation de la distribution aléatoire de nourriture sur la grille.

3.D) Représenter les actions de l'agent sur le mobilier

Modélisation informatique du déplacement de l'agent d'un lieu à un autre.

Elle est implémentée comme précédemment.

Modélisation informatique des repas de l'agent.

Celle-ci est expliquée dans le paragraphe traitant de l'actionneur pour manger.

3.E) Représenter la prise de décision des actions de l'agent sur le mobilier

Nous sommes dans un monde 1D la prise de décision de l'action de l'agent est très simple, presque simpliste :

```
Si dans la cases devant l'agent se trouve de la nourriture
  alors il mange la nourriture dans la case devant
  sinon il avance
```

Table 31 : Implémentation de la prise de décision des actions de l'agent sur le mobilier

4) Présenter les résultats (sans les interpréter)

4.A) Présentation du programme à l'utilisateur

Lancement de l'exécutable
Il se lance au moyen de la commande *main*.

Présentation de l'écran au lancement du programme
Voir la figure 1 de ce chapitre.

Utilisation du programme.
Une fois l'exécutable lancé, nous obtenons un écran graphique, sans invite de commande. Quand nous tapons la lettre *a* au clavier (a pour aide) nous obtenons, sur la console, cet affichage :

```
Usage : main
'a, A, h, H, ?' Pour afficher cette aide et sortir
'q' Pour quitter
'u' Pour avancer une fois
'd' Pour avancer dix fois
'c' Pour avancer 100 fois
Toute autre touche affiche cette aide et sort
```

Table 32 : Aide en ligne affichée à la console

La touche *u* fait avancer l'agent d'un pas ; la touche *d*, de 10. En fin de ligne l'agent retourne à la ligne suivante ; en fin de trame, il remonte en position 0, 0. Le cycle continue sans fin, car le but est de circuler selon un vecteur 1D infini.

4.B) Résultats

a) L'implémentation de l'agent

En présence de nourriture, l'agent fait le geste de manger. Cette décision à prendre demeure encore simple. Elle peut encore être effectuée au moyen d'un automate fini car sa table demeure encore petite. Ce ne sera plus le cas quand le nombre des perceptions en entrée va augmenter.

b) La gestion de la nourriture

Affichage d'un état interne à l'agent : le niveau de sa réserve de nourriture

Au sein de notre progression de simulations, *1DCueilleur,* la modélisation informatique de cette étape présente un agent situé[74] qui cueille et mange la nourriture trouvée sur son chemin.

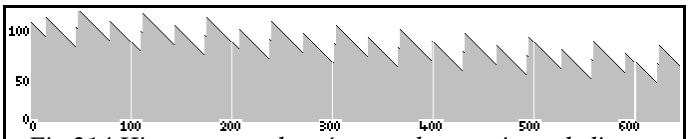


Fig.214 Histogramme des réserves de nourriture de l'agent

Puisqu'elle est distribuée aléatoirement sur le domaine, l'agent mange irrégulièrement. Ainsi apparaît la question de la pérennité de cet être. Le programme permet de se faire une bonne idée sur la situation en affichant un histogramme de la réserve d'énergie au fil du temps. Le résultat, sous sa forme brut apparaît ainsi :

Analyse de ce résultat.
Sur la figure, le temps est porté directement en abscisse : chaque ligne verticale correspond à un cycle, i.e. à un élément de temps unité. La quantité de nourriture en réserve est portée en ordonnée.

74 *Agent situé* est un terme dédié. En anglais, il se traduit par *situated agent*.

Nous obtenons une courbe en dents de scie. Quand l'agent trouve de la nourriture, il la mange et son potentiel vital augmente instantanément. Quand il parcourt vainement le monde à la recherche d'énergie, à chaque instant ses réserves décroissent.

5) Conclusion

a) Introduction de la causalité

Nous fournissons à l'utilisateur une taxinomie composée d'une large gamme de simulations, allant de la cellule végétative, au traitement cognitif de l'information. *1DCueilleur*, est une étape de notre taxinomie. À ce niveau de complexité, le but de l'agent est de manger pour se déplacer, et se déplacer pour manger. Pour cela il doit agir causalement : un module de décision relie les capteurs aux actionneurs. Ainsi nous venons d'introduire la causalité.

b) Introduction de l'A6S (Agent Somatique Situé en Survie Sur un Support Simple)

Notre premier agent situé, l'ASM était vraiment limité, mais ici, avec l'agent *1Dcueilleur*, nous sommes déjà en présence de ce que nous définissons comme un vrai agent situé : l'A6S, alors voici le moment de détailler cette notion centrale de notre travail.

Agent en Survie.

Parce qu'il effectue des procès indirectement sous-tendus par le but de survivre, alors nous pouvons les appeler *actions*.

Somatique.

Car il porte en lui une partie somatique qui possède sa vie propre et sert en particulier à la reproduction de l'être entier.

Situé Sur un Support.

L'agent est en relation spatiale avec le lieu qui lui sert de support.

Simple.

Le support n'est pas outrageusement simplifié comme celui des jeux digitaux, mais nous le choisissons très simple, afin de pouvoir le traiter par les mathématiques.

Dans ce cadre, notre but est de formaliser l'agent situé, de le faire apparaître comme le prolongement de l'automate à états finis, et de fournir aux chercheurs en sciences sociales un outil mathématique formalisé.

6) Perspectives

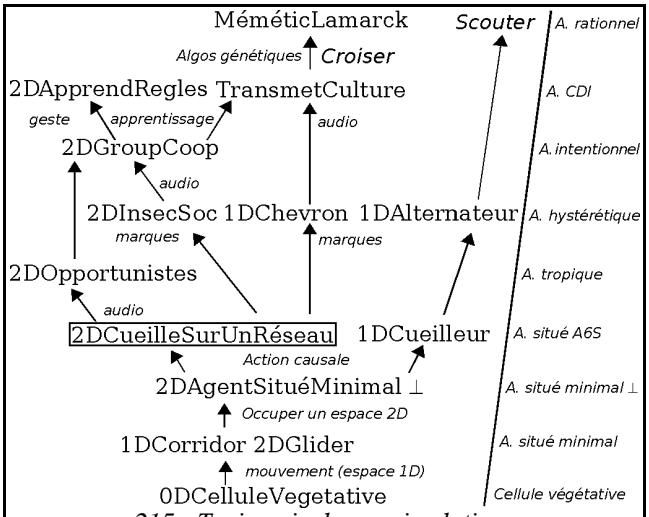
Au moyen des simulations à venir, nous allons renforcer la complexité du traitement causal de l'agent en introduisant un monde à 2 dimensions, et/ou des objets plus variés sur le support.

Ch. 34 2DCueilleSurUnRéseau, chercher sa nourriture parmi les obstacles

1) Introduction

1.A) Passer à un monde 2D

À cette étape, nous quittons le support simpliste en 1D, pour simuler un monde en deux dimensions. Cette simulation modélise un agent situé plus élaboré : un cueilleur qui cherche sa nourriture au milieu des obstacles, i.e. sur un réseau de lieux possédant chacun quatre voisins. Pour ce faire, il reçoit de nouveaux actionneurs afin de pouvoir tourner.

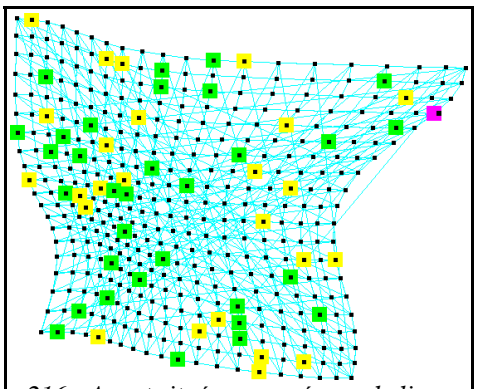


215 - Taxinomie de nos simulations

1.B) Situer cette étape dans notre taxinomie

L'introduction d'un support à deux dimensions, parsemé d'obstacles (cases jaunes) induit plusieurs effets : d'abord, l'être perçoit un environnement plus varié : son nombre de capteurs/senseurs augmente. Ensuite, dans ce monde plus complexe, il effectue une gamme d'actions plus large : il tourne à gauche ou à droite : son nombre d'actionneurs augmente.

En conséquence, chez cet agent, qui perçoit mieux et agit plus, le traitement causal de l'information se complexifie. Ainsi, dans cette simulation, apparaît l' agent situé typique : l'agent situé réactif *perception décision action*.


216 - Agent situé sur un réseau de lieux

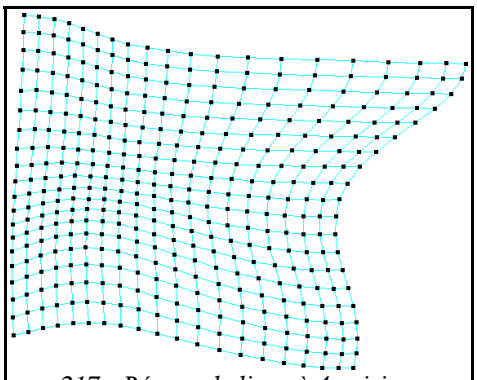
2) Matériels, méthodes et implémentation

2.A) Comment pour modéliser cette étape ?

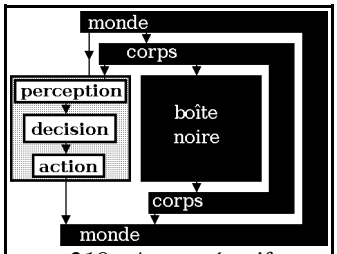
a) Modéliser le monde (support + mobilier)

Pour modéliser le monde de cette simulation, nous réutilisons les mêmes outils que dans le chapitre précédent. En effet, pour *1DCorridor* nous disposions déjà d'un réseau 2D de lieux, où chacun d'eux est muni de quatre pointeurs (un par voisin).

Alors nous continuons d'utiliser cette représentation, mais cette fois, au lieu de seulement utiliser le pointeur *est*, nous renseignons aussi les pointeurs *nord*, *sud* et *ouest* que nous avions délaissés, quand nous travaillions en 1D.


217 - Réseau de lieux à 4 voisins

Notez que pour cette simulation nous souhaitons disposer d'un monde varié, c'est pourquoi, d'abord la nourriture est distribuée aléatoirement sur les cases. De plus, pour ajouter de la diversité, les lieux situés sur le bord ne sont pas rebouclés au même niveau, ce qui donnerait au support une forme de boule : nous utilisons plutôt un rebouclage très déstructurée, afin de rendre relativement erratiques les trajets de l'agent qui parcourrait le support en avançant en ligne droite.



218 - Agent réactif

b) Modélisation de l'agent

Le traitement de l'information effectué par l'agent demeure simple. Il possède seulement quelques capteurs et quelques actionneurs. Ce processus peut encore être effectué au moyen d'une table ; pourtant, dans l'implémentation informatique de *2DCueilleur*, il est déjà effectué au moyen de règles :

```
if     (ChoseDevantNature == food)  Mange_pAgent_A7Date(pAgent, date); // Si devant ya nourriture alors mange
else if(ChoseDevantNature == sol)   Avance_pAgent           (pAgent);  // sinon si devant ya rien alors avance
else if(ChoseDevantNature == arbre) TourneDroite_pAgent     (pAgent);  // sinon si devant ya un arbre alors t droite
else if(ChoseDevantNature == mur)   TourneGauche_pAgent     (pAgent);  // sinon si devant ya un mur alors t gauche
else fprintf(stderr, "Erreur dans Agent_Agis\n");                      // sinon : erreur
```

Table 33 : La simulation informatique utilise des règles très simples

c) Justifier la modélisation de l'agent

Dans la modélisation de l'agent, justifier l'apparition d'un autre paramètre : la direction.

1DCueilleur est un espace à une dimension, où chaque lieu ne connaît que son successeur. Dans ce support, l'agent, se déplace selon un unique sens que, conventionnellement nous avons copié sur celui de l'écriture. Ainsi, cet être ne se déplace que vers le lieu suivant, et sa relation au support se limite à être *posé sur*. Finalement, afin de simplifier le traitement, nous ne représentons pas son sens, car il est implicite.

Par contre, *2DCueilleur* est un espace à deux dimensions, où chaque lieu connaît ses quatre voisins. Il en découle que, sur ce support, l'agent est muni d'un sens complexe, i.e. d'une direction qui peut être instanciée par une de ces quatre valeurs : *nord, sud, est, ouest*. En conséquence, la relation de cet agent au support comporte deux traits : *posé sur* et *sens/direction*. Ainsi il se déplace dans quatre directions et la modélisation informatique de ce procès devient plus complexe. Certes, quand il avance, l'agent passe dans la case qui est devant lui, mais la recherche de cette case devient plus complexe, car maintenant, elle se fait en fonction de sa direction.

Dans la modélisation de l'agent, justifier l'apparition du changement de direction.

Dans la modélisation de l'agent, l'apparition des capteurs et actionneurs n'est pas téléguidée par le *designer*, mais ces derniers apparaissent simplement, en fonction de la nature du support : dans ce réseau de lieux sur lesquels nous distribuons aléatoirement des pièces de matière[75], la relation de cet agent avec son support, comporte maintenant un paramètre nouveau, la *direction*. Ceci induit immédiatement et naturellement[76] la naissance d'un nouvel actionneur chargé de la faire varier : le changement de direction.

```
Case * pCase_QuelleEstLaCaseDevantToi_pAgent(Agent * pAgent) {// retourne un ptr sur la case devant l'agent

 Case * pCaseIci= pAgent->pCaseOuIAm;                    // Avec le ptr sur l'agent, retrouve la case oÃ¹ il est
 Case * pNxtCase;                                        // Tampon : Un ptr sur la case qui est devant

 switch (pAgent->Direction) {                            // En fonction de la direction de l'agent
  case nor : pNxtCase= pCaseIci->pNxtCaseNor; break;     // met le ptr sur la case qui est devant dans le tampon
  case sud : pNxtCase= pCaseIci->pNxtCaseSud; break;
```

75 Dans un lieu nous avons besoin de mettre un objet (ob-ject : la chose qui est jetée ci-devant). Vu que la matière est un substantif, afin de l'objectiver, nous devons en prendre un bout : une pièce.

76 La justification de cette affirmation constituerait en fait le sujet d'une thèse, mais ce serait là, de la recherche fondamentale !

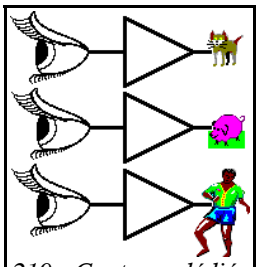
```
  case est : pNxtCase= pCaseIci->pNxtCaseEst; break;
  case wst : pNxtCase= pCaseIci->pNxtCaseWst; break;
  default  : fprintf(stderr, "Erreur dans Avance_pAgent\n");
 }

 return pNxtCase;                                    // retourne le tampon
}
```

Table 34 : Recherche du pointeur sur la case devant l'agent, pour simulation le procès avancer.

Dans la modélisation de l'agent, justifier l'apparition des capteurs (senseurs).

Dans *2DCueilleur*, le contenu du support devient plus varié. Le lieu peut être du sol, contenir de la nourriture ou constituer un obstacle. Puisque *la fonction crée l'organe*, la nature différente des objets à percevoir induit la création d'une fonction de perception.

De plus, dans une représentation sub-symbolique, chaque objet fait apparaître un capteur dédié (illustration ci-contre). Donc, naturellement, dans notre modélisation, nous utilisons des capteurs qui signalent du sol, de la nourriture ou un obstacle.



219 - Capteurs dédiés

3) Présenter les résultats (sans les interpréter)

3.A) Lancement, présentation et utilisation du programme

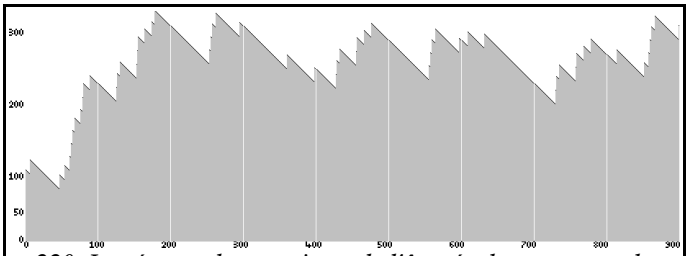
Dans *2DCueilleur*, ces différentes rubriques demeurent inchangées. Pour obtenir plus de détails se reporter à *1DCueilleur*.

3.B) Résultats

Affichage du niveau de la réserve de nourriture.

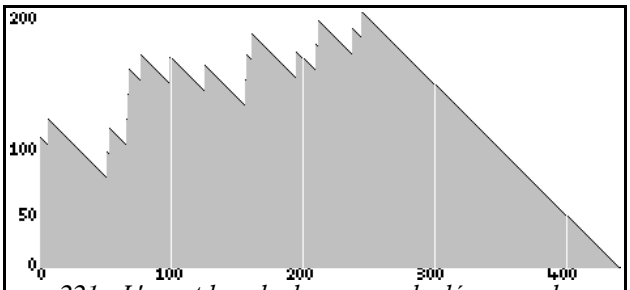
L'agent situé cueille et mange la nourriture trouvée sur son chemin.

Sur les histogrammes, les sauts vers le haut du niveau de l'énergie correspondent à la prise d'un repas par l'agent. Les pentes descendant régulièrement correspondent à la consommation journalière de nourriture par le métabolisme de l'être.



220 La réserve de nourriture de l'être évolue au cours du temps

Le problème de la pérennité de l'être.



221 - L'agent boucle dans un cycle dépourvu de nourriture

Puisque la nourriture est distribuée aléatoirement sur le support, l'agent mange irrégulièrement. Ainsi apparaît la question de sa pérennité.

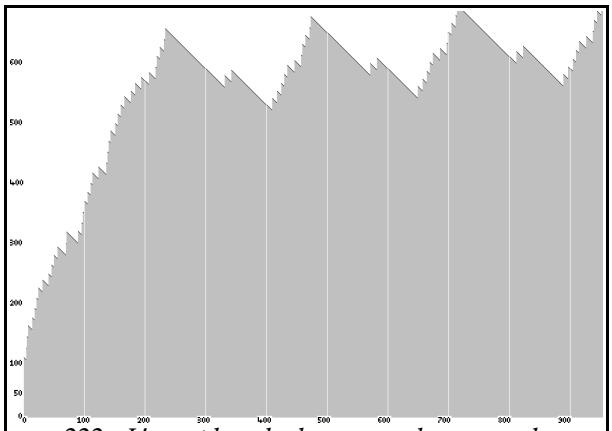
Regardons les différents histogrammes que nous obtenons, en faisant varier la distribution de nourriture et d'obstacles.

La notion de bouclage dans un cycle.

Face à un mur l'agent se comporte toujours de la même façon, ainsi, dans un support parsemé de beaucoup d'obstacles, il peut se mettre à tourner en rond, i.e. à boucler dans un même cycle. Et puisque l'agencement du monde demeure constant, il y bouclera indéfiniment.

Boucle dans un cycle dépourvu de nourriture.

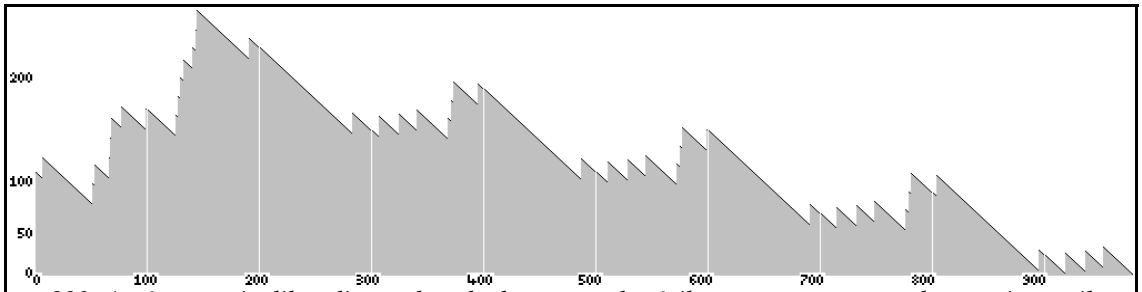
Quand il est pris dans un cycle, sa survie dépend de la longueur du parcours. La première question est de savoir si l'herbe a le temps de repousser avant que l'agent ne repasse la manger. Dans la figure ci-dessous l'agent boucle sur un parcours où il ne trouve pas de nourriture. Alors nous le voyons régulièrement péricliter, puis disparaître : il meurt de faim.



222 - L'agent boucle dans un cycle pourvu de nourriture

Boucle dans un cycle avec nourriture.

La figure ci-contre illustre le cas d'un agent qui grossit rapidement avant de boucler dans un cycle où il trouve suffisamment de nourriture pour survivre et même continuer de grossir.



223- Après un trajet libre, l'agent boucle dans un cycle où il ne trouve pas assez de nourriture : il dépérit.

À l'inverse, ci-dessus la Fig 8 illustre le cas d'un agent qui se promène librement avant de boucler dans un cycle où il trouve trop peu de nourriture pour survivre : il va finir par mourir.

Danger de la quantification.

Pour l'instant, nous sommes dans des domaines peu contraints. Beaucoup de paramètres utilisés demeurent encore libres : la densité de la distribution de nourriture et d'obstacles, le coût de l'action, la valeur calorifique de la nourriture, nous pouvons les faire varier à notre gré et inverser le cours des choses. Ainsi, sans changer la topologie d'un réseau, seulement en augmentant la valeur calorique de la nourriture ou en diminuant le coût de l'action, nous pouvons faire vivre indéfiniment, un être qui périclite le long d'un parcours en boucle. C'est pourquoi ici, il est prématuré d'aller plus loin dans la quantification. Ceci explique que nous avons évité les quantifications et sommes restés à des approches qualitatives.

4) Conclusion : vers une complexification

Avec *2DCueilleur*, nous introduisons la dimension 2, la surface, qui fait apparaître les directions et augmente la complexité de l'agent. Ainsi nous arrivons à un niveau où le modèle possède trois procès : *avancer, manger, tourner.*

5) Discussion : l'action de l'agent réactif est-il sous-tendu par un but

5.A) Définition de la notion de procès

De façon générale, par procès, nous entendons une opération qui change l'ordre du monde ; et, de façon particulière dans ce mémoire, nous avons décrit un changement d'état du monde, sous la forme d'une variation des relations entre le support et son mobilier (les agents ou les pièces de matière).

Apparition du procès comme modification des relations au sein du support.

Dans *1DCorridor*, quand l'agent change de place, apparaît le premier procès. Ensuite, dans *1DCueilleur*, quand l'agent change la place de la nourriture, apparaît le deuxième, : il la prend sur la case devant lui pour la mettre dans une poche, au contact de la cellule végétative (i.e. dans son estomac). Enfin dans *2DCueilleur*, apparaît le troisième : l'agent tourne ou change de direction : il modifie la relation qu'il entretient avec son support.

5.B) Définition de la notion d'action

Le passage de procès à action.

Dans notre chapitre *Les concepts de base de notre travail* nous expliquons la position de la philosophie de l'action : *un procès ne peut être appelé action que s'il est sous-tendu par un but*. Au travers de ce mémoire, nous distinguons rarement l'action et son procès. Maintenant, après ces définitions, nous sommes en mesure d'expliquer plus précisément ce choix.

D'abord, chez les agents intentionnels, le but est bien visible et nous n'hésitons pas à parler d' action. Ensuite, dans ce paragraphe, nous traitons des agents réactifs : afin de pouvoir dire qu'eux aussi agissent, nous montrons que leurs comportements sont eux aussi sous-tendus par des buts.

L'agent intentionnel élémentaire.

Avant de commencer, pour la clarté de l'exposé, faisons une petite digression : dans la vie courante, un être forme un projet, un but qu'il mémorise, et qui ainsi oriente son activité.

De la même façon, en informatique, un agent intentionnel élémentaire force un drapeau interne qui ensuite oriente son fonctionnement vers un but, i.e. pour satisfaire un besoin.

2DCueilleur modélise un agent réactif combinatoire.

Par contre, dans *2DCueilleur*, nous sommes en présence d'un agent réactif bien conçu, qui traite combinatoirement l'information. À l'instar du modèle béhavioriste, *stimulus–réponse*, il agit spontanément. Par définition, il ne possède pas d'état mental interne. Il agit en fonction des états du monde externe, et aussi de ses états somatiques internes (faim, soif, chaleur, froid...).

Pas de gestion des états mentaux internes.

Puisqu'il ne possède pas d'état mental interne, il ne s'occupe pas de cette gestion. Donc, chez lui, nous ne trouvons jamais de règle qui parle explicitement de la gestions des buts, i.e. de les forcer, ou de les abandonner. Il ne tient jamais un discours du genre : *Je force cet état interne et ensuite j'agis en fonction de lui*.

5.C) Introduction de finalité implicite indirecte

Les buts sont permanents mais demeurent moins visibles.

Puisqu'il ne forme pas de but, qu'il ne force pas d'état interne, alors chez lui, ils sont soit toujours absents, soit toujours présents. S'ils sont absents, ils sont insignifiants, alors intéressons nous plutôt à ceux qui sont présents : dans ce cas ils sont permanents.

Mais, à cause de cela, puisqu'ils sont permanents, ils sont câblés intrinsèquement, implicitement dans ses règles, dans la structure *perception décision action* de l'agent. C'est alors

ce qui explique, que dans ce cas, les buts qui sous-tendent ces relations causales, sont difficilement visibles.

Cependant ces buts existent bien.

Cependant, même s'ils sont difficilement visibles, ces buts existent bien. Nos agents (*1DCorridor*, *1DCueilleur*, *2DCueilleur*) ont tous trois le même besoin de nourriture. De même, leur traitement de l'information est conséquemment orienté à le satisfaire, et ils réussissent bien dans cette finalité, ce qui constitue ainsi un but.

Des agents réactifs bien conçus.

Rappel d'une condition primordiale : nous travaillons avec des agents bien conçus, i.e. adaptés à leur milieu. Cette conception de qualité peut provenir au moins de deux sources : d'un *designer* ou du mécanisme d'évolution des espèces.

5.D) Conclusion

Introduction de la notion d'agent réactif indirectement intentionnel.

Finalement nous introduisons la notion d'agent réactif indirectement intentionnel : ces agents réactifs adaptés portent implicitement en eux-mêmes une finalité indirecte provenant de leur conception.

Conclusion : les agents réactifs indirectement intentionnels, agissent.

De simples agents réactifs, s'ils sont bien conçus, s'avèrent indirectement intentionnels : implicitement, ils sont sous-tendus par des buts. Ainsi, chaque procès qu'ils effectuent, peut s'appeler *action*, et nous pouvons dire qu'ils agissent.

6) Perspectives

Nous allons maintenant passer à des êtres plus complexes : les agents intentionnels.

Ch. 35 *1DAlternateur,* agent pulsionnel qui commute entre deux buts

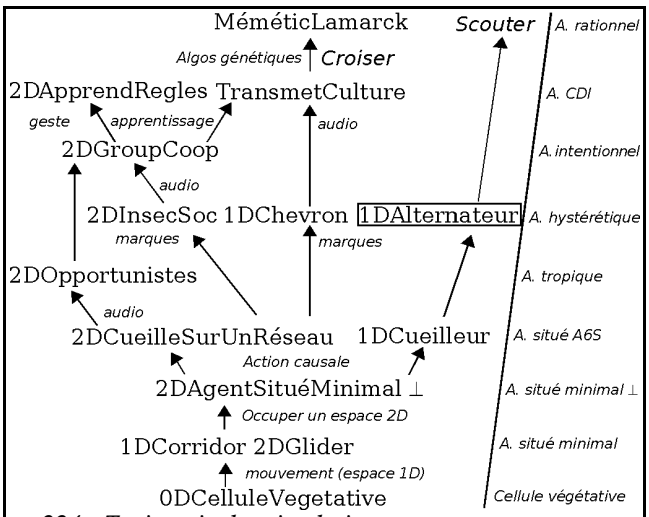
1) Introduction

1.A) Mémoriser en marquant le monde

1DAlternateur modélise l'agent hystérétique pulsionnel : le long d'un itinéraire de lieux, un cueilleur va alternativement, chercher sa nourriture à l'ouest et se désaltérer à l'est.

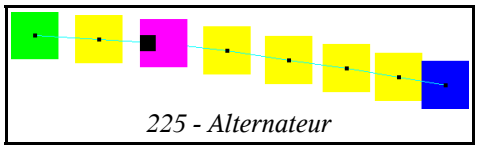
L'agent est hystérétique[77] car, quand il change de direction, il marque le monde : il mémorise une information en changeant sa propre relation avec le support (externe).

L'agent est pulsionnel car son comportement est guidé par l'assouvissement des ses pulsions internes (dans l'exemple : de faim et de soif).



224 - Taxinomie des simulations que nous proposons

1.B) Situer cette étape dans notre taxinomie


225 - Alternateur

Retour à un support 1D : un itinéraire de lieux.

D'abord, nous travaillons avec des lieux, qui au maximum, sont en relation avec deux voisins. Nous voici donc revenus sur un support 1D : l'agent se déplace le long d'un itinéraire de lieux.

Deux ressources regroupées aux extrémités du support.

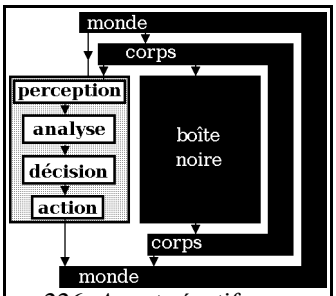
À cette étape de la progression, le changement vient surtout de la répartition des ressources. Elles étaient omniprésentes et diffuses dans les premières simulations, puis nous les voyions se concentrer progressivement dans les étapes suivantes, maintenant nous ne les trouvons que concentrées aux deux extrémités du support.

Dans le synoptique de l'agent apparaît un nouvel étage d'analyse.

Ce changement important de la configuration des ressources induit chez l'agent une adaptation de son traitement de l' information. En sortie de son étage *perception* apparaît une boîte éponyme qui *analyse* ses capteurs externes et internes.

Nous sommes en présence d'un agent hystérétique.

Cet étage analyse la situation qui provient des percepts externes et des pulsions internes de l'agent, i.e. de ses niveaux de faim et soif. Comme résultat de ce traitement, sur sa sortie, il fournit/positionne des commandes, i.e. des signaux



226- Agent réactif avec analyse

77 Selon l'étymologie, hystérétique vient du grec husterein : retarder.

neuraux/électriques internes qui basculent/oscillent entre deux démarches : aller chercher de l'énergie à l'ouest et de l'eau à l'est.

Nous sommes en présence d'un agent finaliste.

Ainsi, nous sommes en présence d'un agent clairement finaliste, car il perçoit deux buts, et les analyse, ce qui orientent son action de façon diamétralement opposée.

Nous sommes en présence d'un agent réactif.

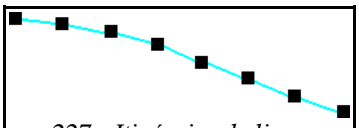
L'agent demeure réactif car chaque décision qui commande une action n'est pas mémorisée dans un planning, pour être ensuite interprétée. Au contraire, elle est décidée en temps réel, par l'analyse de ses perceptions, et elle active immédiatement un actionneur.

2) Matériels, méthodes et implémentation

2.A) Quels outils informatiques pour modéliser cette étape ?

Modélisation du support.

Pour modéliser le support de cette simulation, nous utilisons un itinéraire de lieux, implémenté par un tableau 1D de cases, munies d'un pointeur sur chacun de leurs deux voisins (horizontaux). Mais cette fois, nous renseignons à la fois les pointeurs e*st* et *ouest,* car l'agent transite dans ces deux sens.



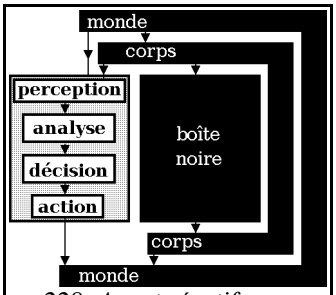
227 - Itinéraire de lieux

Pour l'agent qui progresse vers l'extérieur, les deux cases aux extrémités de l'itinéraire possèdent des pointeurs sur elles-mêmes. Alors, quand il est tourné ainsi, il n'avance plus, et demeure bloqué à leur niveau.

Modéliser l'agent pulsionnel comme un agent réactif enrichi.

Le traitement de l'information effectué par l'être demeure simple. Nous retrouvons le synoptique de l'agent réactif : *perception décision action*, auquel nous ajoutons un étage d'analyse.

Le procès traite seulement quelques capteurs et n'active que quelques actionneurs. Cependant, il ne peut plus être effectué au moyen de table, à cause des fonctions de comparaisons. Dans l'implémentation électronique, il faut câbler un comparateur et dans le code informatique, il est déjà effectué au moyen de règles :



228- Agent réactif avec analyse

```
void Agent_Agis(Agent * pAgent) {
 ...
 ChoseDevantNature = pNxtCase->Nature;                    //Tamponnement d'1 type résultant d'1 fonc°

 if                                                       // L'agent ne mange que s'il a faim
  (
   (ChoseDevantNature == food)                            // comparaison entre deux symboles
   &&                                                     // fonction logique ET
   (pAgent->EnergyLevel < SeuilDeSatiete)                 // Comparateur numérique
  )
  Mange_pAgent(pAgent);                                   // activation de l'effecteur
 else if                                                  // L'agent ne boit que s'il a soif
  (                                                       // Si devant lui l'agent perçoit de l'eau
   (ChoseDevantNature == eau)                             // devant lui l'agent perçoit de l'eau
   &&                                                     // et
   (pAgent->EauLevel < SeuilDeSatiete)                    // si l'agent a soif
  )
  Bois_pAgent(pAgent);                                    // alors il boit
 else if (ChoseDevantNature==sol)Avance_pAgent(pAgent); // sinon, si c'est possible, il avance
 else fprintf(stderr, "L'agent reste sur place\n");
}
```

Table 35 : Codage de l'activation des actionneurs

Modéliser l'agent pulsionnel comme la commutation de deux agents réactifs.

Dans *1DAlternateur*, l'agent fonctionne selon deux registres : la recherche d'eau et celle de nourriture. Il présente deux comportements sous-tendus par deux buts : obtenir de l'eau et obtenir de l'énergie.

L'agent réactif analyse ses états internes *énergie* et *eau*, en les comparant à des niveau d'alerte. Quand l'un de ces indicateurs passe en dessous du seuil de référence, l'agent change de direction pour rallier le lieu où obtiendra cette ressource.

```
if
 (pAgent->EnergyLevel < AlerteLevel)     // Comparateur numérique
 {
  pAgent->Direction= wst;                // Si dessous le niveau d'alerte alors change de direct°
 }                                       // Si il a besoin d'énergie, il fixe son cap a l'ouest
else if
 (pAgent->EauLevel < AlerteLevel)        // Comparateur numérique
 {
  pAgent->Direction= est;                // Si dessous le niveau d'alerte, alors change de direct°
 }                                       // Si il a besoin d'eau, alors il fixe son cap à l'est
```

Table 36 : La simulation informatique comprend des règles qui utilisent des comparateurs numériques.

Chaque ressource, située aux deux extrémités du chemin de lieux, conditionne les actions de l'agent. Quand il fait demi-tour, il se détourne de la ressource qu'il consommait, et ne la voit plus. Il avance et implicitement se dirige vers l'autre. Il la consommera quand elle apparaîtra devant lui.

```
if                                                  // L'agent ne mange que s'il a faim
 (
  (ChoseDevantNature == food)                       // comparaison entre deux symboles
  &&                                                // fonction logique ET
  (pAgent->EnergyLevel < SeuilDeSatiete)            // Comparateur numérique
 )
 Mange_pAgent(pAgent);                              // activation de l'effecteur
else if                                             // L'agent ne boit que s'il a soif
 (                                                  // Si devant lui l'agent perçoit de l'eau
  (ChoseDevantNature == eau)                        // devant lui l'agent perçoit de l'eau
  &&                                                // et
  (pAgent->EauLevel < SeuilDeSatiete)               // si l'agent a soif
 )
 Bois_pAgent(pAgent);                               // alors il boit
```

Table 37 : L'agent ne consomme une ressource que si elle apparaît devant lui.

Nous sommes en présence d'un agent réactif, qui analyse ses états internes. Il commute son fonctionnement, en forçant un état externe[78], qui ensuite oriente ses actions entre deux comportements, deux quêtes : nourriture et eau.

3) Présenter les résultats (sans les interpréter)

3.A) Lancement, présentation et utilisation du programme

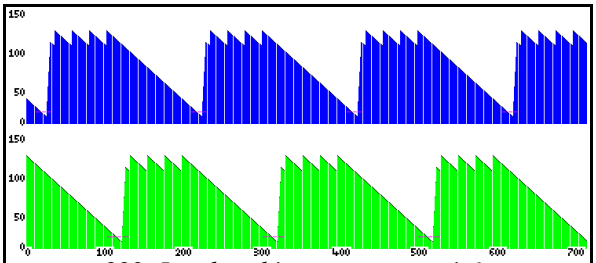
Dans *1DAlternateur*, ces rubriques demeurent inchangées. Pour obtenir plus de détails se reporter à *1DCueilleur*.

3.B) Résultats

Affichage du niveau des réserves de l'agent.

La figure de droite présente les histogrammes des réserves de l'agent : en haut, en bleu : sa réserve d'eau. En bas, en vert : sa réserve de nourriture.

Analysons le cycle d'un histogramme.



229- Les deux histogrammes croisés des réserves de l'agent

78 Cet état est externe : il n'est pas mémorisé au sein de [illegible] a direction, qui consiste en une relation entre lui et son support.

Puisque nourriture et eau ne manquent pas, l'agent effectue un cycle régulier, les histogrammes sont périodiques. Enfin, comme l'agent alterne entre les deux ressources, ses cycles sont déphasés d'une demi-période.

Analysons en détail le cycle d'un histogramme.

Comme les ressources *eau* et *énergie* jouent le même rôle les cycles des deux histogrammes sont duals et identiques, nous choisissons d'étudier le premier cycle vert, car il est plus lisible.

- Au départ, à la date 0, l'agent ne mange pas et sa réserve du nourriture diminue.
- Après avoir passé la date 100, elle passe sous le niveau 16, qui correspond au seuil d'alerte (segment de droite magenta).
- L'agent commute et se tourne vers l'ouest.
- En 6 cycles d'horloge, il traverse l'itinéraire de lieux. Pendant ce trajet, il jeûne encore, et la courbe continue de descendre jusqu'au niveau 10. Enfin il arrive devant la nourriture.
- Pendant 5 cycles d'horloge, il mange une nourriture qui vaut 20 unités, mais il en consomme 5*1 pour son métabolisme. Partit du niveau 10, il arrive au niveau : 10 + (5*19) = 105.
- Comme le seuil de satiété est 100, il demeure en place et ne mange pas pendant 4 cycles, ce qui le fait descendre du niveau 104 au niveau 101.
- Arrivé à 100, il mange et passe à 110 et le cycle recommence : pendant 3 cycles de 10 unités de temps, il reste sur/devant la ressource et mange dès qu'il ne se sent pas rassasié.
- Mais pendant qu'il mange, il ne boit pas et sa réserve d'eau baisse régulièrement, au point de déshydrater son métabolisme.
- Un peu après la date 200, l'agent a soif, et commute son fonctionnement : il se tourne à l'est et part vers l'eau.
- Pendant ce temps, sa réserve d'énergie diminue, jusqu'au seuil critique où ...
- Et la boucle est bouclée, le cycle recommence comme au départ.

4) Conclusions par rapport à différentes sémantiques

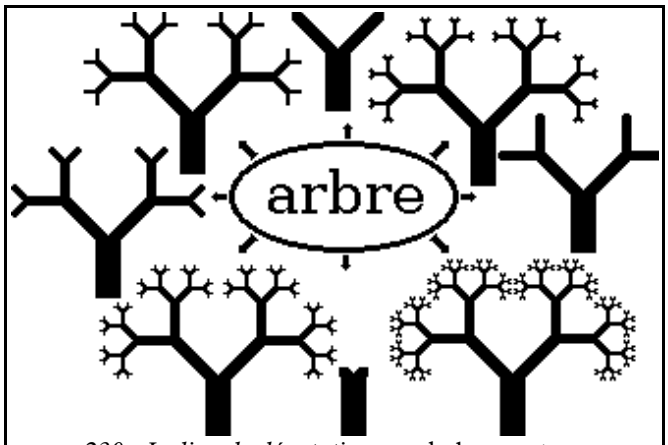
4.A) Introduction

Maintenant nous disposons d'un modèle qui va nous permettre de toucher du doigt certains phénomènes informatiques et sémantiques, que nous allons examiner afin de tirer quelques conclusions.

4.B) Sémantique dénotationnelle

Définition de cette sémantique.

Elle traite du lien dénotationnel qui unit le symbole de référence à son contenu, i.e. au contenu auquel il réfère. Dans cette mouvance Frege et Russel ont beaucoup travaillé sur le problème dit du *contenu*. Ce problème apparaît dans la vie quotidienne lors de l'analyse et la description d'une scène. Le spectateur regarde une photo et énonce : *arbre*. Le fait d'entendre un tel message nous permet d'en savoir bien plus à propos de cette scène, mais en fait, cette information est partielle car il existe des milliards d'arbres qui peuvent rentrer dans cette catégorie. Le problème de la dénotation vient de ce que l'énoncé du symbole ne suffit pas à capturer la totalité de l'information. Nous sommes dans un démarche imparfaite.



230 - Le lien de dénotation symbole – contenu

L'exemple décrit ci-dessus illustre le fait que le même symbole peut dénoter des objets bien différents.

Remarque : nous avons illustré notre propos au moyen d'un dessin où le contenu du symbole est un objet, mais le problème se pose pareillement quand il s'agit de dénoter une action sur le monde.

Se positionner pour éviter les problèmes de dénotation.

Le lecteur doit être conscient qu'à l'intérieur de toutes les simulations de cette progression, nous nous somme positionnés clairement pour éviter les problèmes de sémantique dénotationnelle. C'est ce qui justifie la construction de notre représentation de la connaissance. Ceci explique pourquoi, sur le support, au sein d'un lieu, nous plaçons 'une unique pièce de matière. Ainsi, en utilisant ce support digital, nous évacuons intentionnellement les problèmes de sémantique dénotationnelle dans nos modèles.

Résultat.

Nous avons enlevé le maximum d'éléments de l'intérieur de nos modèles ; en faisant attention de *ne pas jeter le bébé avec l'eau du bain*. Alors, à ce prix, nous évitons de nous retrouver avec une coquille vide. Nos modèles conservent un sens, et nous y obtenons toujours un procès qui est bien sous-tendu par un but.

Perspectives.

Ces modélisations, tellement simples que le profane pourrait les qualifier de simplistes, constituent une plate-forme très épurée, qui va nous permettre ensuite de monter en pression, c'est à dire de construire des objets et des situations de plus en plus complexes. Le but est de provoquer progressivement tous ces problèmes de dénotation et de les disséquer afin de les étudier.

Par exemple, voici une idée de simulation facilement réalisable avec des agents situés et qui amènerait à toucher du doigt les problèmes de dénotation et de contenu. Il s'agit tout simplement de disposer des pierres sur une grille de lieux et de travailler sur la notion d'enclos, et les notions qui en sont proches : mur, porte, passage.

4.C) Sémantique de l'efficacité d'une fonction, d'une couche, d'une structure

a) Introduction : l'efficacité d'une couche au sein d'une structure

Après avoir parlé de la sémantique dénotationnelle... pour dire que nous n'en parlerions plus, voici le moment de poser la base de notre travail théorique. Nous souhaitons définir la sémantique d'une fonctionnalité, que nous réutiliserons tout au long de ce mémoire.

Avec *1DAlternateur*, nous obtenons un modèle d'agent pulsionnel. Avec lui, nous sommes en présence d'un être, dont le comportement est sous-tendu par deux buts. Ces derniers commutent sa programmation entre deux registres de fonctionnement, deux efficacités. Ceci nous amène à traiter le problème de l'efficacité d'une couche au sein d'une structure.

b) Mesurer l'efficacité d'une structure

Nous avons déjà évoqué ce problème, et dit qu'il était en général complexe, mais nous avons expliqué notre façon simple de la traiter, en s'inspirant des mécanismes de la vie artificielle : *c'est au fruit qu'on juge l'arbre*, et nous mesurons l'efficacité d'une structure au fait que l'être qui lui correspond, survive. Plus concrètement, par exemple, ceci peut être fait en observant au fil du temps, l'évolution de la taille de la population d'un groupe d'agents qui l'utilisent.

c) Définir la sémantique d'une couche ajoutée au sein d'une structure initiale

Afin de déterminer un sens, il nous faut disposer de deux mesures.

Au départ, à l'étape N, nous disposons d'une structure dont nous avons évalué l'efficacité. Nous disposons ainsi d'une mesure M_0 de référence de son efficacité. Ensuite, à l'étape N+1, nous ajoutons à cette structure initiale, une nouvelle couche ou fonctionnalité, puis nous utilisons la même méthode pour mesurer l'efficacité de la structure améliorée. Ainsi nous disposons d'une seconde mesure M_1.

Définition du sens d'une couche ajoutée au sein d'une structure initiale.

Par définition nous dirons que le sens S_{10} de cette couche ajoutée à la structure initiale est la différence entre les deux mesures : $S_{10} = M_1 - M_0$. En effet, si la nouvelle couche apporte une amélioration au modèle initial, elle aura du sens, et S_{10} sera positif. Si cette couche ne change rien, alors elle n'a pas de sens et $S_{10} = 0$. Finalement, si elle va dans le mauvais sens, S_{10} sera négatif.

4.D) Perspectives

Voici que, dans le cadre d'un groupe d'agents en situation sur un support, nous avons formalisé cette sémantique de l'efficacité d'une nouvelle couche.

Dans la suite de ce mémoire nous réutiliserons plusieurs fois ce résultat. En effet, à chacune des étapes de la progression, quand nous introduirons une nouvelle couche, nous vérifierons son intérêt, au moyen de cette nouvelle méthode.

Ch. 36 *2DOpportunistes* : le groupe et la communication passive

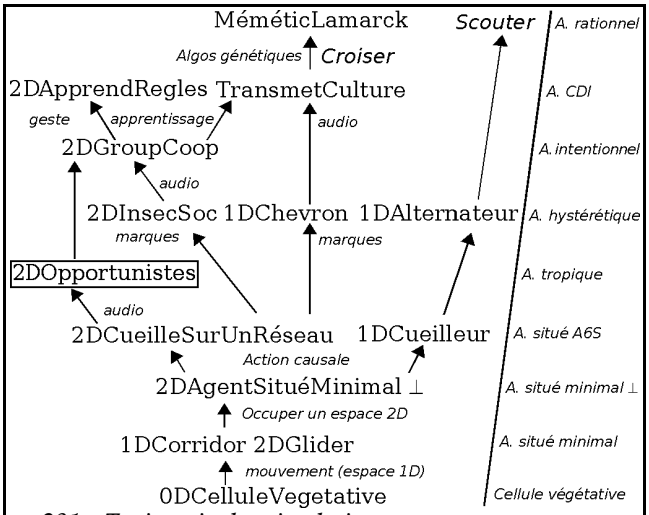
1) Introduction

1.A) Situer cette étape dans notre taxinomie

Cette étape nommée *2DOpportunistes* ne constitue pas une simulation de la vie artificielle. Au contraire, ce chapitre théorique introduit le groupe et la communication, qui va avec. Ainsi il présente un nouveau mode de fonctionnement qui est repris ensuite dans plusieurs autres simulations.

Ce chapitre expose deux aspects : d'abord il explique comment implémenter une communication basique avec des agents réactifs. Ensuite, il présente le phénomène de groupe qui en résulte.

Cette simulation est implémentée selon les mécanismes proposés par Braitenberg.



231 - Taxinomie des simulations que nous proposons

Dans cette organisation d'une communication passive, les agents sont répartis sur le support. Au centre un agent fait du bruit. Les autres se tournent vers lui et les rejoignent.

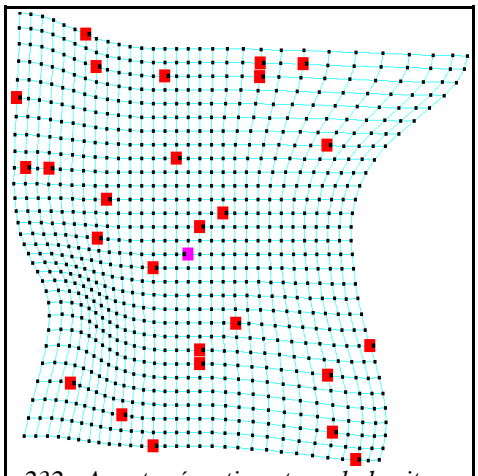
1.B) Le groupe induit la communication

Introduction du groupe.

Si le grain ne meurt : jusqu'à maintenant, dans nos modélisations, quand un individu se reproduisait, alors il mourrait. Dès que cette règle est remplacée par une reproduction qui ne fait plus disparaître le parent, sur le support, le nombre d'agents augmente et nous nous retrouvons avec un groupe.

Introduction de la communication.

L'introduction du groupe induit rapidement l'apparition de la communication. En effet, quand les agents au sein d'un groupe coopèrent, ils augmentent fortement l'efficacité de l'ensemble. Cette amélioration est rapidement sélectionnée puis renforcée par le mécanisme d'évolution des espèces.


232 - Agents répartis autour du bruiteur

La démarche de regroupement.

Nous distinguons deux démarches de regroupement : passive et active. Dans la démarche passive, le groupe se tourne et rejoint celui qui fait du bruit involontairement. Nous étudierons plus tard la démarche active, où le groupe rejoint l'agent qui appelle volontairement les autres.

2) Matériels, méthodes et implémentation

2.A) Quels outils informatiques pour modéliser cette étape ?

a) Modéliser le monde (support + mobilier) : agents situés sur un réseau 2D de lieux

Nous voici dans un domaine 2D : nous travaillons avec des lieux, qui sont, au maximum, en relation avec quatre voisins. Au sein de ce monde, les agents rouges sont distribués sur un réseau d'endroits, autour d'un agent central de couleur magenta, qui fait du bruit.

b) Modélisation de l'agent tropique comme un agent réactif enrichi

Dans cette simulation, le traitement de l'information implémenté demeure simple. À la base, chaque être est un agent réactif enrichi : nous lui ajoutons un étage d'écholocation. Il est muni de capteurs pour entendre le bruit que fait l'agent central, et d'un circuit pour traiter cette information.

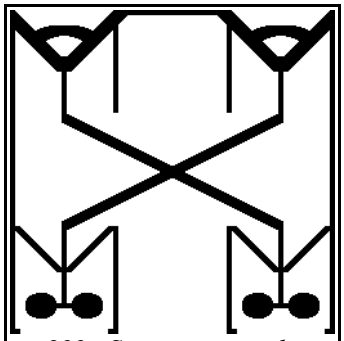
c) Un étage d'écholocation, à l'instar de Braitenberg

L'agent réactif de Braitenberg.

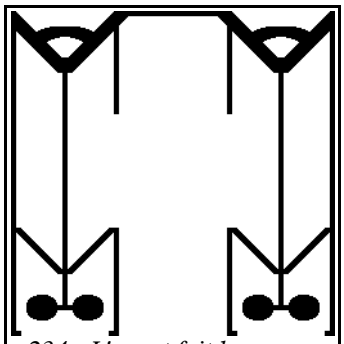
Ci-contre un agent réactif. Il est tourné vers le haut de la page. Il possède deux capteurs globaux (en forme d'œil) sur la face avant, et sa propulsion est assurée par deux hélices en face arrière (au bas du schéma).

L'agent réactif se tourne vers la source .

Chez cet agent le câblage est croisé : le capteur de gauche est connecté au moteur de droite, et celui de droite est connecté à gauche. Supposons que cet agent soit placé dans le champ d'une source réelle situé devant et à droite, dont la loi de propagation décroît avec la distance (en 1/x ou en $1/x^2$ selon le modèle). Disons que le capteur de droite reçoit plus de stimulation que le gauche. Braitenberg montre que dans ce cas, le moteur gauche est plus activé que le droit : le différentiel de propulsion fait tourner l'agent vers la source.



233 - Se tourner vers la source



234 - L'agent fuit la source

Le montage possède une symétrie axiale et le même raisonnement effectué avec une source à gauche, déboucherait sur une conclusion duale.

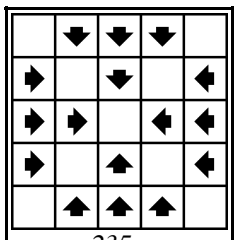
L'agent réactif fuit la source (ci-contre à gauche).

Chez cet agent le câblage est droit : chaque capteur active le moteur de son côté. Quand cet agent est placé dans le champ de la même source, et le capteur de droite reçoit toujours plus de stimulation que le gauche. Alors dans ce cas, le moteur droit est plus activé que le gauche et l'agent se détourne de la source.

Le même raisonnement effectué avec une source à gauche, déboucherait sur une conclusion duale.

Conclusion.

Notre agent réactif, enrichi du montage de Braitenberg, mis en présence d'une source sonore est donc capable d'écholocation grossière, mais cette aptitude suffit à lui conférer un avantage sélectif.



235 - Héliotropisme

d) Implémentation

Selon un point de vue informatique, dans cette simulation, nous n'avons pas implémenté finement l'agent de Braitenberg, nous n'avons pas simulé un signal, émis par l'appelant, qui se propage au sein du réseau, pour être ensuite reçu et traité par chaque agent ; nous avons seulement simulé tout cet échange et codé le résultat, i.e. le fait que chaque agent se tourne vers l'appelant : en fonction de sa position relative à l'appelant, l'agent suiveur se tourne vers l'émetteur et ensuite, avance vers lui. C'est ainsi, que graduellement il le rejoint, s'il ne butte pas contre un autre. Dans ce cas, tant que dure la collision, il est bloqué et demeure immobile.

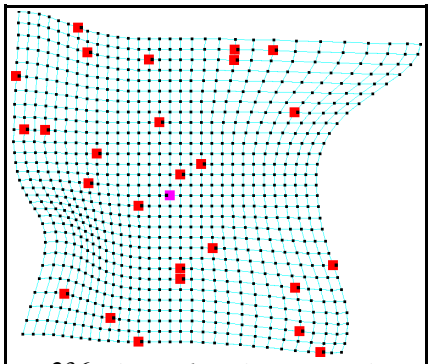
```
void AgentSuiveurTourneToiVers7AgentNatriarche_SilEstLoinAvance(Agent * pAgentSuiveur, Agent * pAgentNatriarche) {
                                                    // En parametre, je reçois les pointeurs sur les 2 agents
 int Pos_I= pAgentNatriarche->pCaseOuIAm->InxI;     // Au moyen des pointeurs sur les deux agents
 int Pos_J= pAgentNatriarche->pCaseOuIAm->InxJ;     // je recupere les cases ou ils sont dans le tableau, puis
 int Pos_i= pAgentSuiveur->pCaseOuIAm->InxI;        // recupere les coord J et I de cette case dans le tableau
 int Pos_j= pAgentSuiveur->pCaseOuIAm->InxJ;        // j'obtiens donc leur coord J et I dans le tableau

 fprintf(stderr, "\n J:%d I:%d", Pos_J, Pos_I);     // 3 Affichages pour mieux suivre le deroulement du prog
 fprintf(stderr, "\n j:%d i:%d", Pos_j, Pos_i);
 fprintf(stderr, "\n Pos_I:%d Pos_i:%d", Pos_I, Pos_i);

// Pour calculer les deltas, on se centre sur le Natriarche (qui est au centre),
 int Delta_J= Pos_j - Pos_J;                        // donc on ôte ses coord a celle du suiveur
 int Delta_I= Pos_i - Pos_I;                        //      -1 0 +1         //le Natriarche est au centre

 fprintf(stderr, "\n Delta_J:%d Delta_I:%d", Delta_J, Delta_I);//Affichage pr mieux suivre le deroulement du prog
                                                   // le suiveur se tourne selon sa position relativement au natriarche
 if ((Delta_I>0) && (abs(Delta_J)< Delta_I)) { pAgentSuiveur->Direction= wst; fprintf(stderr, "fcap wst\n"); }
 if ((Delta_I<0) && (abs(Delta_J)<-Delta_I)) { pAgentSuiveur->Direction= est; fprintf(stderr, "fcap est\n"); }
 if ((Delta_J<0) && (abs(Delta_I)<-Delta_J)) { pAgentSuiveur->Direction= sud; fprintf(stderr, "fcap sud\n"); }
 if ((Delta_J>0) && (abs(Delta_I)< Delta_J)) { pAgentSuiveur->Direction= nor; fprintf(stderr, "fcap nor\n"); }
}
```

Table 38 : Sous programme dans lequel l'agent suiveur se tourne vers l'agent appelant (le natriarche)

3) Présenter les résultats

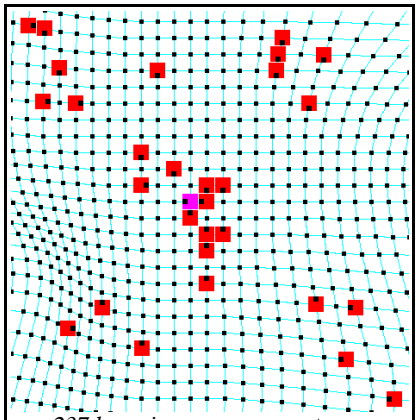
3.A) Lancer, présenter et utiliser le programme

Dans *2DOpportunistes*, ces rubriques demeurent inchangées. Pour obtenir plus de détails se reporter à *1DCueilleur*.



236 - Avant : les suiveurs répartis autour de l'appelant

3.B) Résultats



237 les suiveurs convergent vers l'appelant

Au départ (ci-dessus à droite).

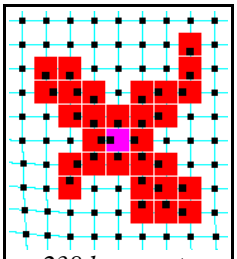
L'agent appelant, en magenta, est placé au centre du réseau de lieux. Les agents suiveurs, en rouge, sont distribués aléatoirement autour de lui. Leur direction, marquée par un carré noir excentré, est aussi aléatoire.

Les agents suiveurs convergent vers l'appelant.

Sur l'image de gauche, dès que l'appelant commence à convier les suiveurs, ils se tournent vers lui et avancent. À chaque cycle ils se rapprochent de lui, et ensemble, convergent rapidement vers le centre.

À l'arrivée (ci-contre, à droite).

Finalement, à l'arrivée, les agents se trouvent groupés autour de l'appelant.



238 les agents arrivés

3.C) Interprétation des résultats

L'attroupement final en forme de X.

À l'arrivée, les agent se regroupent autour de l'appelant. Théoriquement, ils devraient s'organiser en forme de patatoïde, mais nous obtenons plutôt une forme en X. Ceci est dû à l'algorithme d'orientation des agents : il les oriente vers les diagonales. Le problème peut être corrigé en rajoutant du code : si un agent est bloqué, il se tourner sur le côté afin de regarder s'il trouve une place libre, et avance. Cette méthode est implémentée dans la simulation *Croiser*.

Pas de quantification.

Dans cette simulation encore, nous ne trouvons pas de quantification, parce qu'elle ne serait guère utile. Nous sommes en train de construire une progression de modèles, et les étapes que nous construisons servent à justifier, à expliquer le sens de l'édifice.

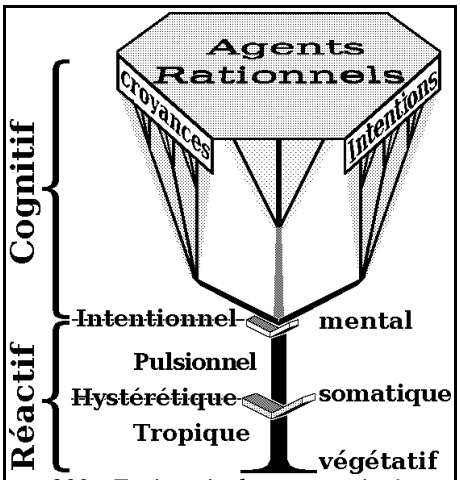
4) Conclusion : un agent tropique bien simple

Un agent tropique dans les deux sens du terme.

Avec cet être nous disposons d'un agent tropique dans les deux sens du terme. D'abord il est tropique dans le sens où c'est un agent réactif non hystérétique. Ensuite, il est tropique dans le sens où, comme le tournesol, il est héliotrope.

Un agent tropique que nous réutiliserons.

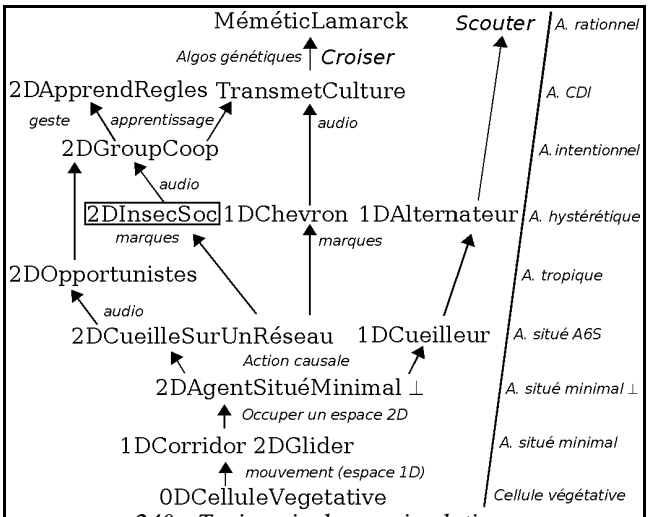
Avec cet agent réactif tropique, nous disposons d'un être très simple que nous réutiliserons dans nos modélisations *2DGroupCoop* et *2DApprendRegles*, afin de construire des êtres plus complexes.



239 - Taxinomie des agents situés

Ch. 37 *2DInsecSoc* : la communication active du groupe

1) Introduction

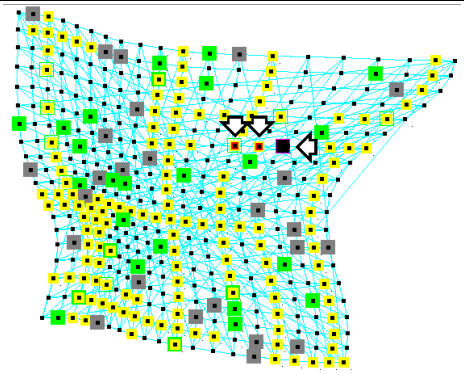


240 - Taxinomie de nos simulations

Initier la communication volontaire.

À l'étape précédente, en introduisant le groupe, nous avons initiée la communication. Dans ce premier modèle *2DOpportunistes*, celle-ci était d'abord passive, i.e. involontaire.

À cette étape, nous introduisons une communication active, où l'agent marque volontairement le support à l'intention du groupe : il effectue ce geste car une règle de son comportement le lui commande.


241 L'agent marque le support avec une hormone

Le groupe induit le début de la communication.

Cette étape nommée *2DInsecSoc*, pour insectes sociaux, continue de travailler avec le groupe et prolonge la communication : dans la première étape, elle état passive et involontaire, mais maintenant les agents communiquent volontairement : Dans cette simulation, l'agent (flèche horizontale) décide de marquer le support au moyen d'une hormone de stress qui indique à tous un danger (les deux cases rouges, dessous les flèches). Quand un agent trouve cette marque, il détourne son chemin.

2) Matériels, méthodes et implémentation

2.A) Quels outils informatiques pour modéliser cette étape ?

Modélisation du support : introduction d'obstacles.

Nous reprenons le domaine 2D habituel, mais cette fois-ci nous introduisons des obstacles, des cases grises, distribuées aléatoirement sur le support.

Modélisation de l'agent pulsionnel comme un agent réactif enrichi.

Le traitement de l'information effectué par l'être demeure simple. À la base, chaque être est un agent réactif enrichi de deux fonctions. D'abord nous lui ajoutons un actionneur qui lui sert à marquer d'une phéromone, le lieu où il se tient. Ensuite, nous l'équipons d'un capteur, afin de détecter cette phéromone sur la case devant lui.

2.B) Simuler la communication par phéromones

a) Quel scénario traiter pour illustrer cette communication ?

Marquage d'une zone dangereuse au moyen d'une phéromone de stress.

À l'instar des fourmis, les exemples classiquement présentés en IAD[79] traitent le marquage de trajets vers la nourriture. Dans ce travail, nous ne pouvons pas reproduire des exemples trop souvent traités, c'est ainsi que nous avons choisi d'illustrer un fonctionnement selon un registre différent : un agent marque une zone dangereuse, à l'intention de ses congénères. De plus nous avons choisi de simuler un cas original, où le marquage que l'agent laisse pour ses semblables, finalement sert à lui-même :

La sortie d'un cycle dangereux au moyen de marquage par phéromone de stress.

Dans ce scénario, l'agent réactif, en situation sur un réseau garni d'obstacles, est pris dans un cercle vicieux. En effet : en présence des mêmes obstacles, il effectue mécaniquement le même traitement. Dans notre cas, ceci l'amène à boucler dans un cycle perpétuel.

De plus cette boucle est très courte. La nourriture, une fois mangée, n'a pas le temps de repousser : l'être périclite. Cependant, quand il commence à souffrir de la faim, il marque son passage d'une hormone de stress et, après avoir effectué un cycle de plus, quand il retombe sur ce marqueur de danger, afin de l'éviter, il tourne, et ainsi sort du cycle vicieux.

b) Implémentation de ce scénario

Marquage des cases au moyen d'une hormone de stress.

Chaque case du support comporte un champ *StressLevel* et si le niveau d'énergie de l'agent *EnergyLevel* descend au-dessous d'un seuil minimum, quand nous avançons, nous forçons ce champ au niveau 1.

```
void Avance_pAgent(Agent * pAgent) {// pour avancer l'agent jusqu'à la case suivante
...
                                          // Si l'être est stressé, il le marque.
 pCaseIci->StressLevel= (pAgent->EnergyLevel < StressEnergyLevel) ? 1 : 0;
...
}
```

Tab 39 Le marquage des cases avec la phéromone de stress est fait dans la procédure Avance_pAgent

Évitement d'une case marquée par une phéromone.

Il se fait au niveau du jeu de règles de l'agent. L'agent regarde d'abord si la case devant lui est marquée par le stress, et si c'est le cas, il tourne à droite.

```
void Agent_Agis_A7Date(Agent * pAgent, int date) {
...
 pAgent->EnergyLevel-= ActionEnergyCost;                  // Traite la consommation de l'énergie
                                                          // Si la case devant est marquée par le
stress
 if (pNxtCase->StressLevel) TourneDroite_pAgent (pAgent);  //  alors l'agent tourne à droite
 else if ...
...
}
```

Table 40 : L'évitement d'une case stressé se fait au niveau du jeu de règles de l'agent

3) Présenter les résultats

3.A) Lancement, présentation et utilisation du programme

Dans *2DInsecSoc*, ces rubriques demeurent inchangées. Pour obtenir plus de détails se reporter à *1DCueilleur*.

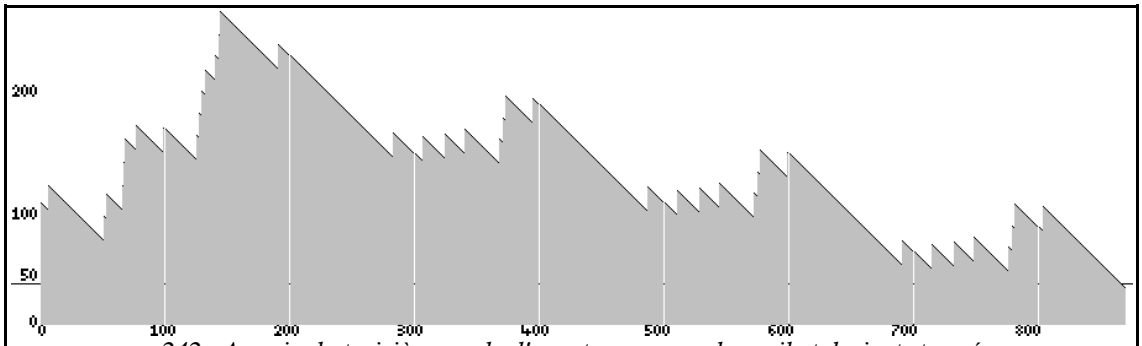
79 IAD : intelligence artificielle distribuée.

3.B) Résultats

Pour illustrer ces résultats nous présenterons une session particulière.

L'agent se met à boucler.

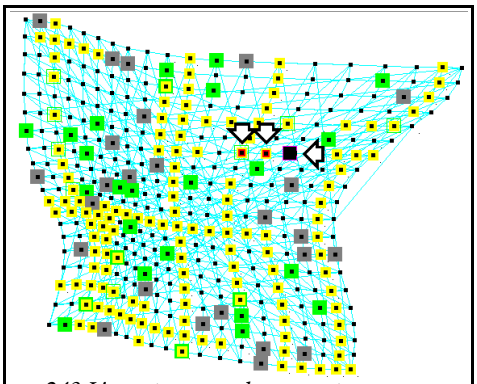
L'agent réactif effectue toujours les mêmes actions et entre dans un cycle dont il ne ressort pas. Nous avons illustré cette situation : le lecteur peut se reporter, au début de ce chapitre, figure 1, où les cases parcourues sont matérialisées en jaune.



242 - Au sein du troisième cycle, l'agent passe sous le seuil et devient stressé

L'agent parcourt plusieurs fois le cycle et s'épuise progressivement.

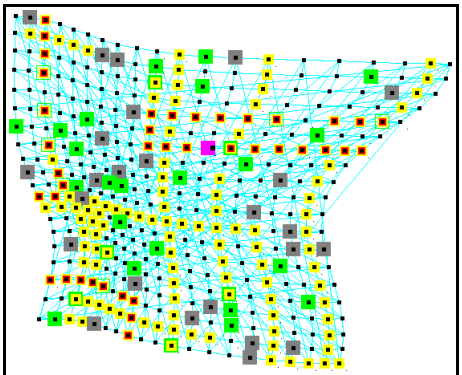
Dès qu'il commence à boucler, l'agent mange moins. Sur la figure ci-dessus, D'abord nous voyons que, pendant sa quête fructueuse de nourriture, son niveau d'énergie monte régulièrement. Ensuite, quand il tourne de le cercle vicieux, nous remarquons bien les trois cycles qui s'étagent en descendant.



243 L'agent marque le support avec une hormone

L'agent passe sous le seuil d'alerte.

Tout à la fin du chronogramme, vers les dates 865, l'agent passe dessous le seuil d'alerte et il commence à marquer les lieux avec la phéromone de stress. Dans la figure de droite, la flèche tournée vers la gauche indique l'agent qui se dirige à l'est, il vient de marquer deux cases en rouge (celles qui sont montrées par les flèches verticales).



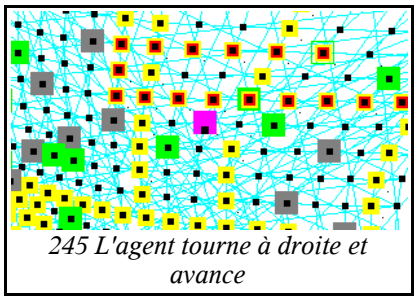
244 - L'agent se trouve face à son stress

Après un cycle l'agent retombe sur ses marques.

Après avoir parcouru un cycle, l'agent repasse là où il est déjà passé, et trouve les marques de stress qu'il a laissées (ci-contre à gauche).

L'agent se sort du cycle vicieux.

Comme il se trouve face à une case marquée par du stress, l'agent tourne à droite, et avance : ainsi il sort de ce cercle vicieux où il était tombé (ci-contre à droite).



245 L'agent tourne à droite et avance

3.C) Interprétation des résultats

L'agent se sort d'un cycle vicieux.

Dans cet exemple l'agent se sort d'un cycle vicieux, il repart droit devant lui, en quête de nourriture.

Difficultés à quantifier l'avantage sélectif introduit.

Ici encore, il est difficile de quantifier l'avantage sélectif introduit. Cette aptitude lui permet d'éviter de mourir de faim, et constitue pour lui un avantage sélectif, à condition que la production de phéromone ne puise pas trop dans ses réserves d'énergie.

4) Conclusions

2DInsecSoc constitue une étape de transition qui balise le chemin. Elle vient juste après une communication involontaire, et introduit progressivement un échange volontaire local entre agents. Elle présente une communication à la mode, le marquage par phéromones, qui est effectivement très simple, car naturel.

Mais il s'agit d'un média limité car il est local : il atteint seulement les agents qui *passent dans le coin*. Il est donc adapté à cette situation et à beaucoup d'autres, où il faut marquer des phénomènes statiques et locaux.

Cependant, d'autre situations où il faut marquer des phénomènes dynamiques ou globaux (alerte météo, prédateurs...) requièrent une communication qui, tout en restant volontaire, s'effectue alors globalement. C'est ce que nous traiterons, dans le chapitre suivant.

Ch. 38 *2DGroupCoop* : le groupe et la communication globale active

1) Introduction

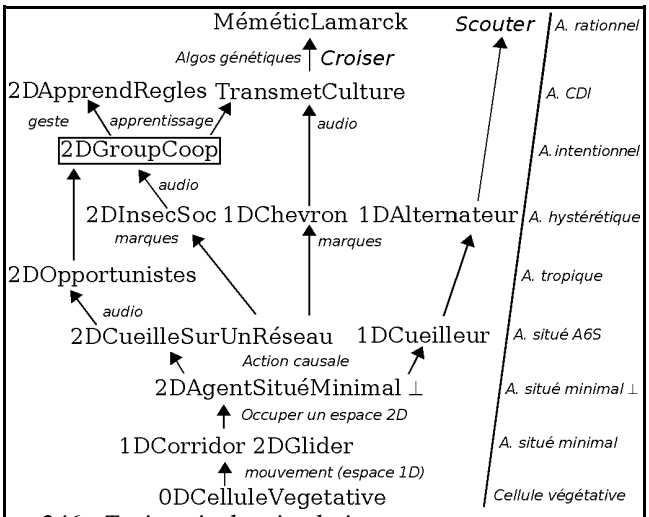
Continuer sur le groupe et la communication.

Dans cette étape *2DGroupCoop*[80] nous continuons à travailler avec le groupe : nous prolongeons la communication active. La portée des échanges, qui était locale, devient globale.

Communication volontaire et globale.

Dans les étapes précédentes, la communication induisait un effet limité, car elle était soit passive, soit locale.

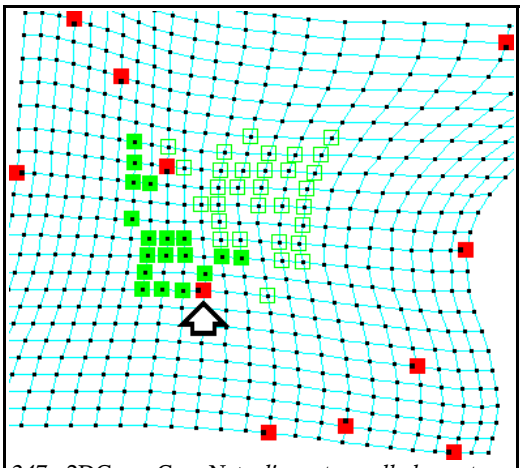
Maintenant, dans notre modèle, quand l'agent émetteur prévient tous les autres, nous passons à une communication volontaire et globale : elle devient plus efficace.



246 - Taxinomie des simulations que nous proposons

Présentons graphique de cette simulation.

Dans la simulation ci-contre, à gauche, l'agent (flèche vers le haut) qui cherchait de la nourriture vient d'en trouver (case verte). Il vient d'appeler les autres : maintenant qu'ils sont alertés, ils cessent de chercher dans leur coin, et progressent vers lui. L'appel de leur collègue a changé leur état interne : nous déduisons que ce sont des agents intentionnels.


247 - 2DGroupCoopNat *: l'agent appelle les autres*

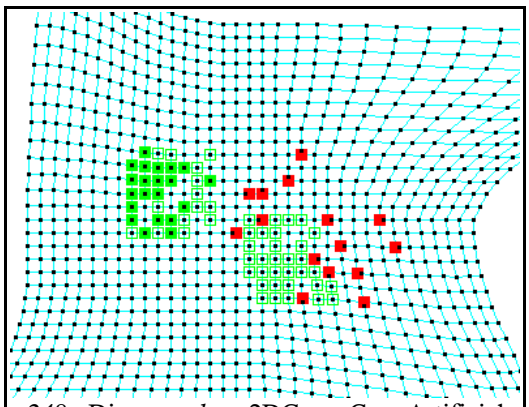
Présentation de cette simulation.

Dans *2DGroupCoop*, nous modélisons la nature, alors ces agents mieux organisés sont plus performants, et, en situation de compétition, leur population serait renforcée par le mécanisme de sélection des espèces : à travers cette simulation, nous débouchons sur une modélisation de l'organisation sociale.

80 Cette étape est nommée *2DGroupCoop*, pour *groupe coopératif dans un monde 2D*.

2) Matériels, méthodes et implémentation

Quels outils informatiques utilisons-nous pour modéliser cette étape ?



248 - Diaspora *dans* 2DGroupCoopArtificiel

2.A) Modélisation du support :

Pas d'obstacle et une nourriture groupée.

Nous reprenons un support très simple : le réseau 2D habituel, où chaque lieu possède quatre voisins. Cette fois nous n'avons pas besoin d'y introduire des obstacles. Par contre, la nourriture, qui précédemment était distribuée aléatoirement sur le support, est maintenant groupée par tas.

Nous utilisons deux façons de disposer le mobilier sur le support.

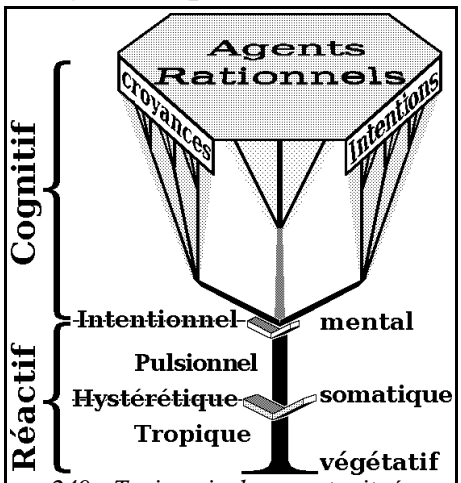
D'abord nous présentons *2DGroupCoopNat*, avec le suffixe *Nat* qui signifie *naturelle*, où nous trouvons simplement un seul lot de nourriture. Cette simulation, construite de façon très naturelle, est d'une grande simplicité, mais fournit un résultat diffus, pas très net.

Ensuite nous présentons *2DGroupCoopArt*, avec le suffixe *Art* qui signifie *artificielle*, où nous répartissons la nourriture en deux lots séparés. Cette simulation, construite de façon moins naturelle, permet d'obtenir un résultat plus tranché, plus net.

2.B) Modéliser l'être comme un agent intentionnel hystérétique[81]

Le traitement de l'information effectué ici, se complexifie un peu. Chaque être devient un agent intentionnel car il change d'état interne dès qu'il est appelé par son collègue qui a trouvé de la nourriture. Maintenant nous sommes en présence d'agents hystérétiques, i.e. doués de mémoire.

L'agent est muni d'un circuit *monostable*, i.e. un état interne très simple qu'il force à 1 quand il entend que quelqu'un a trouvé de la nourriture, et qui retombe tout seul au bout de 4 cycles si personne n'est venu confirmer la découverte



249 - Taxinomie des agents situés

Forçage à 1 du monostable

```
if((pNxtCase->Nature==food)&&(pNxtCase->ReBirthDate<=date)){ //Si agent trouve de la nourriture mangeable
...
 LastFoodFoundDate= date;          //  actualisation de la date de la dernière absorption de nourriture
 EtatInterneDuGroupe= VersFood;    //  alors il claironne qu'il en a trouvé(hypothèse de nourriture groupée
 pAgentNatriarche= pAgent;         //  L'agent qui vient de trouver de la nourriture, le signale
 return;
}
```

Table 41 : L'agent qui trouve de la nourriture le signale et devient natriarche[82] (tous se dirigent vers lui)

Forçage à 0 du monostable

81 Hystérétique i.e. un agent doué de mémoire, par opposition à *tropique*.

82 Natriarche, pour n-atriarche, avec la lettre *n* de *neutre*, située dans l'alphabet entre le *m* de *matriarche* et le *p* de *patriarche* : un néologisme pour désigner, dans ces simulations sans mâle ni femelle, le leader asexué du groupe.

```
if (LastFoodFoundDate+4 <date)// Si pendant au moins 4 cycles aucun agent n'a trouvé de nourriture alors
EtatInterneDuGroupe=SeekFood; //il le dit,passe le groupe en diaspora recherche de nourriture:SeekFood
```

Tab 42 : Le passage en diaspora*, i.e. en recherche de nourriture, se fait si pendant au moins 4 cycles, personne ne trouve de nourriture*

Comment est codée la recherche de nourriture ?

La *diaspora* est très efficace pour chercher de la nourriture car, à l'instar des agents *haine* de Braitenberg, les agents s'éloignent du centre de gravité du groupe en se tenant éloignés les uns des autres au moyen de signaux audio. Pour simplifier l'implémentation, la chose n'est pas directement codée, mais seul le comportement résultant est simulé.

Pour effectuer cette simulation, chaque agent calcule la répulsion en $1/x^2$ qu'il subit de ses partenaires, se tourne selon le sens de cette force, et avance.

```
for (i=0; i<AgentNbr; i++) {                              // Va chercher SwarmTabAgent de l'être car chacun
 int InxSwarmTab_pAgentCourant= SwarmTabAgent[i]->InxDeAgentDansSwarmTabAgent;//connaît son index dns Tab
 if                                                       // Parcourt tout le tableau
  ((InxSwarmTab_pAgentCourant != InxSwarmTab_pAgentFuyant)// Donc on exclut lui-même
   &&
   (SwarmTabAgent[i]->EnergyLevel>LethalLevel))           // Donc on exclu ceux qui sont morts
 {
  Case * pCaseCourant= SwarmTabAgent[i]->pCaseOuIAm;      // Rempli un tampon
  int InxJ_pAgentCourant= pCaseCourant->InxJ;
  int InxI_pAgentCourant= pCaseCourant->InxI;
  int DeltaI= InxI_pAgentCourant - InxI_pAgentFuyant;     // On se centre sur l'agent fuyant, Donc on doit
  int DeltaJ= InxJ_pAgentCourant - InxJ_pAgentFuyant;     // soustraire ses coordonnées à l'agent courant
  if (DeltaI>0) cGoToI+=(1000/(DeltaI*DeltaI));           // La contribution de chaque agent est :
  if (DeltaI<0) cGoToI-=(1000/(DeltaI*DeltaI));           // - projeté sur les axes x et y
  if (DeltaJ>0) cGoToJ+=(1000/(DeltaJ*DeltaJ));           // - inversemnt proportionné au carré de distance
  if (DeltaJ<0) cGoToJ-=(1000/(DeltaJ*DeltaJ));           // - sommée avec les autres
 }                                                        // fprintf(stderr,"cGoToJI%d%d\n", cGoToJ,cGoToJ)
}
```

Table 43 : Codage de la diaspora*, chaque agent tourne dos au groupe et avance*

Au total le code d'un agent réactif se complexifie.

Pour effectuer cette simulation, chaque agent tourne le dos au natriarche et avance.

```
void Agent_AgisA7Date(Agent * pAgent, int date) {          // enum Nature {sol, arbre, mur, agent};
 Case * pNxtCase;                                          // pointeur tampon

 if (pAgent->EnergyLevel < LethalLevel) {                  // Gestion de la mort de l'agent
  Case * pCaseOuIAm= pAgent->pCaseOuIAm;                   //  pointeur tampon
  pCaseOuIAm->pAgent=  NULL;                               //  La case ne contient plus d'agent
  fprintf(stderr, "mort ");                                //  Traitement de la mort des agents par épuisement
  return;                                                  //  Et on choisit de s'arrêter ici
 }
                                                           // Gestion de l'énergie
 pAgent->EnergyLevel-= ActionEnergyCost;                   // Chaque action de l'agent, lui fait consommer de l'énergie
                                                           // Gestion des chocs entre agents
 pAgent->ToDrawBumped= 0;                                  // Au début de chaque cycle, l'agent ne s'est pas tapé.

 if (EtatInterneDuGroupe)                                  // Affichage de l'état interne du groupe
  fprintf (stderr, "diaspo %d ", pAgent->InxDeAgentDansSwarmTabAgent);// enum EtatInterne {SeekFood, VersFood};
  else
  fprintf (stderr, "ToFood %d ", pAgent->InxDeAgentDansSwarmTabAgent);
                                                           // Remplissage du tampon pNxtCase
 pNxtCase= pCase_QuelleEstLaCaseDevantToi_pAgent(pAgent); //Regarde ce qu'il y a devant et instancie la pCase correspondante

                                                           // Gestion des chocs entre agents
 if (pNxtCase->pAgent) {                                   // Qq soit l'EtatInterneDuGroupe, traitement des chocs entre agent
  fprintf(stderr, "chocs entre agent\n");                  // Dans ce cycle de traitemt de tous ls agents, 7 agent s'est tapé
  pAgent->ToDrawBumped= 1;                                 // Il faut donc l'afficher ainsi au moment d'afficher ts ls agents
  TourneDroite_pAgent(pAgent);
  return;                                                  // Et on choisit de s'arrêter ici
 }
                                                           // Traitement de l'action proprement dite

 PrintTesSlotsSurStderr_pAgent(pAgent);                    // Affichage des slots de l'agent pour comprendre

 if ((pNxtCase->Nature== food)&&(pNxtCase->ReBirthDate <= date)) {//Si l'agent trouve de la nourriture et si c'est mangeable

  if (pAgent->EnergyLevel < SeuilDeSatiete)                // s'il a faim
   MangeEtAvance_pAgentA7Date(pAgent, date);               //   Alors il la mange
   else                                                    //   sinon
   Avance_pAgent (pAgent);                                 //   il faut la laisser pour les autres et dégage le terrain

  LastFoodFoundDate= date;                                 //  actualisation de la date de la dernière présence de nourriture
  EtatInterneDuGroupe= VersFood;                           //  alors, il claironne qu'il en a trouvé (car nourriture groupée)
  pAgentNatriarche= pAgent;                                //  L'agent qui vient de trouver de la nourriture, le signale
  return;
 }
                                        // Si ici, alors traitement d'un agent n'a pas (ou pas eu) de nourriture sous le nez
 switch(EtatInterneDuGroupe) {
  case SeekFood :                                          // Si le groupe est en recherche de nourriture
   Vu7AgentFuyant_PollTousLesAgentsEtEloigneTEn(pAgent);   // alors c'est diaspora, chaque agent se tient à distance des !=
   break;
  case VersFood :                                          // Si au moins un agent vient de trouver de la nourriture
    AgentSuiveurTourneToiVers7AgentNatriarche(pAgent, pAgentNatriarche); // Alors chaque agent se tourne vers lui
```

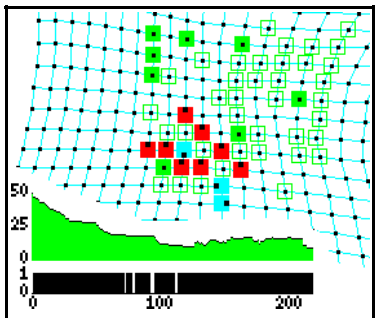
```
    Avance_pAgent (pAgent);                          //  et va vers lui
    if (LastFoodFoundDate+4 < date)                  // Si pendant au moins un cycle personne n'a trouvé de nourriture
     //(pAgent->EnergyLevel < AlerteLevel)           //  l'agent a faim
     EtatInterneDuGroupe= SeekFood;                  //  alors il le dit et passe le groupe en diaspora : SeekFood
   break;
 }
}
```

Table 44 : Tout le code comportemental de l'agent intentionnel

3) Présenter les résultats

3.A) Lancement, présentation, utilisation du programme

Dans *2DGroupCoop*, ces rubriques demeurent inchangées. Pour obtenir plus de détails se reporter à *1DCueilleur*.



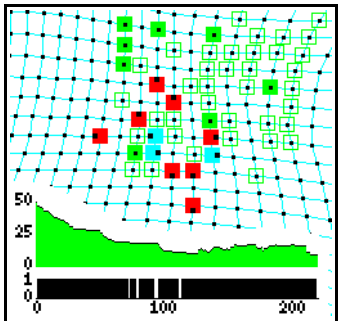
250 - Avant le mode diaspora*, les agents sont groupés*

3.B) Résultats pour *2DGroupCoopNat*

Pour illustrer ces résultats nous présenterons tout d'abord *2DGroupCoopNat*, moins spectaculaire que la simulation artificielle, mais charmante de simplicité.

Avant la *diaspora.*

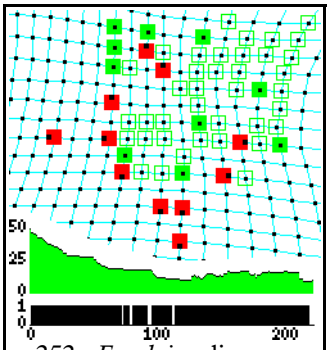
Dans ce cas, les agents sont groupés autour des derniers restes de nourriture qu'il viennent de trouver (ci-dessus à droite).



251 - Le début de la diaspora

Début de la *diaspora.*

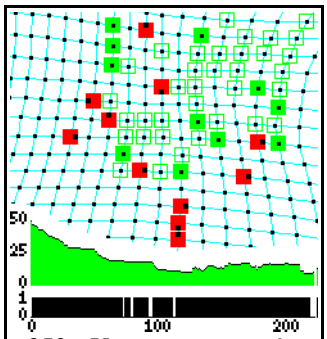
Les agents, qui ne voient que devant eux, n'ont pas trouvé la case juste à côté d'eux. Comme ils jeûnent depuis quatre cycles, le monostable vient de retomber : c'est le début de la *diaspora*. En observant leur direction, nous pouvons voir qu'ils se fuient les uns les autres (illustration ci-contre à gauche).



252 - En pleine diaspora

En pleine *diaspora.*

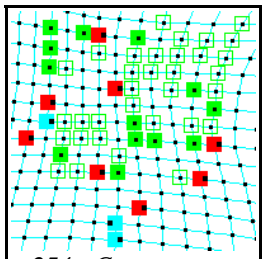
Le groupe est en pleine *diaspora*. Nous voyons bien l'éparpillement des agents et leur front de progression. L'agent tout en haut va trouver de la nourriture (ci-contre à gauche) .



253 - Un agent a trouvé...

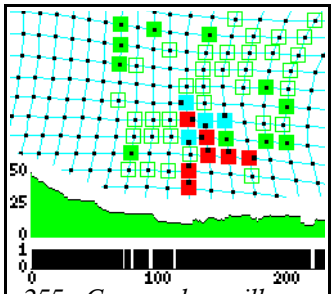
Quelqu'un a trouvé de la nourriture.

Le monostable est remonté, la *diaspora* est finie, quelqu'un a trouvé de la nourriture : les agents convergent vers le centre du groupe (ci-contre à droite).


254 - Convergence

Convergence.
Progressivement, la convergence continue. (ci-contre à gauche).

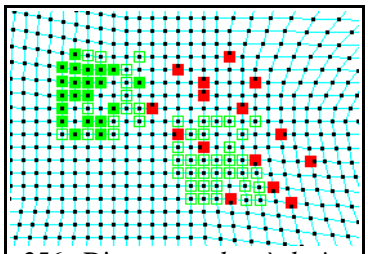
Un groupe d'agent cueilleurs.
Maintenant la convergence est terminée. Le groupe s'est reformé autour d'un petit lot de nourriture. Nous obtenons un groupe d'agents cueilleurs (ci-contre à droite).

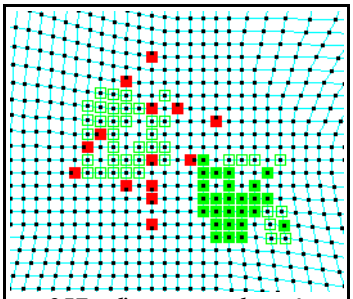


255 - Groupe de cueilleurs

3.C) Résultats pour *2DGroupCoopArt*

Ensuite, pour illustrer ces résultats nous présenterons *2DGroupCoopArt*, la simulation artificielle bien spectaculaire, mais elle présente l'inconvénient d'avoir été téléguidée : la nourriture est distribuée en deux lots bien distincts.

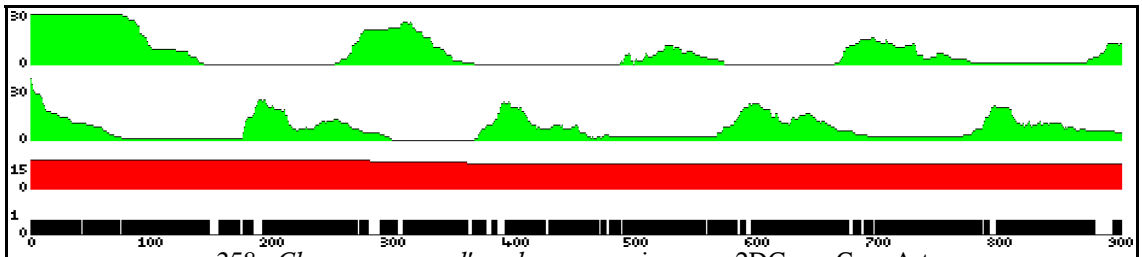

256 - Diaspora *en bas à droite*


*257 - d*iaspora *en haut à gauche*

Présentons de deux configurations opposées.
2DGroupCoopArt est une simulation qui oscille bien, i.e. qui passe par des cycles clairement opposés. Pour illustrer cet aspect commençons par présenter les clichés de deux organisations nettement contrastées :

Tout d'abord, (au-dessus - à droite) voici le réseau, en pleine *diaspora*, quand les agents ont mangé la nourriture qui était en bas à gauche : ils recherchent à nouveau de l'énergie.

Ensuite, (ci-contre à gauche) voici le réseau, au milieu de la *diaspora* suivante, environ 100 cycles plus tard, quand les agents ont mangé la nourriture qui était en haut à droite.



258 - Chronogramme d'une longue session avec 2DGroupCoopArt

Les quatre chronogrammes des deux cycles décalés.
De haut en bas nous observons :
- Courbe 1 : le nombre de pièces de nourriture dans le tas en haut à gauche.
- Courbe 2 : le nombre de pièces de nourriture dans le tas en bas à droite.
- Courbe 3 : le nombre d'agents vivants sur le support. Notez qu'il descend progressivement.
- Courbe 4 : l'état du monostable indicateur de *diaspora*.

Dans le chronogramme ci-dessus, nous distinguons très clairement les deux oscillations, de période 200 unités de temps et qui sont déphasées d'une demi période.

3.D) Interprétation des résultats

Rappel : les agents sont borgnes et myopes.

Avant tout, nous devons garder à l'esprit le fait que chaque agent est borgne et myope : il possède un unique œil et voit seulement la case devant lui. Dans ces conditions, pour lui, la recherche de nourriture, surtout quand elle est groupée par lots, devient difficile.

Quantifier l'avantage sélectif introduit.

Le passage d'agent réactif à agent intentionnel se fait au prix d'un complexification de la circuiterie interne des êtres. En tous cas, cette modification est effectuée bien à propos, car elle permet à ces agents borgnes et myopes de trouver de la nourriture dans ces conditions difficiles et introduit un avantage sélectif évident.

4) Conclusions

2DGroupCoop introduit un premier résultat. Nous avons munis les agents d'une faculté de communication. D'abord, ils ont adapté leur structure pour chercher efficacement de la nourriture.

De plus, grâce à cette aptitude nouvelle, ils se sont organisés en groupe : d'abord ils se dispersent pour chercher de la nourriture, ensuite ils se rassemblent quand ils en ont trouvé.

En conclusion disons qu'un épiphénomène émerge, nous remarquons qu'une organisation sociale apparaît. Le groupe se structure : il s'est organisé en essaim[83].

83 *Essaim* : *swarm* en anglais.

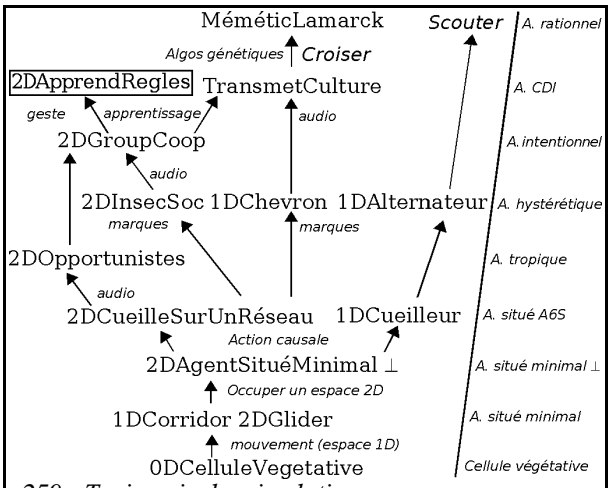
Ch. 39 Apprentissage *: 2DApprendRègles*

(un parent enseigne des règles à un enfant)

1) Introduction de l'apprentissage

Situons cette étape au sein de notre taxinomie :

Nommée *2DApprendRègles*, elle constitue une des premières applications pratique de la modélisation décrite ci-avant. Nous utilisons tous nos résultats précédents afin de construire un exemple d'apprentissage, qui s'avère facile à programmer : l'extrême simplicité de l' agent réactif, associé au contact audio de Braitenberg, permet à un natriarche[84] d'enseigner par l'exemple, trois règles de conduite à un enfant.



259 - Taxinomie des simulations que nous proposons

2) Problématique et démarche : traiter un exemple simple d'apprentissage

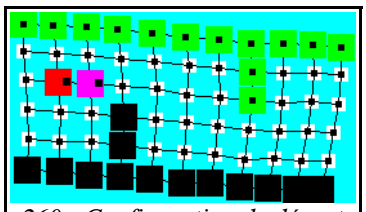
2.A) Présenter des constituants de cette simulation

Le support est un réseau de lieux qui se reboucle indéfiniment.

Les rangées d'arbres verts et les murs de pierre sombre forment un couloir orienté ouest-est qui est encombré d' obstacles.

Les deux agents sont borgnes et myopes : chacun perçoit seulement la nature de la case devant lui (sol, mur ou arbre).

Le natriarche (agent magenta) possède un émetteur audio : il appelle l'enfant qui l'entend au moyen de son récepteur.


260 - Configuration de départ

2.B) Le parent chemine à l'est en appelant l'enfant qui apprend en le suivant

L'enseignant chemine vers l'est.

Dans le couloir, le natriarche (agent magenta) chemine vers la droite, entre les obstacles.

La topologie des lieux est organisée en boucle.

Le couloir se reboucle indéfiniment. Arrivée à l'est, chaque agent se retrouve à l'ouest.

L'enseignant chemine en appelant l'enfant qui le suit.

Il maintient un contact audio avec l'enfant (agent rouge) qui le suit. Ce faisant, l'enseignant montre l'exemple à son étudiant qui apprend en le suivant.

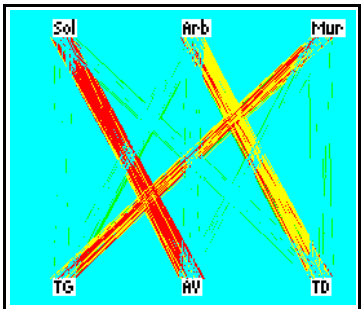
84 Natriarche, pour n-atriarche, avec la lettre *n* de *neutre*, située dans l'alphabet entre le *m* de *matriarche* et le *p* de *patriarche* : un néologisme pour désigner, dans ces simulations sans mâle ni femelle, le leader asexué du groupe.

L'enfant apprend en le suivant.

Ouvert à l'extérieur, l'enfant est doté aussi d'un capteur visuel, mais il perçoit seulement la case devant lui.

Doté d'un méta-regard, l'enfant perçoit aussi une partie de son traitement de l'information. Il accède à la commande des mouvements qu'il effectue pour suivre son enseignant.

De cette façon, ces stimulations externes combinées à la connaissance des actions internes correspondantes, lui suffisent à effectuer un apprentissage neuronal, i.e. à connecter au moyen de simples liaisons (figure ci-contre à droite), ses trois capteurs (sol arbre mur) à ses trois actionneurs (avance, tourne_à_droite, tourne_à_gauche).



261 - Apprentissage de connexions neuronales

3) Outils et méthodes pour implémenter les modélisations

Quels outils informatiques utilisons-nous pour modéliser cette étape ?

3.A) Modélisation du support :

a) Des simplifications drastiques du monde pour modéliser un apprentissage

Cette simulation de notre taxinomie modélise un agent qui apprend, il s'agit tout de même de modéliser un transfert de culture entre génération, ce qui constitue un challenge appréciable. Aussi nous avons procédé à des simplifications drastiques dans la conception de cette expérience.

b) Nous utilisons un support cylindrique

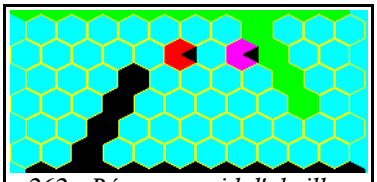
Afin que l'enfant puisse effectuer un apprentissage, nous devons lui fournir une série de stimulations récurrentes. Afin d'éviter l'utilisation d'un support infini, nous choisissons d'en prendre un qui se reboucle sur lui-même : l'agent qui sort à l'est rentre à l'ouest, à la hauteur à laquelle il est sorti, i.e. dans la même ligne.

c) Sur le support, nous plaçons des obstacles, mais pas de nourriture

Nous reprenons le support 2D habituel, et le simplifions : nous n'introduisons pas de nourriture dans le paysage. Par contre, les obstacles, qui précédemment étaient distribués aléatoirement sur le support, sont maintenant ordonnés : une rangée d'arbres au nord, et un mur de pierres au sud.

d) Nous utilisons deux types de support : à 4 et 6 voisins

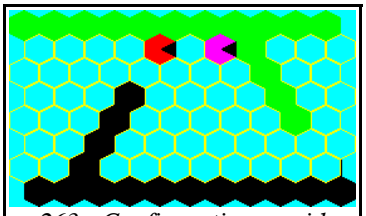
Nous avons écrit deux versions de support. En plus de celle que nous présentons au début du chapitre, où un lieu possède 4 voisins, nous avons ajouté une version où les cases en possèdent 6 (ci-contre à droite). Ainsi nous obtenons un support structuré en nid d'abeilles.


262 - Réseau en nid d'abeilles

Dans ce cas, il est difficile de créer des angles à 90°, alors les murs et arbres sont rangés différemment. Mais pour le reste, la totalité du programme fonctionne selon le même synoptique.

e) Nous avons testé trois architectures de support

Afin de vérifier les résultats obtenus par ce système mixte *enseignement-apprentissage*, nous avons testé trois architectures différentes de support.

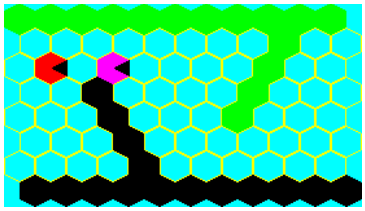

263 - Configuration en nid d'abeilles : parcours facile

D'abord l'architecture classique à 4 voisins.

D'abord nous avons utilisée la configuration classique, où chaque lieu possède 4 voisins, que nous présentons tout au long de ce chapitre.

Puis nous avons testé deux architectures en nid d'abeille.

Dans cette structure hexagonale (à 6 voisins), l'angle à 60° prédomine, alors, il est difficile de créer des angles à 90°, et les murs, arbres sont rangés différemment, ce qui induit deux positionnement des obstacles.



264 - Configuration en nid d'abeilles : parcours difficile

Voici la première (ci-dessus à droite), que nous avons déjà présentée au tout début de ce chapitre. Nous la qualifions de facile, car l'orientation des rangées d'obstacles est aisée pour les agents, qui apprennent facilement à se déplacer.

Nous qualifions de difficile la seconde architecture testée (ci-contre à gauche), car l'orientation des rangées d'obstacles est contraignante pour les agents, qui apprennent plus péniblement à se déplacer.

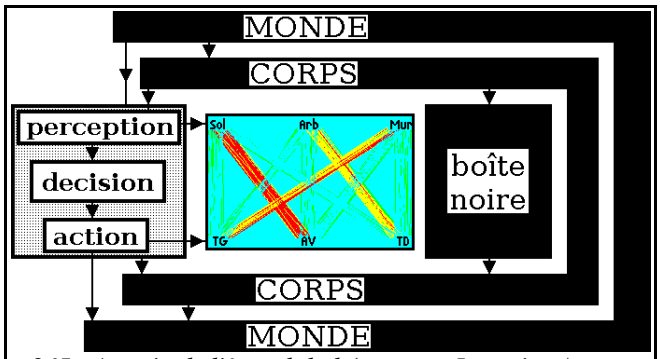
3.B) Modélisation de l'agent apprenant

Dans cette simulation, quand nous regardons cet agent qui apprend un comportement de son parent, nous distinguons chez lui deux êtres, à deux étapes différentes : D'abord, globalement, l'agent qui effectue un apprentissage, puis ensuite, quand la leçon est apprise, l'agent réactif qui, à ce moment là, sait se déplacer sur le support.

L'être qui apprend est un *agent Intensions.*

Nous modélisons l'être qui apprend (ci-contre à droite) comme un *agent Croyances* car il est hystérétique[85].

En effet, dans cette nouvelle étape de la progression proposée, le traitement de l'information effectué par l'agent, se complexifie encore. À ce stade, l'être qui apprend (ci-contre, à droite) devient un agent *Croyances* car il change son état interne en modifiant ses connexions neuronales. Mais, ici l'information apprise s'enregistre dans une mémoire morte : par essence, cet apprentissage est destiné à durer autant que l'être qui le porte.



265 - Au sein de l'être global (un agent Intensions) nous distinguons la circuiterie colorée d'un agent réactif.

Modélisation du fonctionnement de l'être appris comme un agent réactif.

La structure de l'être construit au sein de l'apprenant (partie colorée dans le dessin ci-contre) se réduit à celle d'un agent réactif.

En effet, un travail d'apprentissage est difficile, alors, pour débuter le travail, nous choisissons de commencer à apprendre une configuration, choisie parmi les plus simples.

Grâce à notre précédent travail de dégrossissage, nous savons déjà que des agents réactifs très simples, reliant seulement des capteurs aux actionneurs, présentent cependant un comportement viable.

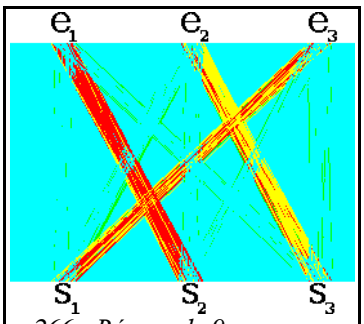
À cet étape, l'enfant dispose déjà de règles qui lui servent à suivre son parent par écholocation. Maintenant, il apprend de nouvelles règles de pilotage réactif, qui viennent s'ajouter à ces premières. En conclusion, à la fin de cet apprentissage, quand son parent qui le guide, disparaîtra, ce jeune être sera devenu un agent réactif autonome : il sera capable de se déplacer seul.

85 Hystérétique : un agent doué de mémoire, par opposition à l'agent tropique.

3.C) L'agent apprend à piloter son déplacement au milieu des obstacles

a) Introduction : simuler un apprentissage neuronal

À cette étape, nous modélisons un apprentissage neuronal : nous devons enregistrer, un ensemble de règles, au sein d'une mémoire morte fabriquée au moyen de neurones. Donc, notre simulation doit présenter le synoptique du fonctionnement d'une mémoire classique ; à ceci près que le domaine change : nous ne sommes plus en binaire, mais en analogique.



266 - Réseau de 9 neurones

b) Synoptique d'apprentissage de règles analogiques

Une fonction de transfert analogique 3x3.

Le montage doit apprendre un jeu de règles, i.e. une fonction de transfert entre 3 capteurs d'entrée et 3 actionneurs de sortie. De plus, nous quittons le domaine de la logique, pour passer en analogique, donc le système doit apprendre un poids. Ceci nous amène à effectuer un apprentissage dans un tableau de 3x3=9 entiers. Ainsi nous obtenons cette implémentation informatique :

```
#define CapteurNbr 3                          // Nous utilisons 3 capteurs en entrée
#define ActionneurNbr 3                       // Nous utilisons 3 actionneurs en entrée
int DecisionTab [CapteurNbr] [ActionneurNbr]; // Une table de 3x3 = 9 entiers suffit
```

Table 45.: La table de décision de l'agent se réduit à un tableau de 3x3 int

Rappel sur le parcours en largeur d'abord.

Muni de trois fonctions unaires *f*, *g* et *h* ; disposant de trois variables *x*, *y* et *z*, en composant ces éléments selon un parcours en largeur d'abord, à l'instar de la façon dont Herbrand construit ses univers, nous obtenons une liste qui commence ainsi : *{x, y, z,*

f(x), f(y), f(z),
g(x), g(y), g(z),
h(x), h(y), h(z),
f(f(x)), f(f(y))... }.

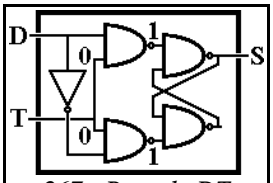
Une hypothèse simplificatrice du parcours en largeur d'abord.

Normalement, avec un tel montage, chacune des sorties s_j se calcule en sommant les contributions k_i des trois entrées e_i.:

Par exemple, la sortie S_1 se calcule ainsi : $S_1 = \sum_{i=1}^{3} k_i \times e_i$. Cependant, afin de compenser la complexité introduite par le traitement analogique, nous faisons l'hypothèse simplificatrice du parcours en largeur d'abord, qui s'avère exacte à l'expérimentation : les poids s_j sont uniquement fonction d'un seul e_i, i.e. : nous avons simplement : $S_n = k_n * e_n$. (pour n = 1, 2 ou 3).

c) Le synoptique d'apprentissage d'un contenu mémoire est décomposé en 3 parties

Même si nous ne sommes plus en binaire, mais en analogique, notre simulation doit cependant présenter le synoptique du fonctionnement d'une mémoire classique (bascule DT ou registre) et donc se décomposer en trois temps :

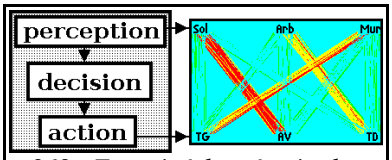


267 - Bascule DT

D'abord, nous devons fournir à la mémoire, l'information à apprendre. Ensuite nous devons fournir les synchronisations nécessaires au montage. Enfin, l'enregistrement de cette information s'effectue au sein de la circuiterie du réseau de neurones.

Fourniture à la mémoire morte des règles à apprendre.

Nous souhaitons effectuer un apprentissage par l'exemple. Dans l'attitude classique, l'enseignant effectue une action, l'apprenant l'observe et la reproduit. Mais ceci implique chez ce dernier une aptitude à l'analyse de scène, qui, même avec



268 - Fournir à la mémoire les règles à apprendre

les outils de l'I.A. s'avère très complexe, or il n'est pas question d'implémenter un tel fonctionnement dans cette simulation qui se veut très simple.

Pour palier ce problème, nous utilisons la méthode de Braitenberg, un montage *amour*, qui permet à l'apprenant de suivre son leader. Sur cette base, l'apprentissage est possible en fournissant au réseau de neurones les perceptions des capteurs, et la commande des actionneurs, i.e. l'action que l'enfant a effectué pour suivre son parent (ci-contre à droite).

Fourniture à la mémoire morte du top de chargement, i.e. de la synchronisation.

Enfin, dans cette simulation, nous devons fournir à la mémoire la pulsation de synchronisation. Puisque l'enseignement fourni à l'apprenant est produit par un agent réactif, alors l'apprentissage s'effectue à chaque cycle unitaire, i.e. à chaque intervalle de temps.

Enregistrement des règles au sein d'une mémoire morte.

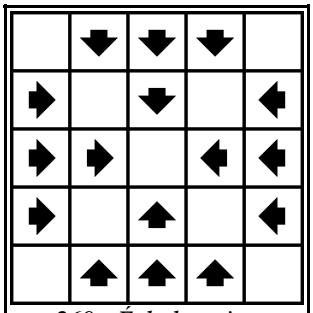
L'algorithme d'apprentissage des règles n'a pas besoin d'être aussi complexe que celui de la *rétro-propagation de gradient*, classique en réseaux de neurones. Au contraire, il est très simple et consiste en seulement quatre règles très naturelles :

- Initialisation :	Toutes les connexions $k(e_i)$ sont initialisées au coefficient médian 0,5 (50%).
- Heuristique du bonheur :	Toute activation fructueuse d'un neurone renforce sa connexion synaptique.
- Heuristique de la vieillesse :	À chaque instant toutes les connexions synaptiques s'affaiblissent un petit peu.
- Remise en forme du signal :	L'activation d'un actionneur se fait en comparant $k(e_i)$ avec 0,5 (le seuil médian).

3.D) L'algorithme de poursuite : comment l'apprenant suit-il son leader ?

Le suiveur est capable d'écholocation.

Notre agent réactif, enrichi du montage de Braitenberg, mis en présence de la source sonore de son leader, est seulement capable d'écholocation grossière. Cependant, cette aptitude s'avère suffisante et lui permet de se tourner vers son enseignant, et le suivre.



269 - Écholocation

Implémentation de l'algorithme de poursuite.

Selon un point de vue informatique, dans cette simulation, nous n'avons pas implémenté finement l'agent de Braitenberg. Nous n'avons pas simulé un signal, émis par l'appeleur, qui se propage au sein du réseau, pour être ensuite reçu et traité par le receveur.

Non ! Nous avons seulement simulé tout cet échange et codé le résultat, i.e. le fait que chaque agent se tourne vers l'appelant : en fonction de sa position relative à l'émetteur, d'abord l'agent suiveur se tourne vers son meneur et ensuite, avance vers lui.

Selon un point de vue plus pratique, voici comment nous implémentons cette fonction d'orientation du receveur. Nous plaçons l'émetteur au centre de ce tableau, et l'enfant receveur sur la case correspondant à sa position relative à son parent. Ainsi, pour chaque case nous obtenons le mouvement à effectuer : conserver la même direction, ou bien fixer le cap vers un des points cardinaux.

4) Résultats des trois apprentissages obtenus sur ces trois configurations

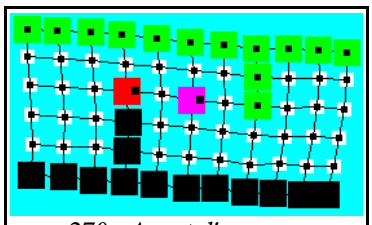
Nous travaillons dans trois configurations, et pour chacune d'elles, le système doit apprendre les neuf règles, qui connectent les trois *capteurs/senseurs* aux trois *actionneurs/effecteurs*.

4.A) Résultats pour l'*enseignement-apprentissage* sur un support à 4 voisins

Dans ce paragraphe nous présentons les résultats de l'*enseignement-apprentissage* obtenus sur un support à 4 voisins, pour chacune des 3 seules configurations qui apparaissent en situation

L'algorithme de guidage de l'enseignant.

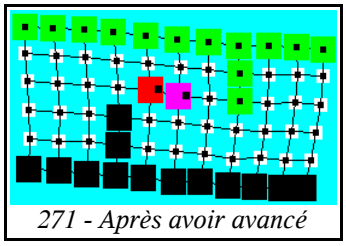
Le parent est autonome. À chaque intervalle élémentaire de temps, il exécute son programme d'agent réactif situé. Ainsi, il avance sur le support, et guide son enfant entre les obstacles.



270 - Avant d'avancer

L'algorithme de poursuite du suiveur.

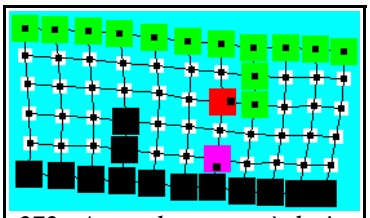
Pour chaque intervalle de temps élémentaire, il consiste en deux actions consécutives : d'abord, grâce à son mécanisme d'écholocation, le suiveur se tourne vers le meneur, puis il avance d'un pas.



271 - Après avoir avancé

Avancer, quand il n'y a rien devant.

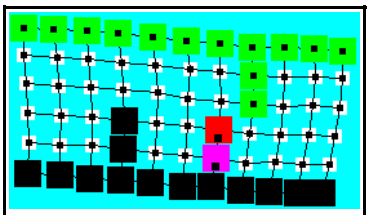
Dans l'image de gauche, le meneur en magenta vient d'avancer, il laisse un espace entre lui et son suiveur. Ce dernier entend son leader devant lui, donc il ne change pas de direction. Puis il avance. Alors nous obtenons la configuration de droite.



272 - Avant de tourner à droite

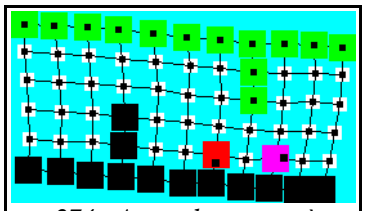
Tourner à droite.

Dans l'image de gauche, le meneur en magenta, est tout en bas, tourné vers le sud. Il laisse un espace entre lui et son suiveur.



273 - Après avoir tourné à droite et avancé

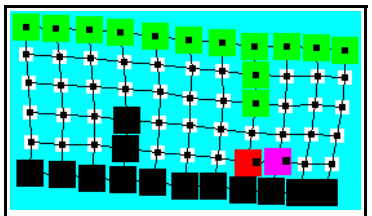
Ce dernier entend son leader sur sa droite, donc il tourne à droite et avance. Nous obtenons alors la configuration de droite.



274 - Avant de tourner à gauche

Tourner à gauche.

Dans l'image de gauche, le meneur en magenta, est devant, tout en bas, tourné vers l'est. Il laisse un espace entre lui et son suiveur.



275 - Après avoir tourné à gauche et avancé

Ce dernier entend son leader sur sa gauche, donc il tourne à gauche et avance. Nous obtenons la configuration de droite.

Analyse de ces résultats, obtenus sur un support à 4 voisins.

Dans ce paragraphe nous analysons longuement les résultats de l'*enseignement-apprentissage* obtenus sur un support à 4 voisins.

Comment se fait la convergence vers l'état d'équilibre ?

Un peu paradoxalement, en analysant toutes les étapes du trajet des deux agents, nous constatons qu'à aucun moment, l'algorithme de poursuite n'est mis en défaut : à chaque cycle, il fonctionne parfaitement. Grâce à la qualité de cet asservissement, l'enfant suit sans erreur son parent. Ainsi l'apprenant reçoit une information optimale : jamais le signal n'est bruité.

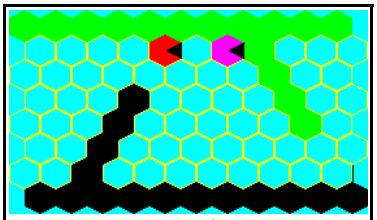
Dans de telles conditions l'apprentissage est facile, et il se déroule parfaitement. La convergence du système vers l'état d'équilibre se passe sans à coup. Ceci explique pourquoi nous pouvons utiliser un algorithme d'apprentissage si sommaire.

4.B) *Enseignement-apprentissage* sur un support à 6 voisins (config. facile)

Dans ce paragraphe nous présentons les résultats de l'*enseignement-apprentissage* obtenus sur un support en nid d'abeille, i.e. à 6 voisins. Dans la configuration présentée ici, l'orientation des rangées d'obstacles entrave peu le déplacement des agents, ainsi, ces derniers apprennent facilement à se déplacer. En conséquence, nous les présentons rapidement, car, bien que la structure du support ait changé, les résultats demeurent exactement semblables.

Algorithme de guidage de l'enseignant sur le support.

Cet algorithme de guidage de l'*enseignant/parent* demeure inchangé.



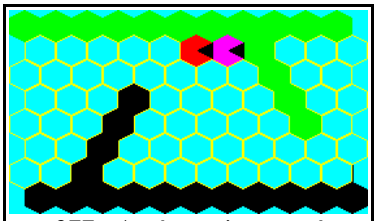
276 - Avant d'avancer

Algorithme de poursuite du suiveur par l'apprenant.

Il est inchangé. À chaque instant élémentaire, le suiveur localise le meneur, se tourne vers lui, puis avance d'un pas.

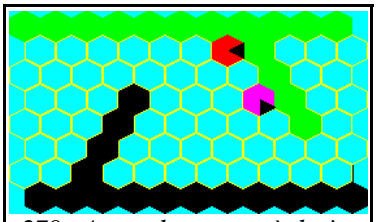
Avancer, quand il n'y a rien devant (image de gauche).

Le meneur en magenta vient d'avancer, il laisse un espace entre lui et son suiveur.



277 - Après avoir avancé

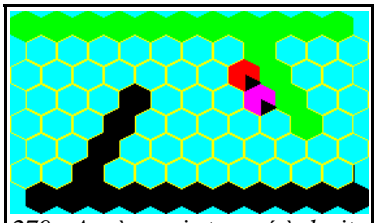
Ce dernier entend son leader devant lui, donc il ne change pas de direction. Puis il avance. Nous obtenons alors la configuration de droite.



278 - Avant de tourner à droite

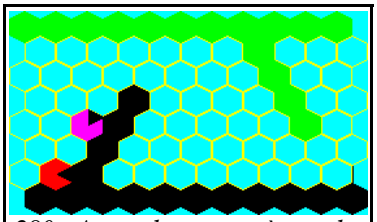
Tourner à droite.

Dans l'image de gauche, le meneur en magenta, est au milieu, tourné vers le sud-est. Il laisse un espace entre lui et son suiveur.



279 - Après avoir tourné à droite et avancé

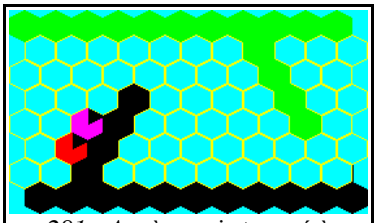
Ce dernier entend son leader sur sa droite, donc il tourne à droite et avance. Nous obtenons alors l'image de droite.



280 - Avant de tourner à gauche

Tourner à gauche.

Dans l'image de gauche, le meneur en magenta, est devant, au milieu, tourné vers le nord-est. Il laisse un espace entre lui et son suiveur.



281 - Après avoir tourné à gauche et avancé

Ce dernier entend son leader sur sa gauche, donc il tourne à gauche et avance. Nous obtenons la configuration de droite.

Analyse des résultats : la convergence vers l'état d'équilibre se fait facilement.
Sur ce support en nid d'abeilles, à chaque cycle, l'algorithme de poursuite fonctionne parfaitement. Grâce à la qualité de cet asservissement, l'apprenant reçoit une information optimale. Ainsi l'apprentissage se déroule parfaitement. La convergence du système vers l'état d'équilibre se passe sans à coup.

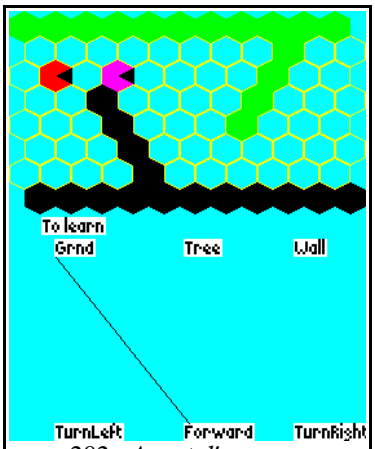
4.C) *Enseignement-apprentissage* sur un support à 6 voisins (config. difficile)

Dans ce paragraphe nous présentons les résultats obtenus sur un support en nid d'abeille, mais dans une configuration où l'orientation des rangées d'obstacles entrave le déplacement des agents, ainsi, ces derniers apprennent difficilement à se déplacer. En conséquence, nous présentons ce travail en détail, car dans cette configuration, du bruit apparaît dans le message que l'enfant doit apprendre.

a) Guidage de l'enseignant sur le support

L'algorithme de guidage de l'enseignant parent demeure inchangé.

b) L'algorithme de poursuite du suiveur par l'apprenant demeure inchangé.



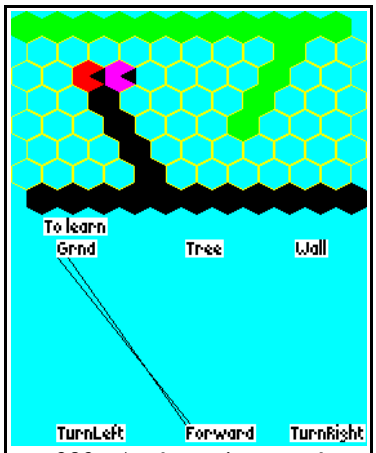
282 - Avant d'avancer

À chaque intervalle de temps élémentaire, le suiveur localise le meneur, se tourne lui, puis avance d'un pas.

<u>Avancer, quand il y a du sol devant.</u>

Regardons ci-contre l'état du monde avant l'action : le meneur en magenta vient d'avancer, il laisse un espace entre lui et son suiveur (en rouge).

Regardons à gauche, l'état des neurones avant l'action : en bas de l'image, nous pouvons voir les nœuds des neurones (au-dessus les percepts : *Grnd Tree Wall*). (au-dessous les actionneurs : *TurnLeft Forward TurnRight*).

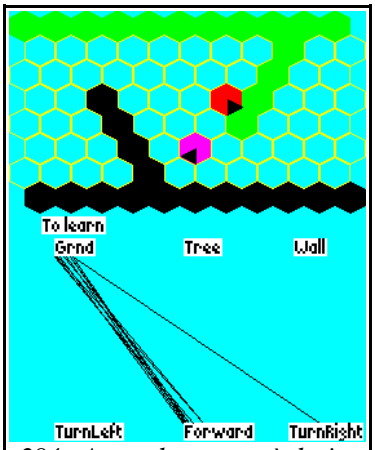


283 - Après avoir avancé

Lors du cycle précédent, le parent a avancé, son enfant l'a suivi, et ainsi il a appris la règle : *si sol devant, alors avancer*. Pendant ce cycle le suiveur entend son leader devant lui, donc il ne change pas de direction. Puis il avance.

Voici à droite, l'état du monde après l'action. Alors, en haut nous obtenons la nouvelle configuration du monde : l'enfant a rejoint son parent.

Voilà, en bas de l'image, le réseau de neurones. Il renforce sa connaissance de la règle : *si sol devant, alors avancer* : nous voyons un second segment de droite noir qui relie *Grnd* à *Forward*.



284 - Avant de tourner à droite

Tourner à droite, quand il y a un arbre devant.

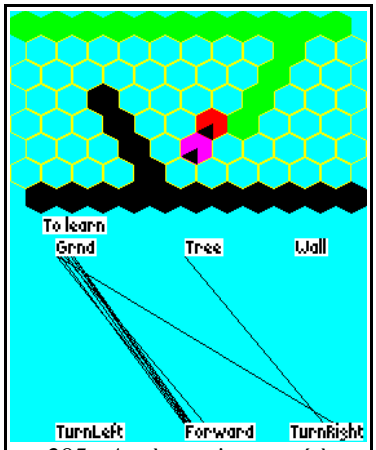
Regardons à gauche l'état du monde avant l'action : en haut, l'enfant rouge est face à un arbre. Le meneur magenta se tenait sur sa droite, et il vient d'avancer : tous deux sont maintenant séparés par un espace.

Regardons en bas, l'état des neurones avant l'action : notre centre d'intérêt n'est plus la règle à propos du percept *Grnd*, mais le percept *Tree*. Nous remarquons que pour l'instant, son apprentissage n'a pas débuté.

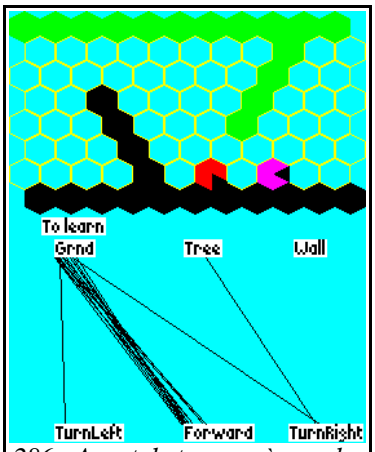
Le suiveur entend son leader sur sa droite, donc il tourne à droite et avance.

Voici à droite, l'état du monde après l'action. En haut nous obtenons la nouvelle configuration du monde : l'enfant a rejoint son parent.

Que devient le réseau de neurones en bas ? L'enfant s'est tourné à droite, puis il a avancé. Ainsi il a appris la règle : *si arbre devant, alors tourner_à_droite.*



285 - Après avoir tourné à droite et avancé



286 - Avant de tourner à gauche

Tourner à gauche, quand il y a un mur devant.

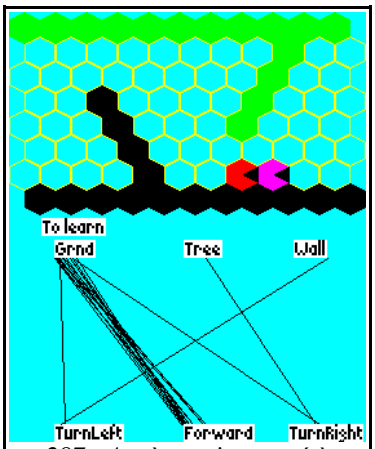
Regardons à gauche l'état du monde avant l'action : en haut, l'enfant rouge est face à un mur. Le meneur magenta se tenait sur sa gauche, et il vient d'avancer : tous deux sont maintenant séparés par un espace.

Regardons en bas, l'état des neurones avant l'action : notre centre d'intérêt est maintenant la règle à propos du percept *Wall*. Nous remarquons que pour l'instant, son apprentissage n'a pas débuté.

Le suiveur entend son leader sur sa gauche, donc il tourne à gauche et avance.

Que devient à droite, l'état du monde après l'action ? Alors, en haut nous obtenons la nouvelle configuration du monde : l'enfant a rejoint son parent.

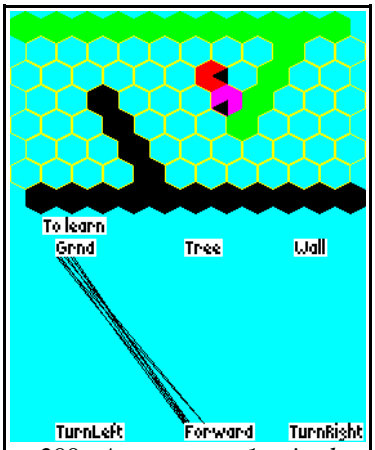
Que devient le réseau de neurones en bas ? L'enfant s'est tourné à gauche, puis il a avancé. Ainsi il a appris la nouvelle règle : *si mur devant, alors tourner_à_gauche* .



287 - Après avoir tourné à gauche et avancé

c) Analyse des résultats sur ce support à 6 voisins volontairement compliqué

Nous avons volontairement choisi de compliquer ce support en nid d'abeilles. Si dans ces nouvelles conditions, nous gardons notre algorithme de poursuite initial, alors nous constatons qu'il ne fonctionne pas parfaitement. Nous analysons deux situations sources d'erreur.



288 - Avant erreur 1 : si sol devant alors tourne à droite

Cependant, en regardant plus finement le problème, nous constatons que nous pouvons encore le conserver au prix de l'affinage de ses coefficients d'apprentissage.

Erreur 1 : pour éviter l'obstacle des arbres.

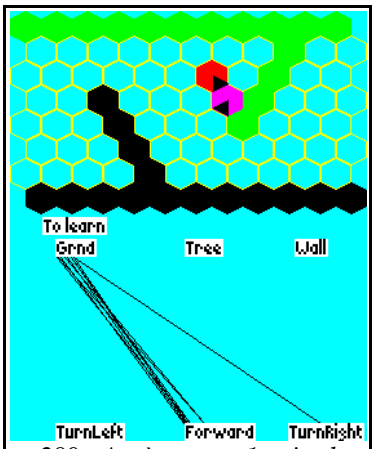
Regardons l'état du monde avant l'action : en haut à gauche, l'enfant rouge est face à une case de sol, il a sur sa droite le meneur magenta, qui a rencontré un arbre, a tourné sur sa droite, et à l'instant précédent vient d'avancer.

Regardons en bas à gauche, l'état des neurones avant l'action : notre centre d'intérêt est maintenant la règle à propos du percept *Grnd*. Nous remarquons que pour l'instant, son enseignement s'est effectué sans erreur.

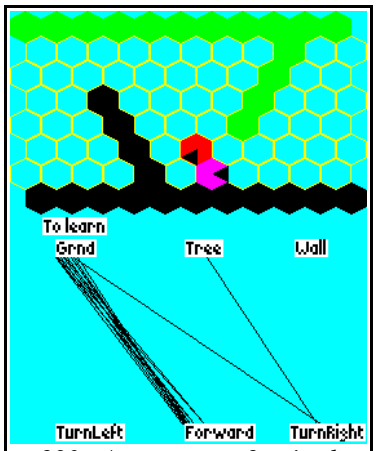
Le suiveur entend son leader sur sa droite, donc il tourne à droite.

Que devient l'état du monde après l'action ? Alors, en haut à droite nous obtenons la nouvelle configuration du monde : l'enfant, qui avait le sol devant lui, vient de se tourner vers son parent.

Que devient la série à apprendre : le réseau de neurones est sollicité par une règle fausse : *si sol devant, alors tourner_à_droite.*



289 - Après erreur 1 : si sol devant alors tourne à droite



290 - Avant erreur 2 : si sol devant alors tourne à gauche

Erreur 2 : pour éviter l'obstacle du mur.

Regardons en haut à gauche, l'état du monde avant l'action : l'enfant rouge est face à une case de sol, il a sur sa gauche le meneur magenta, qui a rencontré un arbre, a tourné sur sa gauche, et à l'instant précédent vient d'avancer.

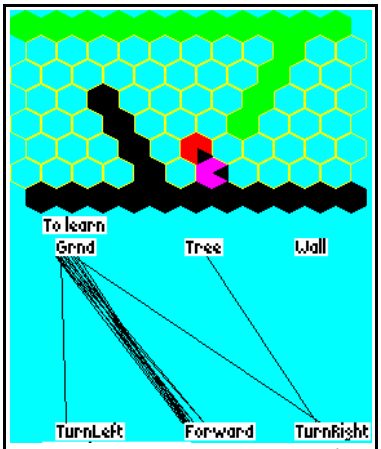
Regardons en bas à gauche, l'état des neurones avant l' action : notre centre d'intérêt est maintenant la règle à propos du percept *Grnd*. Nous remarquons que pour l'instant, son enseignement s'est effectué avec une seule erreur : l'erreur 1.

Le suiveur entend son leader sur sa gauche, donc il tourne à gauche.

Que devient l'état du monde après l'action ? Alors, en haut à droite nous obtenons la nouvelle configuration du monde : l'enfant, qui avait le sol devant lui, vient de se tourner vers son parent.

Regardons à droite ce que devient la série à apprendre : en bas, le réseau de neurones est encore sollicité par une règle fausse : *si sol devant, alors tourner_à_gauche*.

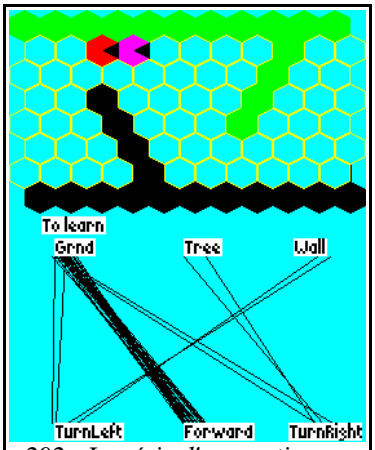
Finalement, après un tour entier de 34 cycles (tourner+avancer), nous obtenons une série d'apprentissages qui est parfaite sur 30 cycles. Ce qui signifie que nous comptabilisons 4 cycles en erreur, au niveau de la règle *si sol devant, alors avance* : deux en direction de *tourner_à_gauche,* et deux autres en direction de *tourner_à_droite.*



291 - Après erreur 2 : si sol devant alors tourne à gauche

Aménager les pondérations de l'algorithme d'apprentissage.

Cependant, nous tenons compte de la prédominance des bonnes données à apprendre : *2/2* soit 100% quand l'agent se tient en face de l'arbre, *2/2* soit 100% quand il est face au mur, et *22/26* soit 84,5% face au sol. Ce résultat explique que, sans pour autant utiliser la rétropropagation de gradient, le réseau peut encore apprendre facilement sa leçon, à condition d'aménager un peu les pondérations de son algorithme.

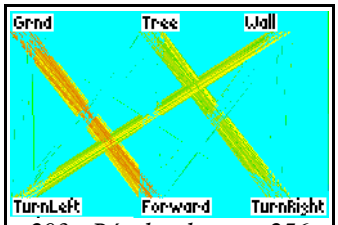


292 - La série d'apprentissage après un cycle

Calcul de l'ajustement des pondérations.

Sachant qu'à chaque cycle tous les neurones s'usent de *1*. Sachant que chaque liaison *capteur-actionneur* se renforce selon les pondérations suivantes : pour les règles concernant le capteur *sol* : *2*, pour celles concernant les capteurs *arbre* et *mur* : *20*.

Ainsi, pour les règles concernant le capteur *sol*, pendant un tour de *34* cycles chaque neurone sera affaibli de *34*. D'un côté, les deux neurones en erreur seront chacun renforcés de *2x2= 4*, ce qui est insuffisant. De l'autre, le neurone *Grnd-Forward* le renforcement sera de *22x2=44*, ce qui est supérieur de *10* à l'affaiblissement (*34*).

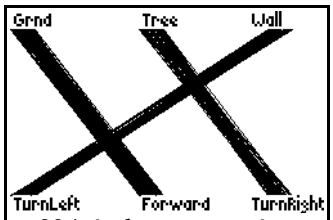


293 - Résultat brut en 256 couleurs, après 200 cycles

Résultats avec ajustement des pondérations.

Finalement, après ajustement des coefficients, au bout de 200 cycles, nous obtenons, le résultat brut illustré ci-dessus à gauche en 256 couleurs : nous devinons que les trois règles principales sont bien apprises.

Après un redressage du résultat obtenu en comparant le poids de chaque neurone *capteur-actionneur*, à la valeur médiane 50%, nous obtenons le résultat contrasté et sans ambiguïté, illustré à droite en noir et blanc.



294résultat contrasté en N&B

d) Au vu des ces résultats, quelles conclusions tirer ?

Pour tester la robustesse du montage *2DApprendRègles*, nous avons compliqué le problème à résoudre. Finalement, au prix de l'affinage de ses coefficients d'apprentissage, nous réussissons encore à le faire fonctionner, sans être obligé d'avancer des hypothèses fortes : ceci montre à quel point il est robuste.

5) Conclusions

5.A) Make it simple, stupid : simple is beautifull

Avec cette simulation si légère, nous sommes en présence d'une réalisation difficile à appréhender. Elle est si simple que nous risquons de nous tromper et de ne pas discerner son intérêt. Ce n'est pas un jouet de poupée, au contraire, elle est construite avec un agent situé réactif tropique, et selon un point de vue global, elle modélise l'enseignement d'un apprentissage[86].

5.B) Outil pédagogique

Selon un point de vue pédagogique, le fait que cette démarche soit globale présente un intérêt certain.

Une simulation qui s'anime devant l'étudiant.

Cette simulation est présentée sous la forme d'un outil informatique interactif : à chaque cycle, sur l'écran, des êtres multicolores s'animent.

Une démarche d'enseignement.

En IA, nous connaissons surtout la démarche classique d'apprentissage, où le programme apprend une série de données externes. Elles ont été précédemment recueillies ailleurs, et mémorisées afin d'être fournies au système. Pour ces raisons, ce genre d'expérience est artificiel : nous sommes en présence d'une démarche d'apprentissage coupée de la réalité.

Par contre, avec *2DApprendRègles*, à la démarche d'apprentissage, nous ajoutons la simulation d'un parent, qui, dans son rôle d'enseignant, montre l'exemple afin de transmettre, cycle après cycle, un savoir. Et, l'apprenant les assimile en direct sous nos yeux ; ainsi, quand un fonctionnement nous échappe, il nous est loisible de disséquer le processus à discrétion : nous pouvons rejouer la scène en demandant à l'ordinateur de fournir plus de détails.

Apprentissage neuronal, au moyen d'un algorithme très simple.

Classiquement les réseaux de neurones ont la réputation d'être difficiles à enseigner. Voici l'avis d'un couple bien informé sur ce sujet :

```
 Les réseaux de neurones sont complexes, à moins de s'en tenir à 2
couches, une couche d'entrée de N neurones, et une couche de sortie à 1 ou
2 neurones.
 Les techniques d'apprentissage sont très matheuses, et si on fait
l'impasse dessus, on vide ces structures de leur intérêt.
 Il en va de même pour les algorithmes génétiques, qui font appel à des
processus d'optimisation pointus, et qui se réduisent à une leçon de chose
si on ne les aborde pas.
```

46 : L'avis d'un couple bien informé[87] sur les réseaux de neurones

À l'encontre de cette position, nous proposons donc des simulations sur les réseaux de neurones et algorithmes génétiques. L'approche globale que nous mettons en œuvre, ne constitue pas une coquille vide. Au contraire, elle confère un sens à la simulation toute entière.

De plus, quand nous enseignons dans le cadre de formations généralistes, il ne faut pas rougir de parfois faire des leçons de choses.

86 Elle comporte deux parties : le volet enseignement, et celui de l'apprentissage.

87 Elle est responsable pédagogique dans une grosse structure (7000 étudiants/an). Lui a monté sa société dans les réseaux de neurones.

5.C) Transmission de règles épigénétiques

C'est une transmission culturelle

Selon un point de vue épistémologique, nous en sommes arrivés à une transmission, entre deux générations, de règles épigénétiques, i.e. de règles dont l'information n'est plus codée dans le patrimoine génétique, mais dans les couches culturelles situées dessus lui (du grec épi = sur). À ce stade, nous sommes en présence d'une transmission culturelle, et en l'étudiant, nous faisons de la mémétique.

Une citation pour souligner l'intérêt de la transmission culturelle : de Karl Popper.

L'évolution culturelle continue l'évolution biologique par d'autres moyens - Karl Popper (1902-1994).

Comparée à la transmission d'un patrimoine génétique entre deux générations, l'intérêt de la transmission culturelle entre deux individus repose sur la rapidité de son cycle. Du côté génétique un cycle dure une génération, de l'autre, le cycle d'une transmission culturelle est dix, cent ou mille fois plus rapide. En conclusion la transmission culturelle court-circuite celle de la génétique.

6) Transition

À la prochaine étape de cette progression de simulations, nous introduirons la modélisation d'une transmission d'information entre générations encore plus rapide : après la transmission culturelle par l'exemple du geste, nous traiterons celle effectuée au moyen du langage.

7) Perspective

2DApprendRègles est basé sur un algorithme d'apprentissage, où une sortie *S* est fonction d'une unique entrée *E*. En utilisant un algorithme amélioré, nous pourrions simuler des transmissions culturelles plus complexes, par exemple l'itinéraire parcouru par un groupe de chasseurs-cueilleurs au milieu d'obstacles. Il suffirait simplement d'utiliser des connexions un peu plus ramifiées où une sortie *S* pourrait être fonction de plusieurs entrées. Certes cette ramification complique un peu l'écriture du programme, mais cette simulation demeure réaliste.

Ch. 40 *TransmetCulture* : un parent transmet des règles à un enfant

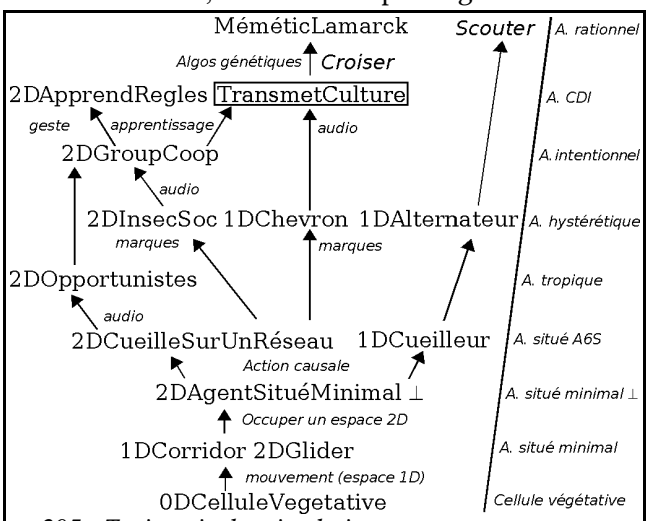
1) Introduction

La culture se transmet au moyen du langage.

D'un côté cette simulation, nommée *TransmetCulture*, constitue un prolongement direct de notre progression, mais de l'autre, elle se situe en rupture.

Continuité, parce qu'elle prolonge directement notre progression de simulations. Comme dans *ApprendRègle* nous y retrouvons des agents situés, simplifiés à l'extrême, sur un support très épuré, et ici encore, le parent transmet une règle de conduite à l'enfant.

Mais cette simulation se situe en rupture, car le réseau est construit différemment, il est constitué de chevrons ; et rupture encore car maintenant, le média de la communication change : à la place du geste, les agents utilisent le langage.



295 - Taxinomie des simulations que nous proposons

2) Problématique : introduire une communication langagière simplissime

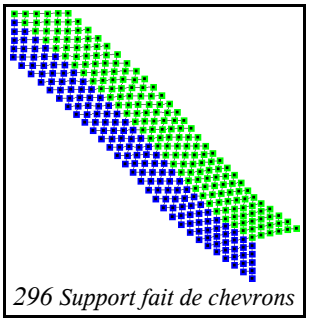
Trouver un monde où les agents sont simples.

Dans cette simulation, le réseau de lieux est construit au moyen de chevrons (figure 3). Cette architecture un peu plus complexe, permet d'utiliser un agent alternateur encore plus simple que celui de *1DAlternateur*. À ce prix, le codage d'une règle de conduite devient très basique et se réduit à la transmission de quelques symboles. Cet aménagement permet l'utilisation d'un langage minimaliste, pour une communication *parent-enfant* très simplifiée.

Simuler des transmissions culturelles entre générations.

Donc dans ce chapitre, nous simulons des échanges culturels *parent-enfant*, puis évaluons l'adaptation des enfants au milieu. Ceci nous permet de tester indirectement l'intérêt de la transmission culturelle entre générations.

3) Les modélisations utilisées dans cette simulation


296 Support fait de chevrons

3.A) Modélisation du support

Un support constitué de chevrons.

Le support est constitué d'un assemblage de chevrons identiques. Mais nous pouvons imaginer d'autres simulations réalisées

avec des topologie très diverses. Leur taille peut varier et au milieu, les cases centrales peuvent disparaître.

Un support *cylindrique.*

En assemblant les chevrons, nous obtenons un support doté d'un axe oblique qui descend vers le sud-est. Le dernier chevron en bas est chaîné avec le premier en haut, mais pour la clarté du dessin, les liens ne sont pas représentés à l'écran.

3.B) Modélisation du mobilier : les matières

Dans cette simulation, nous ne plaçons pas d'obstacle sur le support, mais seulement des matières : de l'eau (en bleu) et de la nourriture (en vert). Quand une ressource est consommée, elle disparaît momentanément, puis elle se renouvelle au bout de quelques dizaines d'instants élémentaires. Ceci explique la zone vide, tournée vers le sud-est, qui apparaît sur le support, quand les agents se font nombreux : à peine l'herbe a-t-elle repoussé qu'elle est consommée (il en va de même pour l'eau).

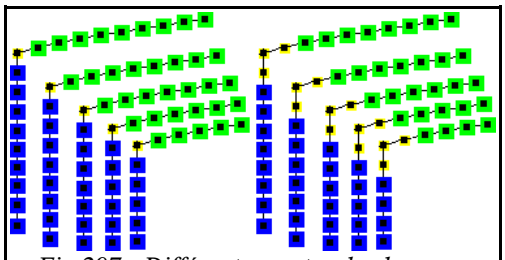


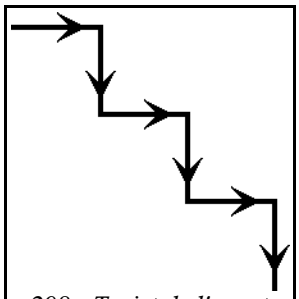
Fig 297 - Différentes sortes de chevrons

3.C) Modélisation du mobilier : l'agent

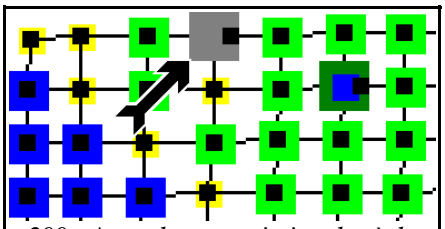
La gestion du déplacement de l'agent est réactive.

L'agent ne peut aller que vers le sud ou l'est, i.e. il ne peut que descendre puis tourner vers sa gauche ; ou aller vers la droite de l'écran puis tourner à descendre.

L'agent est réactif. Par exemple, son comportement peut être codé au moyen des règles suivantes : *si l'agent a faim, il fixe son cap à l'est* ; et *s'il a soif, il se dirige au sud.*



298 - Trajet de l'agent



299 - Avant la transmission de règle

L'agent naît immature.

À sa naissance, l'agent est immature. Il ne sait pas gérer son déplacement car son câblage cérébral est partiellement inachevé. Son immaturité est affichée à l'écran dans son dessin. Par exemple, ci-contre, nous le voyons dessiné au milieu de la première ligne, au moyen d'une couleur grise unie (flèche oblique noire).

L'agent communique.

Nous sommes en présence d'un agent *Croyances* qui communique : il parle, écoute, entend, comprend, et mémorise ces échanges au sein de son *savoir* (de ses croyances).

La communication se déroule selon ce protocole : l'agent immature écoute son environnement. S'il entend un adulte qui agit et commente cette action, l'enfant l'enregistre. Par exemple, il câble son signal de *faim* sur l'actionneur *tourner à gauche*. Ainsi il possède quelques croyances sur le monde : ce sont ses règles de conduite. Elle sont acquises et accessibles. Il y accède en écriture quand il apprend, et en lecture quand il parle.

Syntaxe du (proto) langage.

L'agent mature commente ses actions en utilisant un langage très simple dont la syntaxe[88] est la suivante :

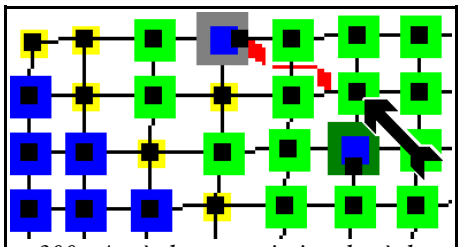
règle ::= condition action
condition::= Faim

88 Les constantes commencent par des majuscules ; les variables, par des minuscules.

condition ::= Soif
action ::= TourneADroite
action ::= TourneAGauche

Exemple de l'apprentissage pour étancher sa soif

L'être immature sait écouter et comprendre ce que lui disent les adultes quand ils commentent leurs propres actions.


300 - Après la transmission de règle

Ci-contre, un agent mature se tenait sur la case avant la fin de la ligne 2 (flèche oblique noire). Il a dit : *j'ai soif alors je tourne à droite*. L'enfant immature de la ligne 1 qui se tenait suffisamment près de lui, a entendu cette règle, il l'a enregistré et par la suite, il se comportera en conséquence.

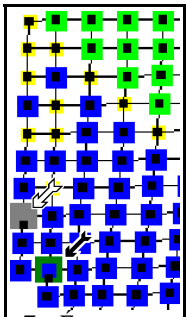
Dans cette illustration, la liaison audio entre l'enfant et la position où était l'adulte est matérialisée par un lien rouge, qui ensuite a été partiellement recouvert par l'affichage à l'écran des cases du support.

Maintenant, à l'issue de cet échange, l'enfant sait comment étancher sa soif. Ainsi il a effectué la moitié de son apprentissage de déplacement, et donc la moitié de son affichage à l'écran (son carré central) recopie la couleur de son professeur.

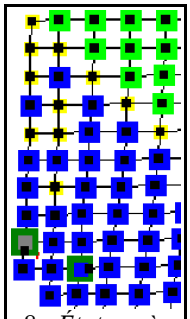
Ensuite, l'enseignant avance. Dans l'illustration ci-dessus à droite, nous le retrouvons en ligne 3, pointé vers le sud.

Exemple de l'apprentissage pour manger.

Voici encore un autre exemple où l'être écoute et comprend ce que lui disent les autres agents quand ils commentent leur propres actions.


7 - État avant

Dans l'illustration de gauche, un agent mature, situé en colonne 2, une ligne avant le bas (flèche oblique noire), dit : *j'ai faim alors je tourne à gauche*. À côté de lui, nous voyons un enfant immature (flèche oblique blanche). Il se tient suffisamment près pour entendre cette règle, alors il l'enregistre et, par la suite, il se comportera en conséquence.


8 - État après

Dans l'illustration de droite, l'apprenant a avancé. L'agent mature a tourné à gauche et a fait de même. Nous le retrouvons en colonne 3, tourné vers l'est. La liaison entre l'enfant et la position où était l'adulte, est matérialisée par un lien rouge (ensuite elle a été presque toute recouvert par l'affichage des cases).

À l'issue de cet échange, l'enfant sait comment se restaurer. Ainsi il a effectué la moitié de son apprentissage de déplacement, et donc, la moitié de son affichage à l'écran, i.e. son cadre externe, recopie la couleur *vert caca d'oie*, de son professeur.

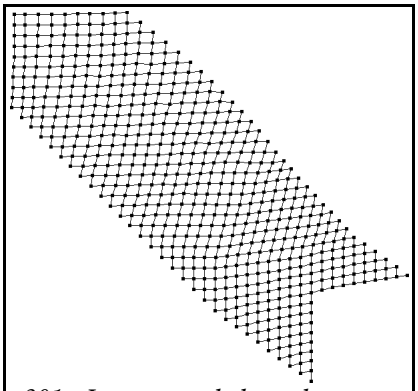
L'agent se reproduit.

Dans ce modèle, la reproduction est asexuée et ressemble à un bouturage : les agents sont tous semblables et dès qu'un d'eux a stocké beaucoup d'énergie, il devient assez gros, et se reproduit identique à lui-même (sauf, bien sûr, pour les fonctions de gouvernance puisque l'enfant naît immature à ce sujet).

4) Implémenter ces fonctionnements

4.A) Implémenter le support, un tableau unaire de chevrons

L'implémentation du support diffère de l'ordinaire à cause de l'utilisation d'une architecture en chevrons. Dans les simulations précédentes, les réseaux ou grilles étaient obtenus au moyen d'un tableau 2D de lieux. Ici il est fabriqué au moyen d'un tableau 1D de pointeurs sur une structure *chevrons*, décrite au début du chapitre, figure 3.



301 - Le support de base des cases

```
#define ChevronNbr 30

chevron * GridTabCase [ChevronNbr] ;
// Déclare un tableau de pointeurs sur des chevrons
// et alloue la place mémoire pour lui
```

47 La nouvelle table pour construire la grille au moyen de chevrons

4.B) Ne pas programmer la communication mais en simuler le résultat

Dans cette simulation, la communication entre les agents se complexifie largement : nous en sommes arrivés à des échanges syntaxiques. Comment est-elle programmée ?

Pour cela, nous avons fait au plus simple et réutilisé la méthode décrite pour les agents du groupe communicant. Loin de programmer en détail la chaîne de traitement de l'information[89], nous avons seulement simulé les résultats de cette communication.

```
SwarmTabAgent_CepAgentTurns_IlEnseigneATousLesJeunesAgentsProchesDeLui(Age
nt* pRefAgent)
Pour chaque agent immature de l'essaim
 si il n'est pas mort
 et
 si ce n'est pas lui-même
  faire :
   si la distance (agent, agent de référence) < distance de référence
   alors,
    selon le changement de direction effectué,
    forcer le drapeau correspondant chez l'agent apprenant
   sinon
    ne rien faire
  fin de faire
fin de si
```

Table 48 : La procédure pour simuler le transfert de règles entre parent et enfant.

4.C) Implémenter des agents : rajouter des champs qui sont testés à l'exécution

Maintenant, les agents sont implémentés avec des champs nouveaux.

Pour cette simulation, nous reprenons les agents classiques et les implémentons avec des champs nouveaux :

```
struct Agent_ {
```

89 La génération de la phrase, la modulation, l'émetteur, la propagation, la réception, la démodulation, l'analyse morpho-lexicale, l'analyse syntaxique.

```
...
  int BehaviorEst;                    // Son comportement pour aller à l'est
  int BehaviorSud;                    // Son comportement pour aller au sud
...
};
```

Table 49 : Dans la structure de l'agent nous ajoutons deux nouveaux champs

Ces champs nouveaux sont testés par la procédure suivante dans le module réactif des agents :

```
Si il a trop faim ou soif
 alors l'agent meurt de faim ou de soif

Sinon si le niveau d'énergie est au-dessous du niveau d'alerte,
 alors change de direction : fcap est
Sinon si la réserve d'eau est au-dessous du niveau d'alerte,
 alors change de direction : fcap sud
Sinon si rien dans la case devant
 alors avance
 sinon inhibition.
```

Table 50 : La procédure qui détermine la fixation et le changement de cap des agents

4.D) Implémentation de la reproduction

La reproduction est le fait de la cellule végétative initiale, elle est donc traitée, à chaque cycle, au niveau somatique, i.e. en priorité, au début de l'action de l'agent.

```
void Agent_AgisA7Date_InCeScreen(Agent * pAgent, int date, SDL_Surface * screen) {
...                                                  // Puisqu'il n'y a rien devant et
 if (pAgent->Poid_SigmaFoodEau>MaxPoidAvantEnfanter){ // si l'agent est costaud alors il enfante
  CreelAgentALImageDeCeluiCiEtPousseLeDansSwarmTabAgent(pAgent);
  pAgent->Poid_SigmaFoodEau= 0;
  fprintf(stderr, "CreelAgentALImageDeCelui\n");
 }
}
```

Table 51 : Dans la structure de l'agent nous ajoutons deux nouveaux champs

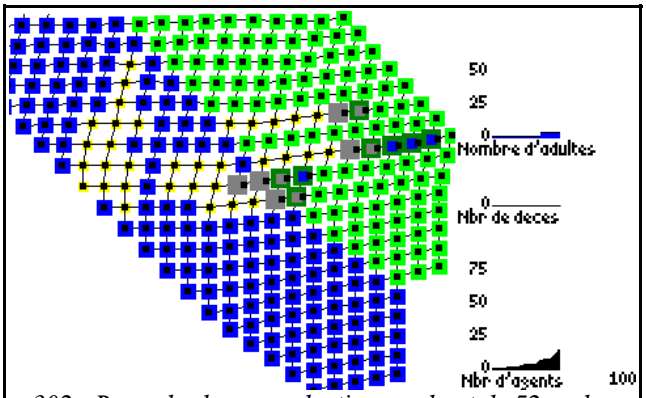
La reproduction consiste à créer un nouvel agent. Nous y dupliquons tous les champs de son modèle (sauf ceux de la gouvernance), et nous le poussons sur la pile d'agents.

```
void CreelAgentALImageDeCeluiCiEtPousseLeDansSwarmTabAgent(Agent * pAgentParent) {
 Agent * pNewAgent;
 if (InxSwarmTabAgent>=AgentNbr)  {        // traitement du dépassement de capacité de la pile
  fprintf(stderr, "================= stackOverFlowToCreateANewAgent ==================\n");
  return;
 }

 pNewAgent= (Agent *) malloc(sizeof(Agent));    // Alloue de la place mémoire pour cet agent

 pNewAgent->InxDeAgentDansSwarmTabAgent=InxSwarmTabAgent; //tt agt sait son inx ds SwarmTabAgent
 pNewAgent->Direction= pAgentParent->Direction;  // Initialisé dans la même directi° que son parent

 pNewAgent->EnergyLevel= StartingEnergyLevel;    // Le niveau d'énergie de la cel vég d'1 agent
 pNewAgent->EauLevel   = StartingEauLevel;       // Le niveau d'eau de la cel vég d'un agent
 pNewAgent->Poid_SigmaFoodEau= 0;                // Le poids de l'agent vu comme
                                                 // proportionnel à la somme de ses ressources
                                                 // energy/eau MaxPoidAvantEnfanter
 //pNewAgent->ToDrawBumped;                      //Inutile d'init il est RAZ au début de l'act°

 pNewAgent->SocialPosition=pAgentParent->SocialPosition;// La position sociale de l'agent
 pNewAgent->Couleur     = pAgentParent->Couleur; // La posit sociale de l'agent détermine sa color

 pNewAgent->BehaviorEst = Indetermine;           // enum Comportement {Indetermine, PrimTemeraire}
 pNewAgent->BehaviorSud = Indetermine;           // A la naissance les agents ne savent par tourner


 pNewAgent->pCaseOuIAm= pAgentParent->pCaseOuIAm;// ptr sur la case où est l'agent

 SwarmTabAgent[InxSwarmTabAgent++]= pNewAgent;   // Une fois l'agent terminé, le pousse sur la pile
};
```

Table 52 : Le nouvel agent est construit à l'image de son parent (sauf pour la gouvernance).

5) Présenter les résultats

5.A) Où les agents se reproduisent

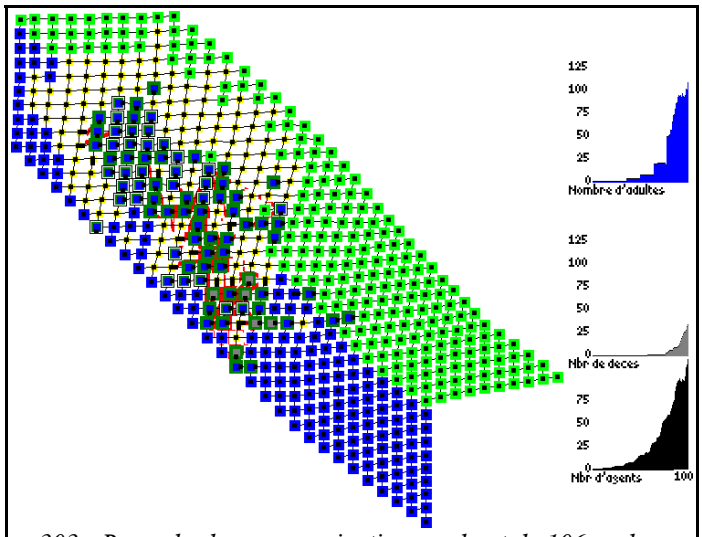


302 - Regarder les reproductions au bout de 52 cycles

L'état de départ.
Nous commençons avec un unique agent placé tout en haut, qui part plein sud. Au bout de 52 cycles il constitue déjà une troupe de 13 agents (4 adultes, 4 adolescents et 5 enfants).

Le renouvellement des ressources consommées.
Comme *Attila*, ils laissent une trace dans le paysage, car, là où ils sont passés, l'herbe et l'eau ont disparu. Cependant, dans cette simulation, après un délai, nous pouvons remarquer que les réserves se reconstituent, i.e. que l'herbe repousse et que l'eau reparaît.

5.B) Les agents communiquent



303 - Regarder les communications au bout de 106 cycles

L'état de départ.
Nous commençons avec la même configuration, mais nous s'arrêtons au bout de 106 cycles.

La croissance de la population.
Au début, la population d'adultes passe par les étapes 2, 5, 9, 21, 51... Nous sommes en présence d'une croissance presque exponentielle de leur population.

5.C) Les collisions entre agents

Les agents qui progressent devant, sur le front, se déplacent encore assez librement, mais ceux qui se tiennent derrière, sont pris dans des embouteillage. À l'écran, il apparaissent entourés d'un rectangle noir[90].

Le renouvellement des ressources consommées.
À cette date, tant d'agents sont sur le terrain que les ressources n'ont pas le temps de se renouveler et manquent. Ainsi, à droite de l'écran, l'histogramme gris du milieu montre que 35 agents sont morts.

Lors des naissances, la transmission de règles est très dense.
À la date 106, beaucoup d'agents viennent de naître, c'est ce qui explique la densité des communications pour les éduquer. À l'écran, les traces rouges correspondantes demeurent encore visibles entre les cases.

90 Dans les premières simulations, programmées en Java, à l'écran, il apparaissaient entourés d'un rectangle jaune.

5.D) Analyse des résultats

a) Nous simulons une transmission culturelle, et testons son efficacité

L'apprentissage effectué par les enfants est immédiat et définitif.

De façon innée, l'enfant est programmé pour enregistrer le premier conseil donné par un adulte qui s'exprime dans son environnement. Puis, même si ensuite il est sollicité par d'autres agents, il ne change pas d'avis.

Remarque : Cette modélisation ne pose pas de problème ici, car, sur le support, nous ne mettons en situation que des agents semblables, mais ensuite, dans le prochain chapitre, les choses changeront...

L'apprentissage des agents immatures n'est pas parfait ?

À sa naissance l'enfant hérite des propriétés de son parent : il est posé au même endroit et dans la même direction. Donc il suit son parent qui lui enseigne comment tourner. Mais il est possible que son apprentissage ne se fasse pas parfaitement.

Il peut arriver qu'un enfant soit mal éduqué, ou qu'il ne reçoive pas de programmation. Par exemple, s'il est bloqué dans un embouteillage, l'enfant peut être distancé par son parent, et apprendre une portion de culture d'un autre adulte, qui peut appartenir à une autre ethnie.

Un apprentissage de mauvaise qualité produit des conséquences dramatiques.

Dans de telles conditions, si un enfant ne peut apprendre à tourner pertinemment, alors il file tout droit, s'éloigne du groupe et butte contre les limites du support où il dépérit.

Mais finalement, ceci arrive rarement et en fait, la surpopulation cause beaucoup plus de morts que le défaut de programmation des agents immatures.

Conclusion : Simuler et tester des transmissions culturelles entre générations.

Donc dans ce chapitre, nous simulons des échanges culturels *parent-enfant*, puis après qu'ils sont éduqués, nous évaluons l'adaptation des enfants au milieu. Ceci nous permet de tester indirectement l'efficacité et l'intérêt de la transmission culturelle entre générations.

b) Résultat relatif au traitement du langage naturel

Dans son cours de DEA, Gérard Sabah nous a demandé un jour quel est le composant le plus intéressant au sein de la phrase : sujet, verbe, compléments ou encore quelque chose d'autre ? Maintenant, que dans notre progression de simulations nous arrivons à la simulation actuelle, nous pouvons fournir une réponse, car elle est la première à modéliser les formes les plus simples d'un (proto) langage.

À ce stade, regardons ce parent dont la règle de conduite est celle-ci :

```
Si le niveau d'énergie est au-dessous du niveau d'alerte,
 alors change de direction : fcap est
Sinon si la réserve d'eau est au-dessous du niveau d'alerte,
 alors change de direction : fcap sud
```

Table 53 : Une partie de la procédure qui détermine la fixation et le changement de cap des agents

Quand il commente ses actions, il communique en transmettant une donnée structurée contenant deux symboles :

```
faim fcap_est
soif fcap_sud
```

Tab 54 : Les deux conseils bruts, fournis par les agents, quand ils changent de cap

Quand nous verbalisons ces échanges très simples, nous les exprimons au moyen de pseudo-phrase et obtenons :

```
J'ai faim je pars à l'est
J'ai soif je pars au sud
```

Tab 55 : Nous verbalisons les conseils échangés par les agents, à propos de la gestion de leur cap

Le résultat est sans appel. D'abord apparaît le verbe d'action *fixer le cap à* ou *partir à,* qui est central à la phrase. Ensuite viennent les modalités de direction : *à l'est* ou *au sud*. Puis viennent les conditions de l'action : *j'ai faim* ou *j'ai soif*.

Finalement vient le sujet. Il n'apparaît pas dans le (proto) langage échangé par les agents, mais seulement ensuite dans sa verbalisation. Il est donc moins important, car il demeure toujours le même, en effet c'est une déictique : *je*.

6) Conclusions à propos de la structure du support

Nous travaillons dans un support de chevrons. Certes ce monde semble bizarre, mais c'est le prix que nous avons dû payer. Cette modélisation, si rapide et si légère permet plus facilement de simuler le transfert culturel.

7) Transition

Dans la fin de ce chapitre, l'évolution de la progression est décrite en filigrane. Sur notre support, nous allons mettre en scène des espèces différentes, des agents à la structure et aux circuiteries plus complexes. Ainsi nous obtiendrons un traitement de l'information nouveau et plus efficace.

8) Perspective

En effet, nous connaissons déjà les AG qui mixent et évaluent des comportements instinctifs. Ainsi, dans la simulation *MemeticLamarck*, que nous décrivons dans le chapitre 33_*MemeticLamarck*, nos mettons en scène des agents d'ethnies différentes, où l'enfant est sollicité par plusieurs cultures : quand le conseil est avisé, alors l'être est adapté et survit ; sinon, il meurt rapidement.

En conclusion, cette simulation d'agents situés nous fournit un outil qui teste indirectement la justesse des comportements métissés des agents.

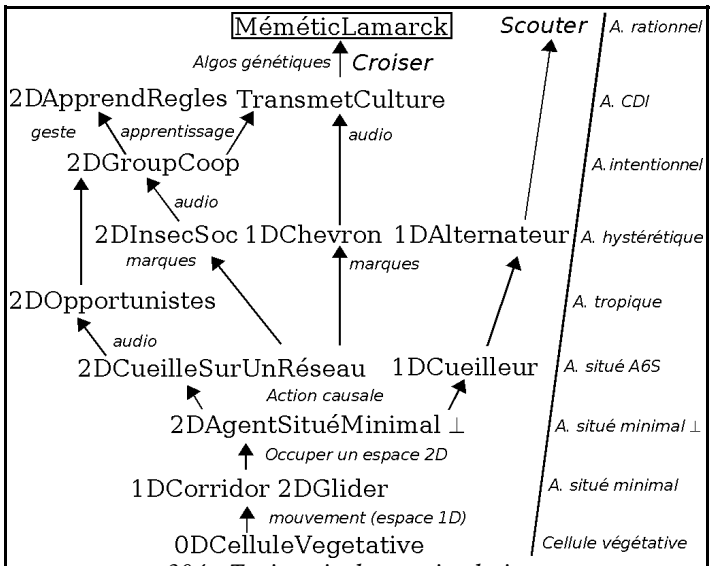
Ch. 41 *MémétiqueLamarck,* transmettre le jeu de règles qu'on a créé

1) Introduction : effectuer un traitement cognitif facile à implémenter

a) Position : Récolter le fruit de notre travail

Cette simulation, nommée *MémétiqueLamarck*, constitue un prolongement direct de notre progression. Elle se positionne même comme une feuille[91] commune aux extrémités de deux branches[92] et constitue un aboutissement.

C'est maintenant que nous allons récolter le fruit de notre travail en réalisant un système qui effectue un traitement cognitif facile à implémenter.



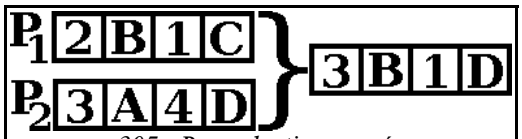
304 - Taxinomie de nos simulations

b) Les algorithmes génétiques

Dans cette simulation nous utilisons les algorithmes génétiques (AG), alors faisons d'abord un retour rapide sur ce sujet.

La reproduction sexuée.

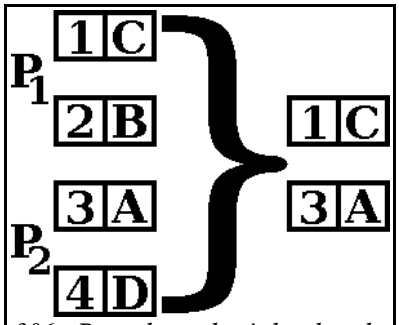
Dans le cadre de la reproduction sexuée, les deux parents apportent un patrimoine génétique plutôt différent. La *méiose* construit celui de l'enfant en empruntant séquentiellement un caractère soit à un parent, soit à l'autre.



305 - Reproduction sexuée

Exemple de reproduction sexuée.

Dans l'illustration ci-contre, le parent P_1 apporte la séquence *(2 B 1 C)*, et P_2 apporte *(3 A 4 D)*. À droite nous observons le patrimoine de l'enfant : *(3 B 1 D)*. Il est construit séquentiellement en empruntant un caractère successivement à P_2, P_1, P_1, P_2.



306 - Panachage de règles dans la simulation MéméticLamarck

Utiliser les AG pour effectuer un traitement bon marché.

Dans la simulation *MémétiqueLamarck* nous regardons les séquences comme des règles de système formel ; et nous

91 En effet, après *MémétiqueLamarck*, dans notre taxinomie, nous ne trouvons plus que *Croiser*, qui ne constitue pas une vraie simulation, mais qui se positionne plutôt comme un jalon, qui montre et explore le chemin à suivre.

92 En fait, *MéméticLamarck* apparaît à cheval sur deux domaines car elle se sert de la communication pour traiter de l'information.

utilisons les AG pour réaliser sur elles un traitement cognitif facile à implémenter. Mais au sein du programme nous effectuons

seulement un traitement grossier : nous ne créons pas de séquence, i.e. pas de règle, mais nous nous contentons de les panacher.

Exemple de transmission avec panachage de règles.

Dans l'illustration ci-contre, le parent P_1 apporte les deux règles $1 \rightarrow C$ et $2 \rightarrow B$. De même P_2 apporte $3 \rightarrow A$ et $4 \rightarrow D$. Alors, à droite de l'accolade, nous observons le patrimoine de l'enfant : $1 \rightarrow C$ et $3 \rightarrow A$:

Mais comment ce résultat a-t-il été construit ? Finalement nous déduisons que sa règle du haut $1 \rightarrow C$ est construite en empruntant à P_1 ; et celle du bas $3 \rightarrow A$ à P_2.

1.B) Position de cette simulation dans notre progression de modèles

Dans la taxinomie de nos simulations, la feuille *MémétiqueLamarck* prolonge l'étape *TransmetCulture* car elle utilise le même monde, i.e. le même support en chevrons et les mêmes agents communicants.

De plus, *MémétiqueLamarck* se place au bout de branche, en position de feuille car elle constitue l'étape ultime et le résultat final de notre progression vers le traitement de l' information : elle illustre un modèle de traitement cognitif léger et très simple.

2) Quels outils informatiques pour modéliser cette simulation ?

Reprendre la simulation *TransmetCulture (MemeticTransmetPrimairePrimaire).*

Pour obtenir le résultat final *MémétiqueLamarck* nous reprendrons la simulation intermédiaire *TransmetCulture (MemeticTransmetPrimairePrimaire)* et nous la prolongerons au moyen d'une simple modification : nous partirons avec deux espèces différentes d'agents.

Introduire deux espèces différentes d'agents.

Pour ce faire, nous modifierons le programme, et nous l'initialisons avec les deux populations d'agents correspondantes.

3) Comment programmer deux espèces différentes d'agents

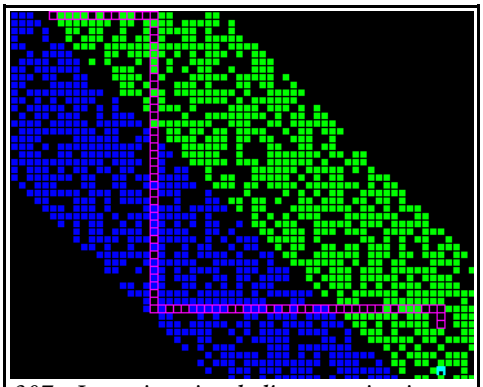
Maintenant regardons les modélisations que nous avons utilisées. Voyons comment nous avons programmé deux espèces différentes d'agents.

3.A) L'espèce initiale : des agents primaires inconséquents

Initialement, dans la simulations *TransmetCulture*, nous avons placé sur la grille des agents dont le comportement codé au moyen de règles, se verbalise ainsi :

- Si l'agent a faim, il fixe son cap à l'est ;
- Si l'agent a soif, il se dirige au sud.

En fait, si nous analysons la justesse de leur comportement, nous pouvons conclure qu'ils ne sont pas parfaitement adaptés à la situation. Ces agents sont primaires. Une fois leurs pulsions internes assouvies, il se laissent aller et ne prennent pas d'initiative : ils vivent au jour le jour car ils attendent qu'une matière leur manque pour tourner et partir à sa recherche. Ce n'est que quand un besoin apparaît qu'ils tournent.



307 - La trajectoire de l'agent primaire est plus ample : il tire de grands bords

À cause de ce mode de fonctionnement, l'agent s'enfonce à l'intérieur des bandes de matière, loin de la ligne médiane ; et quand le besoin d'une ressource se fait sentir, avant de pouvoir la

consommer, il doit d'abord quitter cette zone où il trouve une matière qui lui est inutile. L'agent primaire tourne donc moins souvent, sa trajectoire est plus ample ; un marin dirait qu'il tire de grands bords.

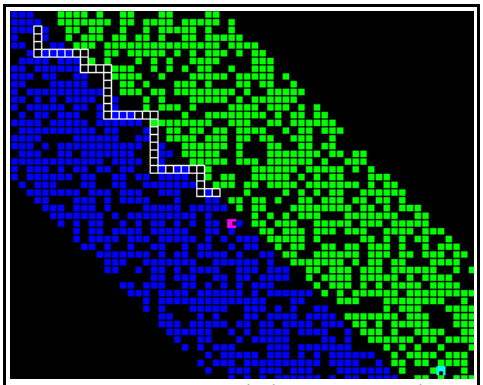
Quand enfin il atteint la zone de la ressource convoitée, il se peut que les premières cases soient déjà consommées par les autres. Donc, nous concluons qu'en agissant ainsi, il augmente son risque de mourir : il est inconséquent, et finalement s'avère moins robuste, moins adapté.

Une espèce mieux adaptée : les agents secondaires finauds et intelligents.

Maintenant, sur la grille analysons les trajectoires des agents dont le comportement peut être codé au moyen des règles suivantes :

- Si l'agent est désaltéré, il fixe son cap à l'est ;
- Si l'agent est rassasié, il se dirige au sud.

En fait, si nous analysons la justesse de leur comportement, nous pouvons conclure qu'ils sont mieux adaptés à la situation. Ces agents sont secondaires. Ces êtres finauds tournent dès qu'il sont rassasiés. Une fois leur pulsion interne assouvie, il ne se laissent pas aller et prennent l'initiative de se fixer un nouveau cap : ils anticipent car ils n'attendent pas qu'une matière leur manque pour tourner et partir à sa recherche. Dès que leur besoin actuel est assouvi, ils tournent pour satisfaire l'autre.



308 - La trajectoire de l'agent secondaire est moins ample, il tire de petits bords

Grâce à ce fonctionnement, l'agent s'enfonce peu à l'intérieur des bandes de matière. Il tourne souvent. Sa trajectoire est moins ample, plus hachée ; un marin dirait qu'il tire de petits bords.

Ainsi le finaud demeure près de la ligne médiane. Il vit donc sur la frontière entre les deux zones, alors, quand un besoin se fait sentir, il est (presque) sur place et peut consommer rapidement. Ainsi, plus adapté, il s'avère plus robuste.

3.B) Implémenter des agents, rajoutons des champs qui sont testés à l'exécution

Dans le cadre de cette simulation, afin de modéliser une transmission lamarckienne, nous implémentons des espèces nouvelles. Pour ce faire, nous utilisons les champs que nous venons d'ajouter aux agents.

Nous avons déjà vu que les agents sont implémentés avec des champs nouveaux :

```
enum Comportement {Indetermine, PrimTemeraire, SecoOpportun};   //

struct Agent_ {
...
  int BehaviorEst;  // Son comportement pour aller à l'est
  int BehaviorSud;  // Son comportement pour aller au sud
...
};
```

56 : Dans la simulation TransmetCulture*, nous avons déjà ajouté ces champs à la structure de l'agent*

Ces champs nouveaux sont testés par les modules réactifs des agents.

Lors de l'exécution de la simulation, à chaque intervalle élémentaire de temps, ces champs nouveaux sont testés par les modules réactifs des agents. Ainsi ils permettent de générer les nouveaux comportements primaires et secondaires que nous avons décrits.

```
Si il a trop faim ou soif
 alors l'agent meurt de faim ou de soif

Si l'agent est secondaire alors
 début
  Si rassasie en eau et pas déjà tourné vers l'est alors
   il va a l'est vers food
   il commente son geste à l'intention des enfants
```

```
  Sinon si rassasie en food et pas déjà tourné vers le sud alors
   il va au sud vers eau
   il commente son geste à l'intention des enfants
 fin

Sinon si l'agent est primaire alors
 début
  Si le niveau d'énergie est dessous le niveau d'alerte, et pas déjà tourné vers l'est
    alors change de direction : fcap est
  Sinon si la réserve d'eau est dessous le niveau d'alerte, et pas déjà tourné au sud
    alors change de direction : fcap sud
 fin

Sinon si rien dans la case devant
 alors avance
 sinon inhibition.
```

Table 57 : La procédure qui détermine la fixation et le changement de cap des agents

4) Concrètement comment programmer la chose ?

La programmation du comportement de chaque agent se fait en initialisant ses champs *BehaviorEst* et *BehaviorSud*. Ensuite, lors de l'exécution de la simulation, ces derniers seront testés par le programme, et ainsi, ils commuteront les différents comportements de l'être.

4.A) L'espèce initiale : des agents primaires inconséquents

Tout d'abord, au début de la première simulation, afin d'obtenir l'espèce initiale, i.e. celle des agents primaires inconséquents et téméraires, nous initialisons ces nouveaux champs ainsi :

```
BehaviorEst= PrimTemeraire; // Programmation d'un comportement primaire.
BehaviorSud= PrimTemeraire;
```

Table 58 : Pour obtenir un comportement primaire, les deux champs sont initialisés à `PrimTemeraire`

4.B) À la naissance, des êtres immatures

Ensuite, au fil de la simulation, les agents en bonne santé deviennent parents. Ils donnent naissance à des enfants qui naissent immatures, i.e. qui ignorent comment tourner. Alors, quand le programme crée ces nouveaux êtres, il initialise leurs champs à `Indetermine` :

```
BehaviorEst= Indetermine; // Son comportement pour aller à l'est n'est pas programmé
BehaviorSud= Indetermine; // Son comportement pour aller au sud n'est pas programmé
```

Table 59 : À la naissance d'un nouvel agent, ses champs sont initialisés à `Indetermine`.

4.C) Une espèce mieux adaptée : les agents secondaires finauds et intelligents

Finalement, nous avons montré que nous pouvons créer des agents mieux adaptés à ce milieu en les faisant tourner plus tôt. Pour obtenir cette espèce améliorée, i.e. ces agents secondaires finauds et intelligents, nous les initialisons ainsi :

```
BehaviorEst= SecoOpportun; // Programmation d'un comportement secondaire.
BehaviorSud= SecoOpportun;
```

Table 60 : Pour obtenir un comportement secondaire, les champs sont initialisés à `SecoOpportun`

4.D) Il existe encore deux espèces hybrides, mais médiocres

Afin de programmer le comportement des agents nous devons initialiser deux champs, avec trois valeurs disponibles. Donc nous calculons que le nombre de types d'agents possibles est $3^2 = 9$. À cette heure, nous en avons déjà décrites trois, parmi celles qui restent, nous choisissons d'analyser et décrire plus précisément deux espèces dissymétriques :

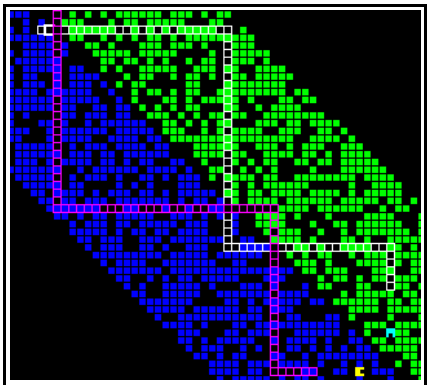
a) L'espèce aquatique *magenta*

Voici l'espèce aquatique *Magenta*, qui vit plutôt dans l'eau parce qu'elle attend d'avoir faim pour marcher à l'est. Pour obtenir des instances de sa classe, nous initialisons les agents à deux valeurs différentes :

```
BehaviorEst= PrimTemeraire;// Si faim Va à l'est
BehaviorSud= SecoOpportun;// Si rassasiée part au sud
```

Table 61 : Initialisation pour obtenir l'espèce Magenta

Ci-contre nous voyons la trace d'un agent *magenta*. D'un côté l'être est téméraire : il fixe son cap à l'est seulement quand il a faim ; mais, de l'autre, il est finaud : il va au sud dès qu'il est rassasié.



309 - Les trajectoires magenta et blanches des deux espèces mixtes

b) L'espèce agricole *blanche*

L'espèce agricole *Blanche* vit plutôt sur terre parce qu'elle attend d'avoir soif pour fixer son cap au sud. Pour l'obtenir nous initialisons les agents ainsi :

```
BehaviorEst= SecoOpportun; // si abreuvée va à l'est
BehaviorSud= PrimTemeraire;// si soif va au sud
```

Table 62 : Initialisation pour obtenir l'espèce Agricole

D'un côté, nous pouvons considérer que l'agent blanc est téméraire car il fixe son cap au sud seulement quand il a soif. Mais, d'un autre côté, il est finaud car il part vers l'est dès qu'il est abreuvé.

c) La solution pour effectuer un traitement cognitif

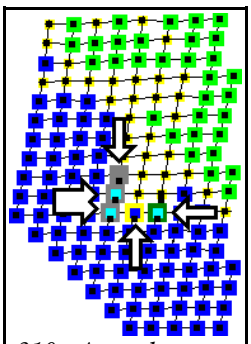
Introduction : rappel de notre démarche initiale vis à vis de la cognition.

Initialement, notre désir est d'exhiber une modélisation qui simulerait un mécanisme de cognition. Dans la partie théorique de notre travail, au chapitre *La cognition chez l'agent rationnel*, nous avons déterminé qu'un traitement cognitif minimal serait de raisonner sur un système, i.e. d'intervenir sur les règles décrivant son monde et d'en tirer les conséquences.

Dans cette simulation *MemeticLamarck*, la démarche consiste en deux étapes. D'abord, au départ, nous devons nous donner un critère d'évaluation de l'efficacité de son fonctionnement. Ensuite, le travail consiste à faire l'aller et retour entre deux traitements : *modification des règles* et *observation du résultat*, dans le but d'améliorer l'efficacité de ce système.

Faire cohabiter ces deux espèces médiocres.

À ce stade nous en sommes à la fin du processus de simulation d'une cognition. L'idée se réduit à simplement faire cohabiter ces deux espèces hybrides au rendement médiocre, afin de provoquer un métissage et d'obtenir un mutant plus adapté.



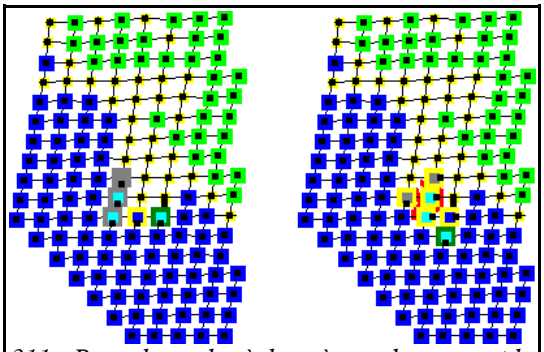
310 - Avant le panachage de règles

Ci-contre, la flèche orientée au sud montre un enfant qui vient de naître. Immature, il est gris avec un nez noir tourné au sud.

Les deux flèches vers l'est montrent deux hybrides tournés au sud. Leur cœur est cyan, mais il faut noter leur bord est gris qui signifie qu'il sont encore en apprentissage

La flèche tournée vers le nord désigne *cœurBleu-BordJaune*, un hybride au cœur bleu et au bord jaune. C'est lui qui, avant tous les autres, va tourner à l'est et transmettre son savoir à ceux qui l'entourent.

La flèche tournée à l'ouest montre un hybride au cœur cyan et au bord vert bouteille.



311 - Panachage de règles : à gauche = avant le transfert ; à droite = après celui-ci.

Le parent n'est pas forcément l'éducateur.

Dans cette simulation, les enfants naissent sans savoir comment tourner, et c'est la communauté qui prend en charge leur éducation, car le parent n'est pas forcément l'éducateur. En effet, avant que le parent ne tourne, un autre adulte peut très bien changer de cap dans le voisinage du nouveau né, et ainsi l'éduquer.

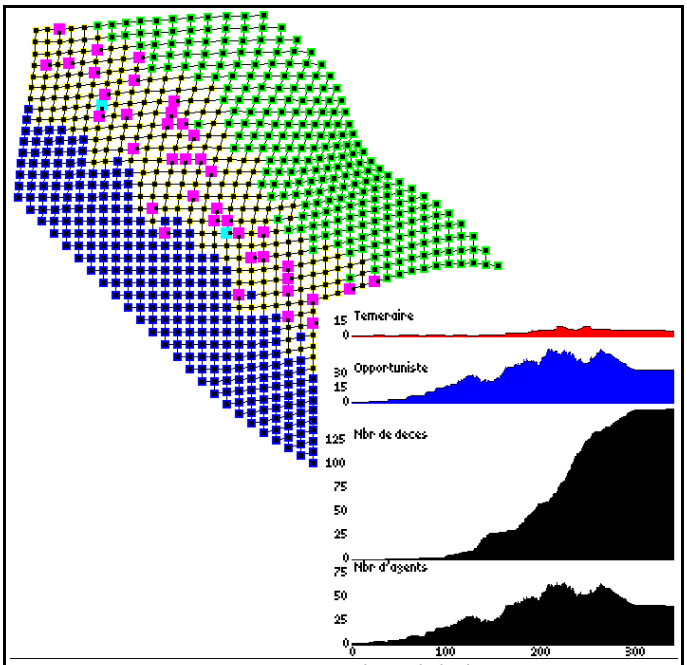
Dans l'illustration ci-dessus à droite, le parent *cœurBleu-BordJaune* tourne en premier. Il fixe son cap à l'est, ainsi il passe son savoir[93] aux trois enfants, dont au moins un n'est pas de sa progéniture.

Cette organisation sociale induit un échange de mèmes, et mélange les patrimoines des agents : des mutations culturelles se produisent.

Le passage de la forme profonde à la forme de surface.

Dans la nature, le passage du génotype au phénotype est effectué par la morphogénèse. Il correspond à une transition de la forme profonde à la forme de surface.

Quel est le pendant de cette étape dans le cadre des A6S[94] sociaux ? En modélisation, le passage de la forme profonde à la forme de surface est effectué par la mise en situation de l'agent dans le monde, car elle produit l'expression de l'agent, i.e. elle génère le comportement de l'être en fonction de l'adversité.



312 - Le résultat global

La pression de sélection.

Dans ce cadre des A6S sociaux, la nécessité de survivre induit une pression de sélection qui favorise la reproduction des êtres les mieux adaptés : ainsi, à terme, en

93 Le transfert est visualisé par un trait rouge.

éliminant les agents les moins viables, la nature élimine les comportements aberrants, et indirectement, elle éradique les mauvaises règles de décision qu'ils portent.

Ci-contre, est présenté l'histogramme d'une session où nous opposons des *primaires/téméraires* contre des *secondaires/opportunistes*. À l'issue de la confrontation, nous trouvons environ quatre fois plus d'agents secondaires (histogramme bleu) que de primaires (histogramme rouge).

La sélection des règles.

En fait, au moyen de cette simulation, en mettant des agent sociaux en situation dans le monde, voici ce qu'indirectement nous faisons :

- En hybridant les règles, nous intervenons sur le comportement d'un système formel, ce qui le fait évoluer.
- Puis nous évaluons la justesse de ce nouveau traitement de l'information, i.e. la justesse de ce nouveau jeu de règles.
- Pour tenir compte de cette évaluation, nous délaissons les mauvaises jeux de règles, et focalisons sur les bons.

d) Conclusion

Finalement, indirectement, en mettant des agent sociaux en situation dans le monde, nous avons effectué un traitement complexe de l'information. Nous avons simulé le fonctionnement de ce que nous avons défini être une cognition facile à implémenter.

94 A6S : Agents Somatiques Situés en Survie Sur un Support Simple.

5) Présenter les résultats

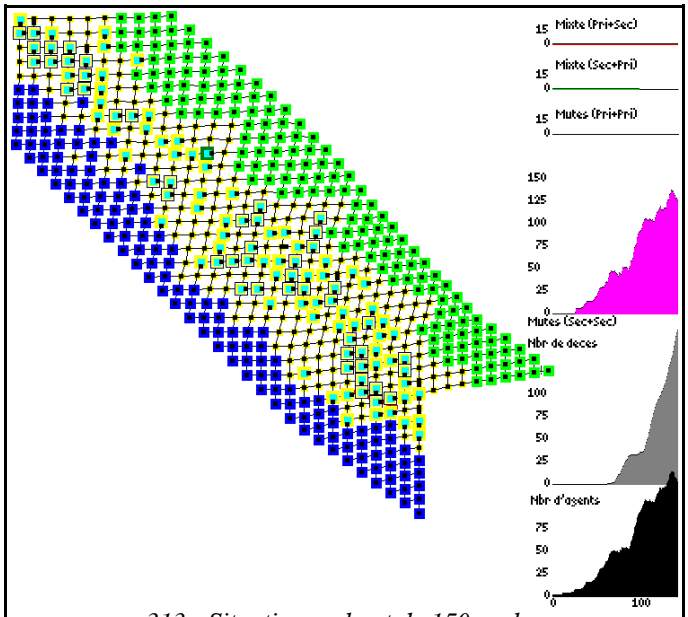
Maintenant nous présentons les résultats obtenus en programmant selon la méthode décrite au paragraphe précédent.

5.A) Où nous voyons la mutation se produire

L'état de départ.

Dans l'exemple ci-contre, nous commençons avec seulement deux agents mixtes : le premier est *primaire-secondaire* et le deuxième *secondaire-primaire*.

Explicitons un peu les histogrammes présentés ci-contre dans la colonne droite : la première ligne magenta, puis la seconde verte montre que le système est initialisé avec ces deux mixtes, et que leur population n'évolue pas. La troisième ligne montre qu'il ne se crée par d'agent mixte *primaire-primaire*.



313 - Situation au bout de 150 cycles

Le résultat obtenu.

Au bout de 150 cycles, nous obtenons une population, où ne vivent pratiquement que des agents hybrides secondaires.

Sur la quatrième ligne de l'histogramme ci-contre nous remarquons bien la croissance de la courbe magenta, qui correspond à leur population (histogramme magenta).

La cinquième ligne affiche le nombre d'agents morts, et la sixième le nombres d'agents vivants.

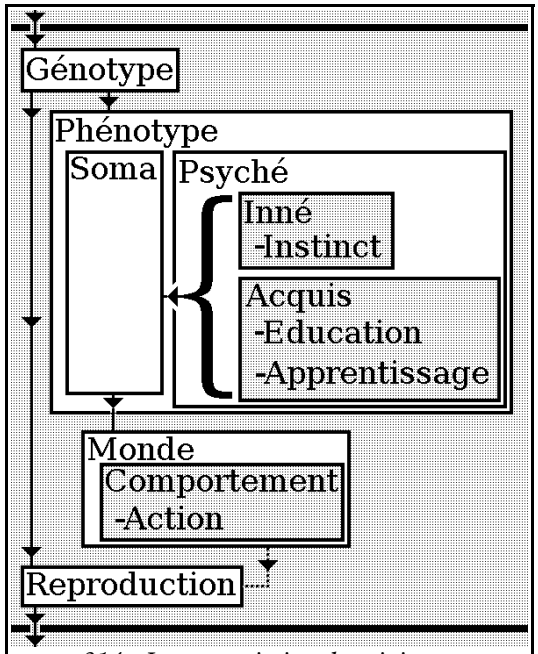
5.B) Discussion annexe : darwinisme ou lamarckisme

Introduction.

En sciences naturelles les deux théories génétiques de Darwin et Lamarck se sont affrontées, jusqu'à ce qu'on reconnaisse que Darwin avait raison.

Cependant, il semble que, si nous regardons au sein de la simulation *MémèticLamarck*, la transmission culturelle de règles de conduite épigénétiques entre deux agents, soit lamarckienne.

La théorie de Darwin [Dar 1859] :



314 - La transmission darwinienne

Selon Charles Darwin, la mutation initiale est transmise si le mutant produit par la morphogénèse se trouve mieux adapté à son milieu.

Si une mutation se produit lors de la duplication du patrimoine génétique, elle est transmise à l'enfant. Ce dernier est un mutant car la morphogénèse l'a construit à partir d'un patrimoine muté. Si cette mutation le rend plus adapté à son milieu, elle augmente ses chances pour survivre et se reproduire. Dans ce cas l'enfant, transmet le patrimoine muté, qu'il a reçu initialement de ses parents.

Le schéma ci-contre illustre la transmission darwinienne : le génotype, interprété par la morphogénèse produit un phénotype, i.e. le corps somatique et la psyché. La décision de l'action est prise dans la psyché, en fonction des règles innées et acquises, i.e. en fonction de l'instinct, de l'éducation et de l'apprentissage. À chaque boucle *perception décision action*, l'action décidée est transmise au corps qui agit dans le monde. La séquence des actions ainsi exécutée produit le comportement qui est implicitement testé en situation. Si cette évaluation est positive, la reproduction est possible. Elle se fait en retransmettant le génotype, i.e. le caractère inné de l'être.

Ainsi, la mutation qui va dans le sens d'une meilleure adaptation du mutant à son milieu est sélectionnée. Sinon, elle est abandonnée.

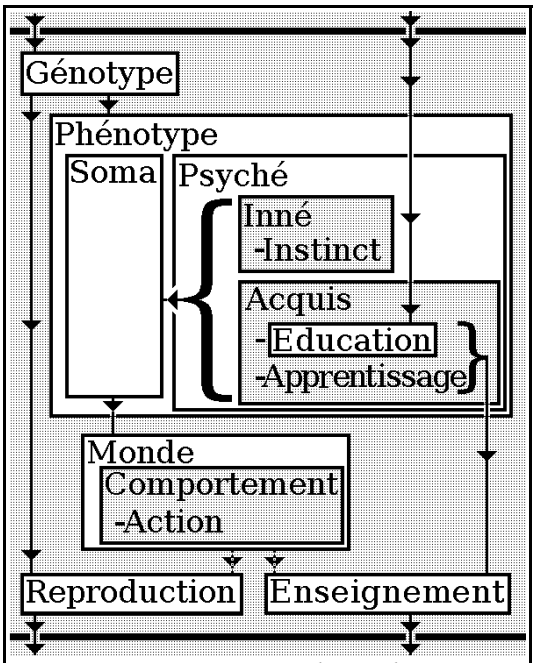
La théorie de Lamarck :

Selon Jean-Baptiste de Lamarck, le caractère acquis du vivant de l'être est transmis à sa descendance.

Un être vit dans un milieu très spécifique. Si, durant sa vie, il acquiert un caractère qui le rend plus adapté à son milieu, alors il transmet ce caractère à sa descendance.

Le synoptique ci-contre reprend le schéma précédant, mais il a été modifié pour modéliser la transmission lamarckienne de la culture acquise.

D'un côté, nous retrouvons exactement le même mécanisme de transmission physique du génotype. De l'autre apparaît la transmission culturelle de règles comportementales épigénétiques : les règles qui déterminent la prise de décision, sont acquises par apprentissage et par éducation. Si elles pilotent bien les actions, i.e. si le comportement de l'agent est adapté, celui-ci se reproduit, mais il transmet aussi son savoir.



315 - La transmission lamarckienne

Dans ce cas, nous remarquons qu'il enseigne aux autres ce qu'il a compris par apprentissage et ce que les autres lui ont enseigné. Manifestement, il transmet les caractères comportementaux qu'il a acquis durant sa vie.

6) Conclusion : peu et beaucoup

Après avoir donné une définition de la cognition, nous avons simulé un monde où nous avons pu observer un exemple de traitement cognitif. C'est à la fois peu et beaucoup.

C'est peu car il s'agit d'une observation partielle, limitée à un exemple que nous créons de toutes pièce afin de pouvoir l'exhiber

C'est beaucoup car :

- Ce résultat est obtenu simplement, donc il est facilement observable et dissécable.
- Nous livrons au chercheur une modélisation de l'agent situé, légère et analysable.

- Nous attirons l'attention des scientifiques sur une zone privilégiée : en copiant la nature, nous utilisons la décentralisation, le groupe et sa communication, alors la construction d'agents intelligents devient possible, et permet d'éviter l'obstacle de la combinatoire.

Ch. 42 Conclusion

1) Agent situé : léger et simple pour représenter et traiter les données

Dans l'histoire de l'informatique, après les langages macroscopiques et les langages objets, viennent les agents, puis surtout les agents situés, qui fournissent une représentation et un traitement des connaissances plus simples et plus légers.

1.A) Langages macroscopiques : représentation centralisée des données/procès

En programmation macroscopique classique, qu'elle soit structurée ou non, la représentation des données et de leur traitement (procès) est centralisée.

a) À la base en informatique, la notion de procès existe déjà

Par définition, le procès est un traitement qui porte sur des données. Il est à la base de l'informatique, qui, par définition est la science du traitement automatique de l'information.

b) Regardons l'appel de programme selon que le programme est structuré ou non

L'appel de programme c'est son lancement, son ordre d'exécution.

En programmation non-structurée : le lancement d'un programme.

En informatique classique, la représentation du code et des données est centralisée au sein d'un unique programme, mais toutes ces informations sont visibles en quelque lieu du programme. Dans ce cadre le lancement d'un programme se réduit à l'ordre d'exécuter un morceau de code dont nous énonçons le nom.

En programmation structurée : le lancement d'un programme.

En programmation structurée, les données sont locales i.e. visibles seulement en contexte : , toutes ces informations sont visibles seulement par le sous-programme qu'on exécuté. Dans ce cadre le lancement d'un programme est constitué de deux choses : la donnée de son nom et la fourniture de ses paramètres

c) Transition : un programme décrit les procès à effectuer, mais d'où provient-il ?

Dans toute l'informatique classique, le programme est parachuté : il est fourni par le programmeur qui l'élabore de A à Z. Mais nous allons voir la programmation événementielle des langages objets et surtout celle, déclarative, des agents situés, qui allègent l'écriture du code.

1.B) Les langages objets : décentralisation, héritage mais abstraction

Grâce à des principes de base tout à fait nouveaux les langages objets constituent une évolution importante dans l'art de la programmation.

La programmation classique est monolitique : elle concentre le code et les données dans le même programme. La programmation objet amène une nouvelle représentation décentralisée du code et des données : chaque objet voit midi à sa porte : par héritage, il reçoit une structure de données dédiée et le programme correspondant pour les traiter.

Tous les objets sont différents, ils ne sont donc pas activés en même temps, mais plutôt en fonction des évènements. On débouche sur la notion de programmation événementielle.

Transition ; cependant, il reste que chaque objet demeure encore abstrait, détaché de la réalité concrète, ce qui nous amène aux agents.

1.C) Introduction de l'action : l'agent est plongé dans le monde où il agit

Maintenant nous passons à une étape clé, nous transformons l'objet en agent en simulant une plongée dans la matière, la réalité.

a) L'agent continue d'effectuer un procès

D'abord chez l'agent, nous retrouvons la notion de procès qui était présente dès le début de l'informatique : lui aussi il effectue un procès.

Mais maintenant le procès est incarné.

La transition objet→agent s'effectue au moyen d'une concrétisation. Elle correspond donc à une plongée dans la matière. Dans ce cadre, le traitement informatique devient et se réduit à une simulation d'actions dans un système maintenant instancié qui représente le monde.

En conclusion, cette plongée dans la matière induit des conséquences.

D'un côté, le système informatique initialement composé d'objets abstrait devient un groupe d'agents dans un monde. Ainsi le procès à exécuter, qui était générique, devient typé et contraint par les lois physiques. Le traitement informatique, qui consistait à traiter des données au moyen de programmes, devient simuler l'action des agents dans le monde. Ceci induit une simplification du traitement, et facilite la simulation de l'évolution de l'état du système, à partir du moment où nous savons son initialisation, son état de départ.

De l'autre, nous sommes en présence d'agents dans un monde concret, où bien des lois physiques peuvent exister. Nous ne sommes pas encore dans un monde d'agent situés où les actions se déroulent dans l'espace, et où les procès sont encore plus contraints, i.e. encore plus facile à simuler.

b) Lemme : nous distinguons deux structures d'agent

Avant de continuer, introduisons un lemme. Nous distinguons deux structures d'agents selon leur positionnement en rapport au temps, i.e. à la mémoire : nous nous demandons s'ils sont hystérétiques ou non.

D'abord nous décrivons l'agent réactif, qui n'est pas inscrit dans le temps, il est instantané, il agit dans l'instant.

Ensuite nous introduisons l'agent hystérétique, qui est inscrit dans le temps, il mémorise des faits à propos du monde ou des projets, et interprète ensuite cette information pour générer ses actions, i.e. son comportement.

c) Introduction de deux procès plus ou moins finalistes

Tout d'abord, nous introduisons la notion de procès finaliste

Il s'agit d'un traitement finaliste i.e. sous-tendu par un but.

Maintenant, nous présentons deux procès plus ou moins finalistes :

D'abord nous décrivons l'agent réactif, un agent indirectement finaliste. Il n'a pas directement généré le comportement qu'il exécute afin d'atteindre un but, mais son concepteur l'a doté d'un comportement adapté à cette fin.

Ensuite, nous décrivons l'agent rationnel, un agent directement finaliste. Au moyen d'un traitement adapté, il génère directement le comportement qu'il exécute afin d'atteindre son but.

1.D) Introduction de l'agent situé

Le fait de construire progressivement l'agent situé induit plusieurs conséquences.

a) Construire un agent situé

Nous construisons un agent, nous l'incarnons dans le monde, mais au lieu d'introduire dans notre modèle toutes les grandeurs physiques, nous nous restreignons aux grandeurs de l'espace, nous choisissons de le placer sur un support.

b) Introduire progressivement l'agent situé induit la reproduction cellulaire

Nous commençons seulement par les grandeurs spatiales. Ainsi nous devons commencer par le point. Mais dans ce monde ponctuel, nous devons avoir reproduction. Ceci nous amène à tout simplifier, au point qu'on arrive à la reproduction de la cellule de base.

c) Simulation l'évolution des espèces avec notre modèle

Simuler l'évolution des espèces dans notre modèle induit deux conséquences :

- Naturellement, la classification de la descendance de la cellule s'organise spontanément selon une structure arborescente, et constitue ainsi spontanément une taxinomie des simulations.
- Selon un point de vue informatique, simuler la reproduction des agents consiste à implémenter une fonctionnalité d'héritage parent→enfant.

d) Conclusion

Le fait de construire progressivement l'agent situé nous amène à placer à la base de notre arbre, la cellule végétative et son cycle élémentaire de reproduction. Ainsi, paradoxalement, quand il commence à grandir, notre arbre est guidé par un déterminisme bien visible : au début, il se construit sans embranchement, comme une branche principale. Ainsi il présente la forme d'un tronc.

2) Dans le cadre des agents situés, nous reconstruisons notre plate-forme

2.A) Au début, nous commençons par le cycle de la cellule végétative

À la base de ce travail de reconstruction, nous mettons la cellule végétative. Son cycle est simulé par un séquenceur constitué d'un unique état. En mathématique ce cycle peut très bien être simulé par un système formel.

2.B) Nous décomposons notre classification arborescente en trois parties

a) Partie 1 AF→ASM⊥

Nous avons déjà dit qu'au début, la reconstruction de l'agent situé se fait progressivement, nous introduisons l'espace étape par étape. Elle est tellement déterministe que nous fabriquons l' arbre en couches concentriques : dans cette zone il présente la forme d'un tronc.

b) Partie 2 ASM⊥→AIntention

La partie 2 correspond à la transition entre l'ASM⊥, qui est à peine causal et l'*agent Intension.* À ce niveau apparaissent les structure de fourche. à mesure que l'introduction du temps i.e. de la mémorisation nous permet de construire l'*agent intentionnel* nous pouvons réaliser des simulations plus conséquentes qui partent vers différentes directions : communication (langage et apprentissage) et le traitement rationnel.

c) Partie 3 Attention→A.BDI et ARC (agents rationnels cognitifs)

L'agent CDI rationnel, plus complexe plus puissant, permet de prolonger les simulations dans les directions précédemment initiées.

2.C) Quel est notre but ?

a) De façon statique (sans phénomène d'émergence des agents)

Nous cherchons à exhiber ponctuellement des agents qui effectuent un traitement rationnel cognitif de l'information, et aussi, des simulations d'agents situés pour qu'elles servent aux chercheurs des sciences humaines et sociales.

b) De façon dynamique (avec phénomène d'émergence des agents)

Nous cherchons à visualiser des phénomènes d'émergence en classant les simulations dans une taxinomie.

Chaque simulation trouvée et exhibée est construite autour d'agents qui proviennent de l' évolution de leur ancêtre déjà classé dans notre taxinomie. Chaque agent et simulation hérite de ses parents, alors, quand nous classons nos agents nous obtenons une structure d'arbre.

Ainsi, quand nous regardons globalement notre arbre, classement nous obtenons une taxinomie qui montre bien la morphogénèse de nos êtres.

3) Nous proposons un système, i.e. une taxinomie de simulations

3.A) Les propriétés de ce système de simulations d'agents situés sur un support

a) C'est un système ouvert

Nous obtenons un système ouvert. Il est organisé en arbre, et nous pouvons y rajouter des simulations.

La chose peut facilement se faire en faisant évoluer les agents, et dans ce cas la simulation se placera en position de feuille dans notre arbre.

Il est aussi envisageable de raffiner la progression de l'évolution en introduisant des étapes supplémentaire sur une branche, entre deux simulations.

Nous pouvons aussi rajouter des capteurs ou actionneurs qui agiraient sur le monde au moyen de nouvelles propriétés physiques. Dans ce cas, sur les taxinomie, elles apparaîtraient comme des branches nouvelles ajoutées à un nœud.

b) C'est un système simple car il accumule les simplifications

Les langages objets.

Ils introduisent de la simplicité dans le système car ils induisent la décentralisation du code et des données.

Les techniques d'émergence.

À la base de notre taxinomie, se trouve la cellule végétative. Avec elle, la vie est déjà là. Initialement, elle fonctionne de suite selon un cycle où elle se reproduit. Alors, nous partons de ce système et cherchons seulement à l'améliorer.

L'effet holistique du groupe.

La plupart de nos simulations mettre les agents en interaction dans le groupe, ainsi apparaissent des phénomènes holistiques qui nous intéressent car souvent ils modélisent ceux de la vie quotidienne.

Effet d'horizon.

Nos simulations mettre en scène des agents dont la portée des perceptions et actions est limitée par l'effet d'horizon à un voisinage proche. Ainsi, à part pour la communication, il n'est en relation qu'avec ces voisins. Cet effet simplifie bien ses relations avec le monde (support + mobilier).

c) C'est un système empirique et non déterministe

Il est empirique pour 3 raisons.

Nous avons déjà décrit l'effet d'horizon qui limite ses relations avec le monde sur deux niveaux : sa perception et ses actions. De plus son processus de décision n'est pas systématique mais se fait selon une stratégie empirique selon un parcours en largeur d'abord pour la construction des règles. Pour ces trois raisons, il n'est pas déterministe.

Introduction de l'empirisme.

Dans nos modèles, nous introduisons l'empirisme, ce qui induit une simplification des règles d'action des agents situés : Si nous formulons en pseudo langage les règles d'action de l'agent situé, elle contiennent des déictiques. Ex : *quand je ne perçois rien dans la case devant moi, j'avance d'un pas !*

3.B) Conséquence conclusion

a) Cette taxinomie a été construite de façon cohérente

Nous l'avons construite en aidant l'évolution : les agents de chaque nouvelle simulation sont les enfants des agents de la simulation précédente, que nous avons un améliorés.

Cette construction qui aide l'évolution nous fournit une fabrication par héritage des agents. Cela suffit pour donner une structure cohérente à notre arbre des simulations.

b) Nous obtenons des agents qui possèdent une structure simple et épurée

Ce n'est parce que les agents sont simples qu'il faut les délaisser ou les négliger. Bien au contraire, Avec des agents qui sont très simples et épurés nous pouvons simuler des fonctionnement qui sont observables. Ainsi les manips que nous proposons peuvent être conséquentes tout en restant accessibles.

c) Nous pointons, découvrons une zone privilégiée d'expérimentation

Quelle est la spécificité de cette zone ? C'est la zone où les simulations sont suffisamment simples pour éviter de tomber sous les coups de la combinatoire, et ne sont pas triviales afin d'effectuer des traitements qui cependant sont conséquent,

d) Ce domaine est ouvert : il demande à être prolongé

C'est un domaine vierge, totalement à défricher, il y est facile d'écrire des applications intéressantes : il suffit de s'y pencher pour les ramasser.

3.C) Chacun des systèmes présent dans notre taxinomie sert à :

a) Les simulations de notre taxinomie servent à modéliser

Elles servent à modéliser divers phénomènes humains qui relèvent de différents domaines :

- Des sciences sociales. (la communication, le langage).
- De la psychologie, ou des sciences cognitives : cognition (apprentissage, Traitement rationnel cognitif).

b) Les simulations de la taxinomie servent à appréhender différentes sémantiques

Dans nos simulations, nous touchons du doigt plusieurs sémantiques.

Sémantique dénotationnelle : dénoter une scène.

En situation sur le support les agents ont une perception limitée par un effet d'horizon. Ainsi il perçoivent des scènes différentes qu'ils vont traiter de la même façon. Par exemple, pour du mobilier sur le support, un groupe de cases peut constituer un obstacle. Ceci concourt à modéliser des phénomènes de sémantique dénotationnelle (à la Frege-Russel) : le même symbole *obstacle* peut dénoter des scènes différentes.

Sémantique de la fonction : caractériser un fonctionnement.

Au sein des simulations, à plusieurs niveaux de granularité (composant agent groupe), nous identifions différentes fonctionnalités dont nous avons besoin de caractériser le fonctionnement. Pour cela nous essayons d'utiliser les outils classiques, la donnée de la table de vérité en logique et celle de la fonction de transfert en numérique.

Sémantique de l'efficacité : évaluer l'efficacité d'un fonctionnement.

Mais il arrive que nous sommes en présence d'une fonctionnalité complexe, qui est constituée par un groupe. Par exemple, pour du mobilier sur le support, le groupe peut être un obstacle, pour des agents sur le support, le groupe peut être un essaim. Dans ces deux cas, il n'est pas

toujours possible de caractériser leur fonctionnement en donnant leur fonction de transfert. Alors nous les caractérisons plus grossièrement en les évaluant, par la donnée d'un nombre : leur valeur calculée au moyen d'un lissage.

c) Une plate-forme pédagogique pour enseigner l'IA et les agents

Ce plate-forme constitue un outil pédagogique pour enseigner l'IA et aussi la programmation des agents.

Notre plate-forme est livrée avec plusieurs applications dans le domaine des agents situés . Ainsi, elle constitue un bon outil pédagogique pour enseigner l'IA et en particulier les SMA. Ses simulations permettent de recréer différentes situations ou scènes qui montrent bien les rouages d'un mécanisme épuré qui est visible. Il peut alors être étudié et disséqué en cours.

De plus cette plate-forme est livrée avec le code source en C, qui peut servir à enseigner les rudiments de la programmation des agents.

4) Conclusion finale

Outil pour modéliser l'informatique et les sciences humaines et sociales.

Finalement, nous disposons d'un outil qui, d'un côté peut servir à la recherche en sciences humaines et sociales, mais aussi en informatique et en sciences cognitives.

Perspectives : avoir les bonnes idées et prolonger ce système qui est ouvert.

Notre démarche de recherche est ouverte, nous nous positionnons selon un investissement à long terme et souhaitons qu'il soit prolongé par les générations suivantes. Nous sommes en présence d'un système qui est ouvert. Alors bien sûr, il faut le construire et le prolonger.

D'un côté nous sommes contraints par les règles physiques, les paramètres et contraintes physiques du système, qui sont incontournables et avec lesquelles nous devons nous débrouiller.

De l'autre, nous devons trouver, imaginer des manips spectaculaires et intéressantes, et tout de même nous appuyer sur des grandeurs physiques choisies, afin de simuler des applications dans la vie quotidienne.

Ch. 43 Bibliographie

[AAH 2009] Patrick Albert, Armetta Frédéric, Hassas Salima,
Agents situés : une nouvelle voie pour le développement d'applications industrielles ,
Hermès-Lavoisier, 2009.

[AHD 1981] Robert Axelrod, Hamilton, D. William,
The Evolution of Cooperation,
Science 211: 1390–96, doi:10.1126 /science.7466396,
http://www.sciencemag.org/cgi/reprint/211/4489/1390.pdf, 27 March 1981.

[Axe 1976] Robert Axelrod,
Structure of Decision: The Cognitive Map of Political Elites,
Princeton University Press, 1976.

[Axe 1984] Robert Axelrod,
The Evolution of Cooperation,
Basic Books, 1984.

[Axe 1997]Robert Axelrod,
The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration,
Princeton University Press, 1997.

[Bar 1960] Yehoshua Bar-Hillel,
A demonstration of the nonfeasibility of fully automatic high quality translation,
appendix III of *The present status of automatic translation of languages*, Advances in Computers,
vol.1 p.158-163, 1960.

[Bar 1964] Yehoshua Bar-Hillel,
A Demonstration of the Nonfeasibility of Fully Automatic High Quality Translation,
Language and information p.174-179, Reading, Mass., Addison-Wesley, 1964.

[Ben 1966] Émile Benveniste,
Problèmes de linguistique générale - 1,
Paris, Gallimard, 1966.

[Ben 1974] Émile Benveniste,
Problèmes de linguistique générale - 2,
Paris, Gallimard, 1974.

[Ber 2000] Gilles BERNARD,
Principes et mécanismes d'économie des représentations dans la modélisation de systèmes linguistiques,
Synthèse pour l'habilitation à diriger des recherches, 2000.

[BG 1988] Byron & Gottfried,
Programmation en C,
Mac GRAW HILL Série Schaum, 1988.

[BH 2003] Bruno Bouzy and Bernard Helmstetter.
Monte carlo go developments,.
In Kluwer, editor, Advances in Computer Games (ACG10), pages 159–174, 2003.

[BF 1969] D. Bobrow & B Fraser,
An augmented state transition network analysis procedure,
Proceedings of IJCAI, 1969

[Boo 1854] George Boole,
Investigation of the laws of thought on which are founded the mathematical theories of logic and probabilities
Macmillan, 1854. Réimprimé avec des corrections, Dover Publications, New York, 1958.

[Bra 1984] Valentino Braitenberg,
Vehicle Experiments in Synthetic Psychology,
Mit Press, 1984.

[Car 2004] Kathleen M. Carley,
Artificial Social Agents,
Carnegie Mellon University, 2004.

[Car 1886] Lewis Carroll (Charles Lutwidge Dodgson),
Logique sans peine,
Éd. Hermann, Paris, 1966.

[Car 1896] Lewis Carroll (Charles Lutwidge Dodgson),
La logique symbolique
Symbolic Logic, Part I.

[Car 1959] John McCarthy,
Programs with common sense,
Teddington Conference on the Mechanization of Thought Processes, proceedings,
http://www-formal.stanford.edu/jmc/mcc59.html, 1959.

[Cas 2006] L. N. de Castro,
Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications,
CRC Press, 2006.

[CBJ 2010] Alain Cardon, J-Paul Baquiast, Christophe Jacquemin,
Panorama actuel de l'Intelligence Artificielle,
http://www.admiroutes.asso.fr/larevue/2001/21/ia.htm, 2010

[CH 1969] J. McCarthy and P. J. Hayes,
Some philosophical problems from the standpoint of artificial intelligence,
Machine Intelligence, 4:463-502, 1969.

[Cha 1983] Jean-Pierre Changeux,
L'homme neuronal,
Pluriel Fayard, 1983.

[Cho 1956] Noam Chomsky,
Three models for the description of language,
IRE Transactions on Information Theory 2: 113–124, 1956.

[Cho 1969] Noam Chomsky,
Syntactic structures,
Mouton The Hague, 1969.

[Cot 2007] Pierre-Alain Cotnoir,
L'assimilateur mémétique,
http://www.vigile.net/_Cotnoir-Pierre-Alain_, 2007.

[CQ 1969] Collins, A. M., & Quillian, M. R. (1969).
Retrieval Time from Semantic Memory.
Journal of Verbal Learning and Verbal Behavior, 8, 240-247, 1969.

[Dar 1859] Charles Darwin,
The origin of species by means of natural selection, or the preservation of favoured races in the struggle for life,
1859.

[Daw 1987] Richard Dawkins,
Gènes égoïstes et mêmes égoïstes,
in Vues de l'esprit Interéditions, 1987.

[Den 1981] Daniel Dennett,
True believers : the intentional strategy and why it works,
in Scientific explanation, A.HEATH, Oxford U.P., 1981.

[Den 1987] Daniel Dennett,
Three kinds of intentional psychology,
in the intentional stance MIT, 1987.

[Des 1990] Jean-Pierre Desclés,
Langages applicatifs, langues naturelles et cognition,
Hermès, 1990.

[Dre 1972] Hubert Dreyfus,
What computers can't do, The limits of artificial intelligence,
Intelligence artificielle, mythes et limites, 1972, 1979 et 1992.

[Fer 1983] Jacques Ferber,
MERING, un langage d'acteur pour la représentation et la manipulation des connaissances.
Thèse de docteur-ingénieur, Université Paris 6, décembre 1983.

[Fer 1989] Jacques Ferber,
Objets et agents : une étude des structures de représentations et de communications en intelligence artificielle,
Thèse de doctorat d'état, Université Paris 6, juin 1989.

[Fer 1995] Jacques Ferber,
Les systèmes multi-agents. Vers une intelligence collective,
InterEditions, 1995.

[Fod 1975] Jerry Fodor,
The language of thought,
Harvard The language and thought series, 1975

[Fod 1985] Jerry Fodor,
Fodor's guide to mental representation : the intelligent Auntie vade-mecum,
Mind XCIV, 76-100, 1985.

[Fod 1985] Jerry Fodor,
Something on the state of the art, in Representation,
MIT press, 1981.

[Fod 1986] Jerry Fodor, *La modularité de l'esprit*.
Les édition de minuit, Propositions, 1986

[Fod 1987] Jerry Fodor,
Psychosemantics, the problem of meaning in the philosophy of mind.
MIT press, 1987.

[Fre 1923] Gottlob Frege 1923,
Gedankengefüge (la pensée mixte),
in Beiträge zur Philosophie des Deutschen Idealismus III: 36–51,
English translations in *Logical Investigations*, ed. Peter Geach, Blackwell's, 1975.

[Gan 1990] Jean-Gabriel Ganascia,
L'âme machine,
Science ouverte Seuil, 1990.

[Gan 2006] Jean-Gabriel Ganascia,
Les sciences cognitives,
Le Pommier, 2006.

[Gil 1962] A. Gill,
Introduction to the Theory of Finite-state Machines,
McGraw-Hill, 1962.

[GLW+ 2008] Tao Gong, Andrea Puglisi, Vittorio Loreto, William S.-Y. Wang,
Conventionalization of Linguistic Knowledge Under Communicative Constraints,
Biological Theory 3(2), pp 154–163, 2008.

[GM 2008] Agnès Guillot et Jean-Arcady Meyer,
La bionique. Quand la science imite la nature,
Dunod, 2008.

[Gou 1958] Georges Gougenheim,
Dictionnaire fondamental de la langue française,
Librairie Marcel Didier, Paris, 1958.

[Gra 1995] Léon-Louis Grateloup,
Cours de philosophie,
Hachette éditions, 1995.

[Gue 2003] Alejandro Guerra Hernandez,
Apprentissage d'agents rationnels BDI dans un univers Multi-Agents,
Thèse de doctorat, Université Paris 13 - institut Galilée, 2003.

[Hau 1985] John Haugeland,
Artificial Intelligence: The Very Idea.
Cambridge, Mass.: MIT Press, 1985.

[Hau 1989] John Haugeland,
L'Esprit dans la machine, Fondements de l'intelligence artificielle, Odile Jacob, 1989.

[Heb 1949] Donald Olding Hebb,
The Organization of Behavior : A Neuropsychological Theory,
Wiley, New York, 1949.

[Hel 2006] Bernard Helmstetter,
Analyses de dépendances et méthodes de Monte-Carlo dans les jeux de réflexion,
Thèse de doctorat, Universite Paris 8 – Vincennes-Saint-Denis, 2006.

[Hin 1989] Jaakko Intikka,
L'intentionnalité et les mondes possibles,
Presses universitaires de Lille, 1989.

[Hof 1985] Douglas Hofstadter.
Gödel, Escher, Bach. Les brins d'une guirlande éternelle.
InterEditions, 1985.

[Hop 1982] John Joseph Hopfield,
Neural networks and physical systems with emergent collective computational abilities,

Proceedings of the National Academy of Sciences of the USA, vol. 79 no. 8 pp. 2554-2558, April 1982.

[Jac 1992] Pierre Jacob,
Le problème du rapport du corps et de l'esprit aujourd'hui,
Introduction aux sciences cognitives essai Folio, 1992.

[Jak 1981] Roman Ossipovitch Jakobson,
Éléments de linguistique générale (1 et 2),
Éditions de Minuit, 1981.

[JAS 2010]
Journal of Artificial Societies and Social Simulation,
http://jasss.soc.surrey.ac.uk/JASSS.html.

[Joh 1983] P.N. Johnson-Laird,
Mental Models,
HARVARD U.P., 1983

[Jou 2007] Nicolas Journet,
Les espoirs déçus de la mémétique,
Sciences Humaines, N°183, 2007.

[Jou 2005] Pascal Jouxtel,
Comment les systèmes pondent,
Editions Le Pommier, 2005.

[Jou 2010]
Le blog de Pascal Jouxtel,
http://systempondent.over-blog.com/5-index.htm, 2010.

[Kae 2005] Leslie Pack Kaelbling,
A situated-automata approach to the design of embedded agents,
Teleos Research, 2005.

[Kam 1983] Hans Kamp,
SID without time or questions,
Ms. Stanford, CA, 1983.

[Kam 1983] Hans Kamp,
Unpublished discourse representation theory project description,
University of Texas Austin, 83

[Kay 1992] Daniel Kayser,
Profondeur variable et sciences cognitives,
Introduction aux sciences cognitives essai Folio, 1992

[Koh 1982] Tuvo Kohonen,
Self-organized formation of topologically correct feature maps,
Biological Cybernetics, 43:59-69, 1982.

[Koh 1984] Tuvo Kohonen,
Self-Organization and Associative Memory,
Springer, Berlin, 1984.

[Koh 2001] Tuvo Kohonen,
Self-Organizing Maps. Third, extended edition. Springer, 2001.

[KR 1988] Kernighan & Ritchie,
Le langage C,
Masson, 1988.

[Lav 2000] Frédéric Laville,
La cognition située : une nouvelle approche de la rationalité limitée,
Revue économique, Vol. 51, No. 6, pp. 1301-1331, Sciences Po University Press, Nov., 2000.

[Lan 1984] C. G. Langton,
Self-reproduction in cellular automata,
Physica D 10: 135–144, 1984.

[Leb 1991] Gérard Leblanc,
Turbo/Borland C++ (troisième édition),
Eyrolles, 1991.

[Len 1983] Douglas B. Lenat,
A program that learns new heuristics and domains concepts,
Artificial Intelligence v21,1983.

[Luc 2004] J-François Lucas,
Mémoire de DEA Intelligence Artificielle&Optimisation Combinatoire,
Paris8&Paris13, 2004

[Luc 2010] J-François Lucas,
Agent's evolution to communication, a global approach to language,
CCI Tbilissi, 2009.

[Mar 2006] J-J. Mariage,
Holistic self reprogramming of neural networks: between self-organization and self-observation.
In Vladimir Golovko, Robert Hiromoto, Akira Imada and Kurosh Madani Eds.,
Special Issue of ISJC, the International Scientific Journal of Computing with the 14 best papers of the ICNNAI'06 conference, vol. 5, n° 3, 54-67, 2006.

[Mar 1982] David Marr,
Vision: a computational investigation into the human representation and processing of visual information,
New York : Freeman, 1982.

[MC 2010] Jean Mehat, Tristan Cazenave,
Ary, a general game playing program,
XIIIe Colloque Board Game Studies, Paris, avril 2010.

[Min 1967] Marvin Minsky,
Computation: Finite and Infinite Machines (1st ed.),
New Jersey : Prentice-Hall, 1967.

[Mor 1847] Augustus De Morgan,
Formal Logic or The Calculus of Inference,
London : Taylor & Walton, 1847.

[MP 1972] Marvin Minsky & Seymour Papert, *Perceptrons:*
An Introduction to Computational Geometry,
The MIT Press, Cambridge MA, 1972.

[Neu 1996] John von Neumann,
Le cerveau et l'ordinateur,
Flammarion, collection Champs, 1996.

[New 1990] Allen Newell,
Unified Theories of Cognition,
Harvard University Press, Cambridge, Massachusetts, 1990.

[OZ 2005] S. Olarius, A. Y. Zomaya,
Handbook of Bioinspired Algorithms and Applications,
Chapman & Hall/CRC, 2005.

[Pap 2009] Anna Pappa,
Constructing Lexicon with Morpho-Syntactic Features from Untagged Corpora, 3rd WSEAS International Conference on Computational Intelligence (CI'09), pp. 221-226, 2009.

[Pap 1981] Seymour Papert,
Le jaillissement de l'esprit - Ordinateurs et apprentissage, Logo : un micro-monde,
Flammarion - Collection Champs, 1981.

[Pia 1964] Jean Piaget,
Six études de psychologie,
Médiations, 1964.

[Pis 1980] N. Piskounov,
Méthodes des différences finies,
Calcul différentiel et intégral t-2, pp. 419-421, Mir Moscou, 1980

[Pit 1990] Jacques Pitrat,
Metaconnaissances, futur de l'intelligence artificielle,
HERMES, 1990.

[Pit 1993] Jacques Pitrat,
Penser autrement l'informatique,
Collection perspectives, HERMES, 1993.

[Pit 1995] Jacques Pitrat,
De la machine à l'intelligence,
HERMES, 1995.

[PLP 2002] J. Poudade, L. Landwerlin, P. Paroubek,
Des Agents Cognitifs Situes Apprennent à designer des Actions,
pdf, LIMSI-CNRS, Université de Paris-Sud Orsay, 2002.

[Plo 1970] G. Plotkin,
A note on inductive generalisation,
In machine intelligence 5, pp. 153-163, 1970

[Plo 1971] G. Plotkin,
A further note on inductive generalisation,
in machine intelligence 6, pp. 101-124, 1971

[Pro 1987], Joëlle Proust,
L'intelligence artificielle comme philosophie,
Le débat N° 47 Nov-Dec 1987 Gallimard, 1987.

[Pro 1990], Joëlle Proust,
De la difficulté d'être naturaliste en matière d'intentionnalité,
Revue de synthèse IV S. Nos 1-2, janvier-juin, 1990.

[Put 1992] Hilary Putnam,
La nature des états mentaux,

Les Études philosophiques p.323-335, juillet/septembre 1992.

[Pyn 1986] Joël Pynte,
La notion de signification dans les sciences cognitives,
Psychologie, intelligence artificielle et automatisme. Psychologie et sciences humaines,
Pierre Mardaga, éditeur, 1986.

[RDG+ 1986] David E. Rumelhart, Geoffrey E. Hinton, and R. J. Williams.
Learning Internal Representations by Error Propagation.
David E. Rumelhart, James L. McClelland, and the PDP research group. (editors),
Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1:
Foundations. MIT Press, 1986.

[Ric 1987] Elaine Rich,
Intelligence artificielle,
Mason, 1987.

[Ric 1984] Chantal et Patrice Richard,
Programmatique, initiation à la programmation méthodique,
Edition Belin, 1984.

[RK 1986] Stanley J. Rosenschein and Leslie Pack Kaelbling,
The synthesis of digital machines with provable epistemic properties,
Proceedings of the Conference on Theoretical Aspects of Reasoning About Knowledge,
pages 83-98. Morgan Kaufmann, Joseph Halpern, 1986.

[RK 1987] Stanley J. Rosenschein and Leslie Pack Kaelbling,
The synthesis of digital machines with provable epistemic properties,
SRI, Representation and Reasoning Program, Artificial Intelligence Center,
Computer and Information Sciences Division & Center for the Study of Language and Information,
Technical notes 412, April 1987,

[Ros 1989] Jean-Gérard Rossi,
La philosophie analytique,
N°2450 Que sais-je ? , Presses universitaires de France, 1989.

[RW 1910] William Russell, Alfred Whitehead,
Principia Mathematica,
1910, 12 et 13, Cambridge University Press, 1910.

[SA 1977] R.C. Schank & R.P. Abelson,
Scripts, plans, goals, and understanding,
Erlbaum, Hillsdale, N.J., 1977.

[Sab 1978] Gérard Sabah,
Contribution à la compréhension effective d'un récit,
Thèse de doctorat d'état, Univ Paris 6, 1978

[Sab 1988] Gérard Sabah,
L'IA et le langage,
Hermes, 1988.

[Sam 1958] Art. Samuel,
Some Studies in Machine Learning Using the Game of Checkers,
IBM Journal 3 3,210–229, 1958.

[SB 2006] Katty Sierra, Bert Bates,
Java Tête la première 2e édition,

O'Reilly, 2006.

[Sea 1972] John R. Searle,
Les Actes de langage,
Éditions Hermann, 1972.

[Sea 1980] John R. Searle,
The Chinese room, in *Minds, Brains, and Programs*,
Behavioral and Brain Sciences, 1980.

[Sea 1983] John R. Searle,
L'Intentionnalié,
Editions de Minuit, Paris, 1983.

[Seg 1992] Juan Segui,
Perception du langage et modularité,
Introduction aux sciences cognitives, essai Folio, 1992.

[SGG+1989] Jacques Sirioux, Michel Gilloux, Marc Guyomard, Christel Sorin :
Le dialogue homme-machine en langue naturelle : un défi ?,
Annales des télécommunications, FRA., Tome 44, n° 1-2, page 53-76, 1989.

[Sim 1947] Herbert Simon,
Administrative Behavior,
Macmillan, 1947.

[Smo 1992] Paul Smolensky,
IA connexionniste, IA symbolique et cerveau,
Intro° aux sciences cognitives essai Folio, 1992.

[Ste 1996] Luc Steels,
A self organizing spacial vocabulary,
Artificial Life Journal, pp. 1-23, 1996.

[Tay 1995] Tim Taylor,
From Artificial Evolution to Artificial Life
http://hyperzoic.com/papers/thesis/html/main.html,1995

[Tur 1950] Alan Mathison Turing,
Computing Machinery and Intelligence,
Mind, 1950.

[Wea 1949] Warren Weaver,
Memorandum,
from MT News International, no. 22, July 1999, pp.5-6, July 1949.

[Wei 1966] Joseph Weizenbaum,
ELIZA a computer program for the study of natural language communication between man and machine,
Communication of the ACM, vol 9, pp 36-45, 1966.

[Wer 1985] Harald Wertz,
LISP Une introduction à la programmation,
Mason, 1985.

[Wil 1956] Maurice Wilkes,
Automatic Digital Computers,
John Wiley & Sons, New York, 1956.

[Win 1971] Donald W. Winnicott,
Jeu et réalité, l'espace potentiel,
Éditions Gallimard, 1971.

[Win 1972] Terry Winograd,
Understanding natural language,
Academic press, New-York, 1972.

[Woo 1970] W. A. Woods,
Transition Network Grammars for Natural Language Analysis,
Communications of the ACM Volume 13, Number 10, October, 1970.

[ZT 1991] O. C. Zienkiewicz R. L. Taylor,
La méthode des éléments finis,
Afnor technique, 1991.

Ch. 44 Index thématique

Index lexical

action.9, 14-18, 50-54, 57, 59-63, 66, 70, 73-75, 79, 82, 83, 87, 91-94, 96, 97, 101, 102, 120-122, 125, 126, 129, 130, 137-148, 154-157, 161-174, 176-179, 181, 185-187, 190-192, 195, 196, 199-208, 210, 212, 215, 217, 219, 221, 222, 226, 228-230, 238, 241, 242, 252, 253, 255, 260-264, 267, 270-272, 274, 277, 289, 296, 297, 300-302, 307, 308, 310, 313, 323, 326, 329
agent intentionnel.....3, 21, 39, 40, 44, 118, 120, 184, 188, 190, 192, 196, 207, 271, 288, 290, 292, 327
agent réactif.....3, 16, 21, 44, 101, 139-141, 148, 159, 164-174, 176-179, 185, 190, 199-204, 206, 207, 210, 231, 253, 257-262, 268, 271-275, 280, 282-285, 289, 292, 293, 295, 297, 298, 326
agent situé .3, 7, 9, 10, 14, 17, 19, 21, 37, 39, 40, 44, 51, 87, 92, 94, 108, 111, 118, 120, 122, 129, 130, 137-142, 147, 148, 150, 152-160, 162-169, 172-174, 176, 185, 190, 195, 200, 201, 206, 217, 218, 221, 227, 230, 241, 245, 253-255, 257-260, 265-267, 269, 304, 323, 325-327, 329
agents rationnels.....3, 6, 21, 42, 99, 101, 174, 175, 217, 327, 334
arbre.....3, 11, 13, 15, 16, 38, 41, 44, 45, 66, 69, 89, 100, 104, 114-117, 126, 168, 183, 191, 199, 211, 212, 219, 224, 225, 229-232, 247, 268, 276, 277, 289, 293, 294, 301-303, 327-329
automate.....3, 6, 9, 10, 13, 14, 19, 21, 37, 50, 51, 82, 91, 108, 118-122, 130, 137, 138, 147, 157, 163, 164, 226-228, 232, 246, 261, 265, 266
causalité.....21, 97, 139, 144, 146, 157, 158, 163, 164, 166, 177, 178, 180, 232, 253, 254, 257, 259, 263, 266
cognitif.....1, 3, 5, 7, 8, 11, 16, 49, 73, 100, 104, 112, 114, 115, 117, 214-219, 221-224, 227, 231, 253, 257, 266, 314-316, 319, 323, 327, 329
cognition.....7, 10, 11, 21, 22, 37, 38, 40, 42, 43, 83, 99, 100, 102, 115, 217, 218, 220-223, 225, 226, 229, 232, 239, 319-321, 323, 329, 333, 336-338
communication.....7, 8, 16-18, 20, 37, 40, 41, 44, 45, 49, 69-71, 99-102, 104, 108, 112, 163, 182, 192, 193, 197, 223, 225, 226, 229-232, 279, 283, 284, 286, 287, 292, 306, 307, 309, 314, 324, 327-329, 336, 339
coopération.....21, 44, 102, 103
croyances.....3, 7, 9, 21, 45, 165, 174-176, 185, 188-190, 192-197, 199-208, 210, 214, 217, 223, 227, 234, 295, 307
culture.....40, 87, 88, 99, 100, 102, 115, 198, 199, 222, 229, 294, 306, 312, 323
décision.....14-16, 18, 52-54, 61, 66, 67, 82, 102, 119, 120, 122-124, 129, 130, 133, 134, 137-139, 141-143, 147, 148, 155-157, 162, 164-172, 174, 176, 184, 185, 190, 195, 196, 200, 201, 203, 204, 206, 216, 217, 221, 222, 226, 234, 245, 253, 258-267, 271, 274, 296, 321, 323, 328
désirs.....3, 21, 39, 165, 174-176, 185, 188-190, 194-196, 206-208, 210, 214, 217, 227
déterminisme.....3, 17, 21, 86, 162, 163, 327
empirisme.....3, 8, 17, 21, 162, 328, 329
enfant.....40, 57, 70, 75, 76, 102, 112, 158, 160, 170, 183, 194, 198, 199, 279, 293, 295, 297-302, 304, 306, 309, 310, 312, 314, 316, 320, 327
enseignement.....3, 17, 18, 21, 197, 225, 294, 297-300, 302, 304
grille.....3, 5, 7, 14, 21, 44, 85, 109, 153, 158, 179, 199, 201, 202, 206, 208, 213, 221, 223, 228, 231, 245, 248, 249, 264, 277, 309, 316, 317
héritage.....20, 21, 39, 160, 236, 325, 327, 329
IA.....3, 4, 9, 16, 38, 42-45, 54-56, 59, 65, 77, 78, 97, 103, 109, 111, 114-116, 170, 211, 214-216, 218-222, 225-228, 304, 330, 332, 338, 339
information7, 10, 12, 14, 16-19, 38, 40, 42, 44, 45, 48-50, 54-56, 59, 62, 64-66, 69-74, 76, 85, 93, 99-102, 106, 112, 114, 118, 120, 121, 123-125, 133-135, 151, 152, 162, 174, 182, 184, 185, 189-193, 195-197, 199, 201, 205, 217-219, 221, 222, 224-228, 230, 234, 253, 257, 260, 262, 266-268, 271-274, 276, 280, 283, 288, 294-296, 299, 300, 305, 309, 313, 314, 316, 321, 325-327, 331, 332, 336, 338
intelligence artificielle.....6, 7, 9, 14, 21, 190, 217, 221, 226, 228, 284, 332-334, 336-338
intentions.....3, 21, 22, 170, 174, 175, 188-194, 196, 197, 206, 208, 210, 214, 227
interprète.....5, 44, 54, 59, 71, 73, 87, 97, 101, 103, 112, 143, 196, 205, 207-211, 214, 215, 229, 231, 232, 326
jeu.....21, 42, 45, 83, 112, 114, 115, 127, 131, 133, 134, 141, 161, 184, 218, 219, 223, 232, 235, 244-247, 266, 284, 295, 296, 309, 314, 321, 334, 340
langage. .8, 18, 20, 21, 37, 43, 45, 48, 49, 56, 59, 61, 64, 66, 69, 71-73, 76, 86, 99, 100, 106, 171, 197, 199, 212, 222, 228, 235, 236, 305-307, 312, 313, 327, 329, 333, 336, 338, 339
linguistique.....1, 3, 5, 18, 21, 38, 47, 103, 104, 112, 331, 335
natriarche.....281
objets.....19-21, 37, 48, 49, 80, 83, 85, 96, 104, 118, 144, 146, 151, 153-155, 157-160, 164, 167, 174, 177, 179, 211, 226, 236, 241, 266, 269, 277, 325, 326, 328, 333
obstacles3-5, 21, 22, 51, 82, 83, 142, 162, 166, 168, 170, 177-179, 198, 201, 202, 206, 207, 221, 226, 230, 267, 269, 270, 283, 284, 288, 293-296, 298-300, 305
parent.....40, 57, 70, 75, 76, 102, 112, 158, 160, 170, 183, 194, 198, 199, 279, 293, 295, 297-302, 304, 306, 309, 310, 312, 314, 316, 320, 327
perception.....8, 11, 14-16, 18, 62, 64, 66, 69, 74, 87, 92, 100, 102, 114, 129, 141, 142, 144, 154, 157, 161, 162, 164, 166, 167, 169-172, 174, 188, 200, 201, 203, 204, 217, 222, 226, 229, 258-260, 262, 265, 267, 269, 271, 273, 274, 297, 323, 328, 329, 339
phylogenèse.....21, 49, 100, 124
planification.....44, 87, 97, 103, 196, 205, 207-211, 214, 215, 229, 231, 232
plate-forme.....3, 5, 6, 16, 21, 37, 47, 89, 108, 277, 327, 330
procès.....7, 8, 12, 16, 43, 48, 50, 51, 56, 57, 59-61, 79, 82, 91, 93, 94, 96, 97, 114, 115, 120-123, 126, 130, 133-140, 142-144, 147, 152-155, 157, 164, 165, 167-170, 172, 173, 176-179, 185, 186, 190, 192, 195, 197, 206, 217, 221, 242, 253, 260, 266, 268-272, 274, 277, 325, 326
programmation.....10, 12, 14, 15, 19, 21, 37, 44, 170-172, 180, 211, 212, 225, 234, 236, 277, 312, 318, 325, 330, 331, 338, 339
programme .12, 13, 38, 42-44, 48, 56-58, 68, 72, 74, 77, 84, 97, 99, 110, 114, 122, 125, 134, 162, 180, 191, 193, 196-198, 202, 204, 205, 208, 214, 223, 234, 236-239, 281, 294, 298, 305, 315, 318, 325
progression…3-6, 12, 13, 15, 16, 19, 21, 22, 37-41, 89, 90, 98, 100, 108, 109, 111-113, 118, 120, 122, 124, 142-144, 148, 153, 156, 158, 161, 168, 188, 198, 209, 225-236, 238, 239, 241, 252, 254, 258, 265, 273, 277, 278, 282, 290, 295, 305, 306, 312-314, 316, 328
rationalité.....21, 38, 100, 336
rationnel.....7, 8, 16, 44, 49, 67, 100, 118, 120, 164, 165, 174, 176, 185, 190, 195, 206, 217-224, 227, 230, 239, 319, 326, 327, 329
séquenceur.....3, 13, 15, 21, 37-39, 50, 82, 108, 112, 118, 120-124, 126, 130-134, 136-138, 141, 147, 153-155, 157, 164, 166, 200, 201, 226, 232, 246, 327
simulation .1-7, 11, 13, 14, 16-22, 37-41, 44, 45, 47, 51, 79-84, 87, 89-95, 97-101, 108, 109, 111-113, 120, 122, 124, 125, 129, 136-138, 142-144, 146, 148, 152-155, 157, 160-162, 166, 168, 170-172, 177, 184, 186, 196, 198-205, 207-209, 214, 215, 218, 221-231, 233-235, 238, 239, 241, 243, 245-248, 252-262, 265-269, 273-275, 277, 279-283, 287-291, 293-297, 304-307, 309, 311-314, 316-322, 326-330, 335
SMA.....9, 59-61, 110, 221, 330

table….7, 11, 13, 18, 37, 38, 41, 47, 50, 53, 57, 59, 61, 65, 67, 79, 82, 84, 92, 93, 96, 97, 99, 101, 110, 116-118, 120, 121, 126, 130, 131, 133, 134, 138, 141, 143, 147, 151, 152, 155, 157, 164, 166, 167, 172, 185, 190, 191, 196, 197, 200, 205, 211, 217, 219, 221, 222, 226, 228, 231, 233, 236-239, 241, 244-252, 256, 258, 259, 261-265, 268, 269, 274, 275, 281, 284, 288-291, 296, 297, 309, 310, 312, 318, 319, 329
taxinomie.....314
Traitement.....7-9, 11, 12, 14, 16, 17, 19, 38, 40, 42-45, 48-50, 52-56, 59, 62, 64-67, 71-73, 79, 88, 93, 100-102, 104-106, 109, 114, 115, 117, 120, 123-125, 134, 139, 142, 143, 154, 155, 157, 160, 167, 174, 180, 184-186, 190, 195, 197, 199, 205, 208, 209, 211, 212, 214-231, 238, 239, 243, 249, 253, 257, 260, 262, 266-268, 272-274, 280, 283, 284, 288, 289, 294-296, 309, 310, 312-316, 319, 321, 323, 325-327, 329
émergence.....3, 6, 13, 14, 16, 17, 19, 21, 39, 40, 85-87, 98-100, 103, 108, 110-112, 163, 224, 230, 231, 327, 328
évolution.....9, 16, 17, 21, 37, 39, 47, 49, 85-88, 98-103, 108, 110-114, 117, 125, 143, 153, 158, 168, 171, 199, 201, 204, 221, 223-225, 229, 233, 253, 259, 272, 277, 279, 305, 313, 325-329