Le séquenceur pour faire de la simulation

Introduction : effectuer des simulations

Dans le quotidien, faire de la simulation à propos du monde réel consiste à imaginer comment il évolue au fil du temps sous l'action des agents qui y agissent.

Nous venons d'étudier les séquenceurs, qui sont des Systèmes Formels (SF) très simples, et nous allons les utiliser pour faire de la simulation, car ils s'avèrent un bon outil pour cela. Mais, au préalable, posons quelques jalons, étudions les bases qui nous seront nécessaires pour cela.

Taxinomie des séquenceurs

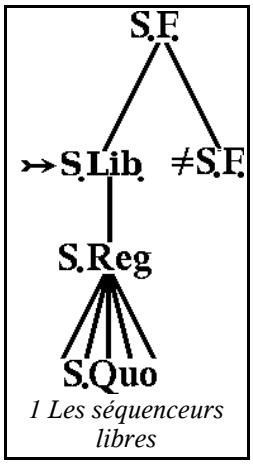
Nous distinguons différentes familles de séquenceurs ; rangeons-les au sein d'un arbre d'héritage.

1) Les SF

Tout au sommet de cet arbre, apparaissent les systèmes formels.

2) Les Séquenceurs libres (SL) ou *séquenceurs à états libres* : sous-classe des SF

Dans le haut de cet arbre, dessous le nœud des SF, se trouvent les séquenceurs à états libres. Ils sont définis et construits comme des séquenceurs dont le domaine est constitué de variables libres[1] et dont le graphe découle de la donnée des règles de réécriture du SF.



1 Les séquenceurs libres

Les SL généralisent une grande variété de séquenceurs

De par leur nature, les séquenceurs sont abstraits. Grâce à cette construction, ils généralisent l'ensemble de tous les séquenceurs imaginables (de toutes les formes de graphe, et de tous les procès[2] possibles).

Décrire le traitement effectué par les SL

Maintenant, voyons pourquoi le procès qu'ils effectuent est indéterminable. Cependant, nous constaterons qu'ils effectuent un traitement caractérisé par la forme de leur graphe.

Séquenceurs dont le domaine est constitué de variables libres

D'un côté, la donnée des règles de réécriture détermine la topologie du graphe du séquenceur, mais regardons plutôt le contenu des état de ce graphe. Nous voyons que les règles des SL font référence à des variables libres, donc au sein d'un séquenceur les procès (les transitions d'état produites par l'application des règles de réécriture) demeurent très variables et donc difficiles à caractériser.

Les SL sont construits au moyen de symboles non connotés, i.e. les états du graphe sont étiquetés par des variables libres. À chaque transition d'état, ce remplissage non contraint laisse le procès effectué par les SL totalement indéterminé. Ceci les rend tellement abstraits qu'on n'envisage pas de les charger de sens. La verbalisation du procès de réécriture se fait au moyen de verbes abstraits. Elle se limite pratiquement à constater que *le système change d'état*.

Ainsi il est impossible d'utiliser ces systèmes pour faire de la représentation, pour modéliser un micro-monde, et encore moins pour simuler des transformations en son sein.

Séquenceurs dont le graphe est déterminé par la donnée des règles

La donnée d'un jeu de règles détermine la topologie du graphe d'un séquenceur. Au début de ce cours, pour passer des SF aux séquenceurs, nous avons spécifié des restrictions sur les règles afin qu'elles induisent une régularité dans les

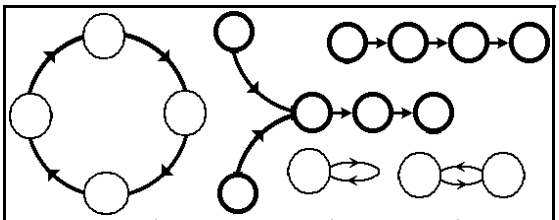
1 Variables qui peuvent êtres instanciées sans restriction de type.

2 Procès : transition d'état. Ici il s'agit de la transition, du changement de l'état du système formel.

graphes. Ainsi émergent des structures caractéristiques : notamment des parcours linéaires, bouclés ou confluents. En conclusion, seule la topologie de ce type de séquenceur caractérise le traitement qu'il produit à l'occasion du parcours de son graphe.

Contrairement au traitement décrit par le procès de réécriture, celui effectué au niveau du graphe est plus facilement caractérisable. À chaque topologie, il est plus aisé d'attacher une verbalisation. Quand le séquenceur parcourt une période très courte, on peut dire qu'il oscille ; quand elle s'allonge, on dit qu'il boucle ou qu'il cycle. Enfin, quand la période devient très grande, le fait que le graphe se reboucle constitue une information, un événement. Alors, on commente : à partir d'ici il recommence son cycle.

La figure ci-contre, présente une instance de chaque principale famille de graphe de séquenceur : à gauche, une boucle ; en haut à droite, une séquence linéaire ; au milieu, un confluent ; au milieu en bas, un état permanent ; et en bas à droite, un bistable.

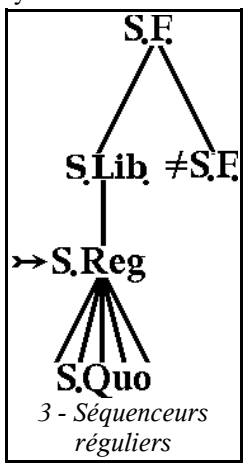


2 - Quelques instances des principales familles de graphes de séquenceurs.

Note sur une convention de notation graphique

Dans un SL, chaque état est étiqueté par une variable libre différente car il s'avère impossible de dégager une régularité au sein de son domaine. Ainsi, dans le dessin ci-contre, aucun des états n'est étiqueté par un symbole, car nous avons choisi cette autre convention de notation, plus intuitive : les états sont représentés vides pour exprimer qu'il ne sont pas contraints. C'est pourquoi appelle cette famille les séquenceurs à états libres : une fois la topologie du graphe définie, le contenu de chaque état peut être instancié par n'importe quel symbole.

3) Séquenceurs réguliers, un domaine numérique défini en compréhension



3 - Séquenceurs réguliers

Introduction

Au milieu de la taxinomie (arborescence), se trouvent les séquenceurs réguliers (SR).

Par construction, ils héritent de leurs parents les séquenceurs libres, des graphes réguliers (linéaires, bouclés ou confluents).

Le domaine des SR est défini en compréhension[3]

Depuis l'état de départ e_0, il est construit par récurrence, en utilisant une fonction *f* binaire analytique : $e_1=f(e_0)$, $e_2=f(e_1)$ ... $e_{n+1}=f(e_n)$. Donc ils prennent leurs valeurs dans des sous-ensembles de Z (entiers positifs ou négatifs).

Les états du domaine des SR sont calculés en itérant une formule de récurrence

Par définition, les états des SR sont définis en compréhension et en fonction de l'état précédent :

$e_{n+1} = f(e_n)$ // où f est une fonction

À la réécriture suivante, on obtient de même :

$e_{n+2} = f(e_{n+1})$ // où f est la même fonction car elle est définie en compréhension

En remplaçant, dans la ligne ci-dessus, e_{n+1} *par sa valeur* $f(e_n)$ on calcule :

$e_{n+2} = f(f(e_n))$

À la réécriture suivante, on procède de même :

$e_{n+3} = f(f(f(e_n)))$

Finalement on déduit la forme générale :

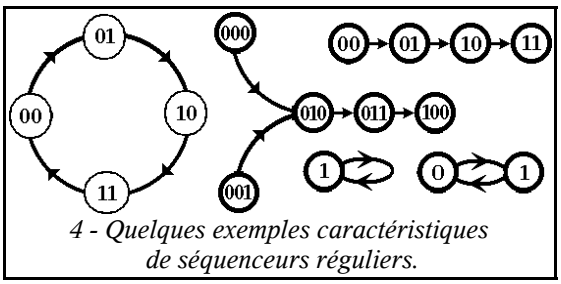
$e_{n+k} = f^k(e_n)$

On conclue que le $k^{ième}$ terme est obtenue par itération, en composant k fois la fonction f.

Par construction, leur graphe est celui d'un séquenceur

Dans la famille des séquenceurs libres nous avons déjà remarqué et décrit des topologies caractéristiques de graphes : *linéaires*, *bouclés* et *confluents*. Puisque les SL sont parents des séquenceurs réguliers, ces derniers, dès leur création, héritent aussi de ces structures.

Remarque : dans la suite de notre étude, nous délaisserons un peu les *confluents* pour focaliser de préférence sur les *bouclés* et les *linéaires*.

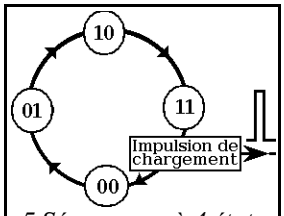


4 - Quelques exemples caractéristiques de séquenceurs réguliers.

3 En mathématiques, la définition d'un ensemble est dite *en compréhension*, si on fournit une méthode qui permet de construire systématiquement la liste de ses éléments.

Pour illustrer les séquenceurs réguliers, dans la figure ci-dessus, nous en présentons cinq exemples : à gauche, une boucle ; en haut à droite, une séquence linéaire ; au milieu, un confluent ; au milieu en bas, un état permanent ; et en bas à droite, un bistable.

Aspect effectif des séquenceurs réguliers



5 Séquenceur à 4 états utilisé en diviseur par 4

De par leur construction, ces séquenceurs peuvent modéliser les phénomènes de la vie quotidienne. Puisqu'ils sont généraux, et construits par abstraction, leurs symboles binaires ne sont pas chargés de contenu, mais leurs graphes sont chargés de propriétés structurantes. et leurs états peuvent des procès.

Par exemple, regardons d'abord ce séquenceur en boucle caractérisé par la topologie de son graphe. Son domaine est {00, 01, 10, 11} et ses règles de transition sont : 00→01, 01→10, 10→11, 11→00.

D'un côté, cette structure de graphe, qui s'apparente au décodeur/démultiplexeur de la logique, est utilisée dans la technologie *token ring* des réseaux pour distribuer des jetons alternativement aux nœuds de l'anneau : soit le cas d'un sous-système n, connecté à l'état N ; à chaque instant élémentaire, le séquenceur passe à l'état suivant, quand il arrive à l'état N, alors n reçoit un jeton et est validé.

Et de l'autre côté, on peut encore l'utiliser pour faire un diviseur modulo 4 : à chaque instant élémentaire, le séquenceur reçoit une impulsion et passe à l'état suivant. À chaque fois qu'il transite de l'état 11 à l'état 00, il envoie une impulsion à un autre sous-système. Ainsi, ce dernier reçoit 4 fois moins d'impulsions que le séquenceur.

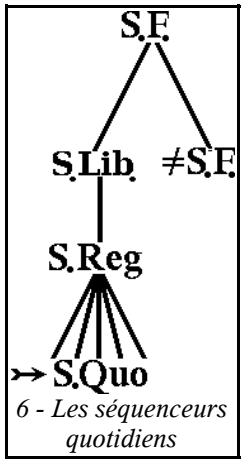
Finalement, on constate ici que la topologie du séquenceur fixe le facteur de la division modulo, et que ses procès déterminent le lieu des impulsions/validations. En conclusion, la prestation fournie par le système dépend de sa structure : elle relève à la fois de sa topologie et de ses procès.

En conclusion :

Depuis l'ensemble immense des SF, chaque étape a élagué drastiquement. Mais l'ensemble de départ est si grand qu'à l'arrivée, il reste encore de nombreux séquenceurs construits sur des séries entières définies par récurrence. Par exemple, nous pouvons citer les six instances de séquenceurs traitées au début du poly :

Type de séquenceur	***Son domaine de définition***	***Sa formule de récurrence***
cellule permanente	{0}	$e_{n+1} = e_n$
alternateur	{0, 1,}	$_{en+1} = \neg\ e_n$; (avec ¬ 1=0 ; et ¬ 0=1).
séquenceur linéaire	{000, 001, 010, 011, 100}	$e_{n+1} = 1 + (e_n)$
boucle	{000, 001, 010, 011}	si (e_n==11) $e_{n+1} = 0$ sinon $e_{n+1} = 1 + (e_n)$
décroissance linéaire	{100, 11, 10, 1, 0}	$e_{n+1} = e_n - 1$
série géométrique de base r = 1,5	{100, 110, 1001, 1101, 10011}	$e_{n+1} = 1{,}5 * e_n$

4) Les séquenceurs quotidiens (SQ)



6 - Les séquenceurs quotidiens

Selon une définition informatique, ce sont des instances typées de séquenceurs réguliers. Mais, de façon plus imagée, on peut encore dire que ce sont des séquenceurs réguliers, plongés des domaines particuliers, par lesquels le cogniticien[4] s'emploie à modéliser la vie de tous les jours.

Positionnement

Ensuite, juste dessous les séquenceurs réguliers, nous plaçons les séquenceurs quotidiens.

Quelles sont les raisons de ce placement ?

Les SQ sont placés dessous leurs parents car ils héritent de leur structure (graphe et procès) et parce qu'ils sont plus spécifiques : ils héritent du type du domaine où ils sont plongés.

Le travail du cogniticien est de modéliser le réel, la vie quotidienne

La vie quotidienne est structurée par de multiples règles : les lois physico-chimiques de la nature, celles des échanges socio-économiques au sein groupe, celles de la psychologie.

Le travail du cogniticien est de modéliser ces structures et ces comportements. Il doit trouver un SR dont la sémantique, i.e. le fonctionnement (graphe + procès) soit proche de celui du monde à modéliser.

Il prend une instance de SR. Il lui ajoute un type, un marqueur spécifique au domaine où se déroule l'action modélisée. Ainsi il obtient un micro-monde, un séquenceur qui se comporte, évolue, change d'état, mais les états de son domaine sont typés.

Ainsi les SQ sont obtenus en les plongeant les SR dans un domaine spécifique. Le cogniticien instancie les séquenceurs réguliers, et leur ajoute un type provenant du domaine où ils sont plongés. Ce faisant, les états numériques des séquenceurs réguliers deviennent des états typés de la forme (nombre, type)[5].

Plonger un SR dans un domaine lui fait hériter le type correspondant

Exemple de la base de temps :

Par exemple, plonger un séquenceur régulier dans le temps (la dimension temporelle), le transforme en horloge, en base de temps. À la lumière de ce nouvel éclairage, *passer d'un état à un autre* se verbalise en : *passer de l'heure n à l'heure n+1*. Ainsi, l'état 0 du SR, se concrétise en *le temps 0* qu'on peut noter indifféremment : t_0, *temps0,* ou encore *(temps 0).*

Exemple de la base spatiale :

Par exemple, plonger un séquenceur régulier dans l'espace (la dimension spatiale), le transforme en chemin, en trajet. À la lumière de ce nouvel éclairage, *passer d'un état à un autre* se verbalise en : *passer du lieu n au lieu n+1*. Ainsi, le lieu 0 du SR, se concrétise en *le lieu 0* qu'on peut noter indifféremment : $lieu_0$, *lieu0*, ou encore *(lieu 0)*.

Plonger un séquenceur conserve graphe et procès mais change ses symboles

Au passage, on retrouve ici ce qui est annoncé en introduction. Plonger un séquenceur dans un domaine spécifique, d'un côté conserve son graphe et ses procès, mais de l'autre particularise, type ses symboles. On lui attribue des symboles connotés, i.e. que dans chaque cas, on transforme les nombres binaires du SR dans des symboles typés, appartenant à la même famille.

Ainsi, dans l'exemple du *plongeon temporel*, les symboles t_i du SQ obtenus appartiennent à une classe, une famille, à un type du *temps*. De même, dans celui du *plongeon spatial*, les symboles qui sont fabriqués appartiennent à une classe, une famille, à un type de l'*espace* : quand on verbalise, on le nomme des *lieux*.

Conclusion :

Appliquer un SR à un domaine particulier, lui attribue un type. Les transformations qu'il subit cessent d'être des procès abstraits. En s'incarnant dans un domaine particulier, le SR prend chair, se concrétisent, et devient un séquenceur quotidien, qui permet de simuler des actions sur ce domaine.

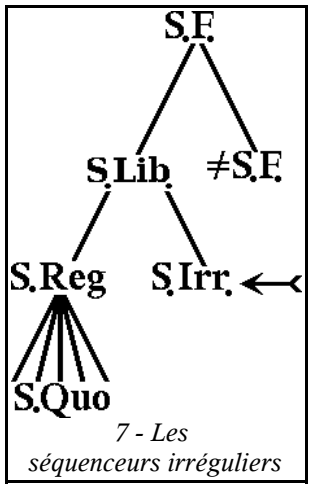
Dans la suite de ce polycopié, nous nous attacherons à montrer comment les séquenceurs, s'avèrent d'excellents outils pour faire de la simulation.

---

4 En IA, le cogniticien est un spécialiste de la représentation des connaissances. Il observe le monde, analyse son état et les règles qui le gouvernent, et travaille à décrire l'état du micro-monde de départ de la simulation.

5 On peut encore trouver d'autre formes équivalentes : typenombre, $type_{nombre}$, [type nombre]. Ex : nom1, nom_1, [nom 1].

5) Les séquenceurs irréguliers (SI), au domaine défini en extension



7 - Les séquenceurs irréguliers

Enfin, au sein de la taxinomie où nous classons les séquenceurs, plaçons les séquenceurs irréguliers (SI).

Parmi les exemples de séquenceurs que nous traitons, les (SI) sont seulement définis en extension. Plusieurs raisons peuvent expliquer cette absence de définition en compréhension :

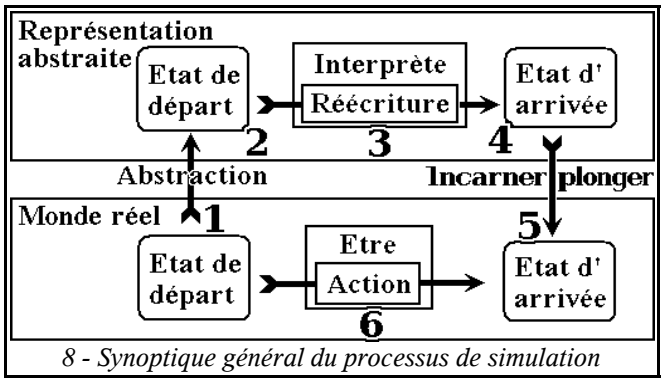
- Ils possèdent un domaine très irrégulier.
- Ils possèdent un domaine trop difficilement descriptible en compréhension.
- Leur domaine a été nommé par des conventions arbitraires que nous ignorons, traduits dans un langage dont nous ne possédons pas le dictionnaire. Ex : les séquenceurs spatiaux, basés sur les lieux : le trajet *Chartres→Nantes*, ou encore le trajet *LieuA→LieuZ→ LieuE→LieuR→LieuT*.

Bref, nous ne disposons pas de définition en compréhension de leur domaine : il se représente seulement en extension. Par ailleurs, leur fonctionnement est représenté exhaustivement par la donnée d'une liste ou d'un tableau de règles.

Pour ces raisons, quand nous ferons de la simulation, nous traiterons d'un côté les séquenceurs réguliers, et nous simulerons autrement avec les SI.

Principe général et synoptique de la simulation

Après avoir classé les différentes familles de séquenceurs au sein d'une taxinomie, étudions maintenant le principe général et le synoptique de la simulation en IA. Elle implique six étapes différentes que nous allons maintenant analyser et détailler.



8 - Synoptique général du processus de simulation

Étape 1 : départ depuis le micro-monde réel

Au départ, on se donne un micro-monde, avant qu'il ne subisse une action.

Étape 2 : une représentation du micro-monde

Une étape de description, abstraction décrit en termes formels l'état du micro-monde.

Étape 3 : la simulation proprement dite

L'étape 3 consiste en la simulation proprement dite : étape après étape, un interprète applique des règles pour réécrire la représentation abstraite de l'état du micro-monde.

Étape 4 : le résultat de ce traitement simulé

À l'issue de la démarche de simulation de l'étape 3, on obtient le résultat de ce traitement : il décrit l'état d'arrivée abstrait calculé du micro-monde.

Étape 5 : état d'arrivée simulé du micro-monde

Finalement, la dernière étape de cette simulation plonge le résultat 4, l'état formel calculé du micro-monde, dans la réalité quotidienne en l'incarnant dans le monde réel. Ainsi il fournit l'état d'arrivée simulé.

Étape 6 : évolution réelle du monde, au fil du temps

Sur le synoptique général du processus de simulation, nous représentons aussi l'étape 6 qui correspond à l'évolution réelle du monde, au fil du temps ; il s'agit bien sûr de l'étape que nous simulons au travers des cinq étapes précédentes.

Observer le monde réel et modéliser sa structure au moyen d'un séquenceur

Avant de pouvoir expliquer la démarche de modélisation du monde réel (de son état et de ses règles), au moyen d'un séquenceur, nous devons introduire une définition :

Introduction de la notion de *séquenceurs quotidiens équivalents*

Définition de deux *séquenceurs quotidiens équivalents*

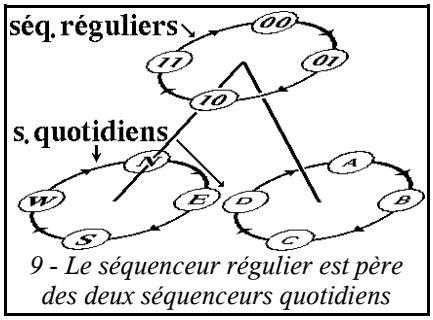
Deux séquenceurs quotidiens sont équivalents s'ils présentent la même forme profonde, i.e. s'ils possèdent le même graphe, et si, à chaque étape de ce graphe, ils effectuent le même procès. En fait, ils ne diffèrent que par leur typage.

Parfois des SQ sont équivalents car ils dérivent du même parent : ils possèdent une même forme profonde (un même graphe et un même procès), mais présentent deux formes de surface typées différemment, car ils sont plongés dans des mondes concrets dissemblables.

Classe d'équivalence de séquenceurs quotidiens

Quand deux séquenceurs quotidiens présentent la même structure profonde, i.e. le même graphe et les mêmes procès, ils sont équivalents. On les range dans la même classe d'équivalence, et ils sont représentés par le même séquenceur régulier parent.

Exemple - le séquenceur qui boucle de 00 à 11



9 - Le séquenceur régulier est père des deux séquenceurs quotidiens

Dans le dessin ci-contre, regardons en bas à gauche, le séquenceur quotidien en boucle, dont le domaine est constitué des lettres *N, E, S, W* (*pour nord, est, sud, ouest*). Selon les conventions de la navigation aérienne, ces caps correspondent aux valeurs *0°, 90°, 180°, 270°*. Pour abstraire ce séquenceur, on ôte son typage en degré d'angle, on divise ces valeurs par 90, et on le généralise par le séquenceur *S*, dont le domaine *D = {00, 01, 10, 11}* permet de compter les objets d'un ensemble de quatre éléments.

Dans le dessin ci-contre, regardons en bas à droite, le séquenceur quotidien en boucle, dont le domaine est constitué des lettres *A, B, C* et *D*. Selon le code ASCII, ces lettres correspondent aux valeurs hexadécimales *65, 66, 67* et *68*. Pour abstraire ce séquenceur, on ôte son typage en soustrayant à ces valeur *65*, et on le généralise avec le même séquenceur *S*.

Finalement, on déduit, qu'au prix des ces deux aménagements de type (division par 90 et soustraction de 65), les deux séquenceurs quotidiens sont équivalents, car ils sont représentés par le même parent : le séquenceur régulier *S*.

Plongeon SR→SQ : de l'abstrait au concret

Matérialiser un SR en le plongeant dans le monde quotidien lui fournit un type. Pour incarner, matérialiser deux séquenceurs réguliers équivalents, on type chacun en le plongeant dans un monde réel et quotidien différent. Ainsi, au niveau de sa forme de surface, chaque SQ reçoit un type particulier et est rangé dans une classe spécifique.

Quelques précisions sur le verbe plonger

Le terme *plonger* est un jargon rapide et facile d'épistémologue. Maintenant nous allons détailler et formaliser cette opération au moyen de la famille des séquenceurs réguliers.

Il s'agit de plonger un SR dans différentes situations de la vie quotidienne. Le même séquenceur se trouve donc utilisé plusieurs fois. Il conserve la même structure (caractérisée par son graphe et ses procès), mais il est mis en œuvre dans différents contextes, i.e. dans différents domaines : des symboles issues de différentes familles étiquettent les nœuds de ses graphes.

Exemple 1 :

Soit le séquenceur à croissance linéaire *S*, dont le domaine numérique binaire est : *{001, 010, 011, 100, 101}*. Nous le plongeons dans différents domaines : alphabétique, décimal, spatial et temporel. Ainsi nous obtenons les quatre instances suivantes de séquenceurs :

S_1 dont le domaine est : *{a, b, c, d, e}*. // Domaine alphabétique
S_2 dont le domaine est : *{1, 2, 3, 4, 5}*. // Domaine décimal
S_3 dont le domaine est : $\{lieu_1, lieu_2, lieu_3, lieu_4, lieu_5\}$. // Domaine spatial
S_4 dont le domaine est : $\{t_1, t_2, t_3, t_4, t_5\}$. // Domaine temporel

Exemple 2 :

Soit le séquenceur irrégulier, dont le domaine défini en extension est : *{A, Z, E, R, T}*. Nous le plongeons dans les domaines alphabétique et spatial. Ainsi nous obtenons les deux instances suivantes de séquenceurs :

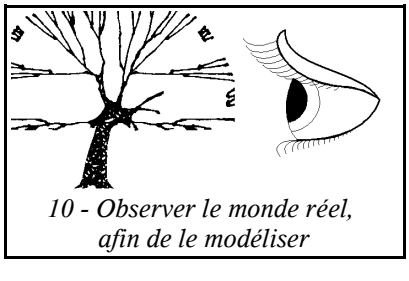
S_1 dont le domaine est : *{LettreA, LettreZ, LettreE, LettreR, LettreT}*. // Domaine alphabétique
S_2 dont le domaine est : *{Paris, Chartres, Le_Mans, Angers, Nantes}*. // Domaine spatial

Dans ce cas, les ensembles sont moins réguliers et la démarche ne fonctionne pas parfaitement : elle est à peine inversible.

Observer, pour modéliser le monde réel au moyen d'un séquenceur régulier

Étape d'analyse

Quand on observe le monde réel afin de le modéliser au moyen d'un séquenceur régulier, on passe d'abord une étape d'analyse. Le but est d'identifier le SQ correspondant au monde analysé, a niveau de son graphe et aussi de ses procès.


10 - Observer le monde réel, afin de le modéliser

Étape d'abstraction

Puis vient une étape d'abstraction (ab - trahere : tirer de quelque chose). On généralise, on transforme le quotidien vers le régulier, en abandonnant le typage. Abstraire un séquenceur qui est plongé dans un domaine, c'est abandonner son type et ne garder que son graphe et ses procès numériques.

Exemple - des séquenceurs qui bouclent de 1 à 5

Par exemple, reprenons les quatre séquenceurs à croissance linéaire S_i de l'exemple précédant. On peut réécrire leur domaine ainsi :

S_1 dont le domaine est : *{lettre_1ière, lettre_2ième, lettre_3ième, lettre_4ième, lettre_5ième}* // type alphabétique
S_2 dont le domaine est : *{1, 2, 3, 4, 5}*. // type décimal
S_3 dont le domaine est : *{lieu1, lieu2, lieu3, lieu4, lieu5}*. // type spatial
S_4 dont le domaine est : *{t1, t2, t3, t4, t5}*. // type temporel

Ainsi, en laissant tomber leur type les séquenceurs S_1, S_3 et S_4 s'abstraient d'abord en S_2, puis ce dernier s'abstrait en S dont le domaine numérique binaire est : *{001, 010, 011, 100, 101}*.

Finalement, pour chacune de ces classes, nous dégageons un représentant. Ce dernier apparaît bien comme une abstraction du séquenceur, car il ne possède plus de type, mais conserve le graphe et les procès.

Exemple - des séquenceurs qui bouclent de 1 à 7

Regardons par exemple le conte de Blanche-Neige, et particulièrement le monde de ses sept nains. Il se décrit par la liste : *Atchoum, Prof, Grincheux, Simplet, Dormeur, Timide* et *Joyeux* ; qui permet de construire le séquenceur quotidien : S_n = *{Nain1, Nain2, Nain3, Nain4, Nain5, Nain6, Nain7}*. En abandonnant le typage on abstrait S_n en S, séquenceur régulier dont le domaine est constitué des entiers de *001* à *111* : D_S = *{001 ... 111}*. Défini en compréhension, il permet de compter les objets d'un ensemble de sept éléments.

Prenons un autre exemple, la liste des jours de la semaine : *lundi, mardi, mercredi, jeudi, vendredi, samedi* et *dimanche*. Elle permet de construire le séquenceur quotidien : S_j = *{Jour1, Jour2, Jour3, Jour4, Jour5, Jour6, Jour7}*. En abandonnant le typage on abstrait S_j vers S, le séquenceur régulier précédent.

Les deux séquenceurs quotidiens S_j et S_n sont équivalents. Le premier S_j peut ainsi compter les jours de la semaine, et le second S_n les nains de Blanche-neige ; mais quand ils sont abstraits, dans la classe des SF réguliers, les symboles du domaine *{001 ... 111}* de leur père *S* ne sont plus typés, s'avèrent moins chargés de contenu et sont semblables.

Bilan

À ce stade nous avons montré comment on observe le monde réel afin de le modéliser au moyen d'un séquenceur régulier. Maintenons détaillons les étapes de simulations effectuées au moyen de séquenceurs.

Le séquenceur utilisé en simulation

Le séquenceur, est un outil très simple, cependant, il est déjà capable de servir en IA pour faire de la simulation.

La simulation : imaginer l'évolution de notre monde réel

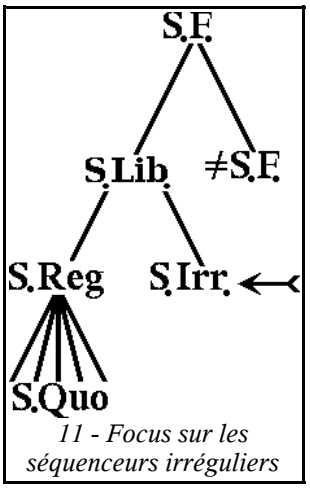
Commençons par préciser la **définition de la simulation classique en IA**. D'abord, Pour simuler l'évolution d'un monde réel, un être doit posséder un modèle suffisamment précis du support et des agents qui y évoluent. Ensuite, il doit connaître l'état de départ de ce monde, la position du mobilier et des agents sur le support, et encore leurs buts, plans et projets. Enfin, il doit réfléchir et imaginer l'évolution de ce monde sous l'action des agents.

Pour l'instant, munis des séquenceurs, nous n'allons pas pouvoir effectuer une simulation aussi fine ; mais voici ce que nous allons réussir à faire.

Les étapes d'une simulation effectuée au moyen d'un séquenceur irrégulier

Introduction

Tout d'abord, nous commençons à faire de la simulation au moyen des séquenceurs irréguliers, i.e. dont le domaine ne peut être défini qu'en extension.



11 - Focus sur les séquenceurs irréguliers

Étape 1 et 2 - Démarche de modélisation : écrire le séquenceur

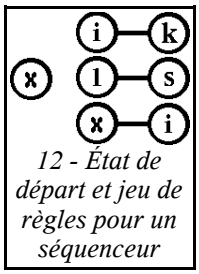
Démarche de description au moyen d'un séquenceur

À ce stade du cours nous travaillons seulement avec les séquenceurs. Ce sont les seuls outils de travail dont nous disposons. Nous devons utiliser un séquenceur pour décrire le monde analysé, et pour cela nous savons que ce dernier dérive d'un SF. Il est seulement un peu transformé : nous l'avons légèrement contraint par quelques spécifications portant sur la formation des règles de réécriture.

Donc, comme pour un SF, dans une démarche de modélisation, nous devons au moins décrire l'état de départ du micro-monde et les règles de réécriture. Par contre, nous n'avons pas à décrire l'état d'arrivée. En effet, ici, nous fonctionnons en simulation, et donc, notre but est de le deviner.

La description de l'état de départ constitue une démarche d'abstraction :

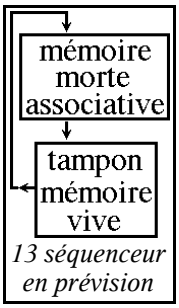
Pour décrire l'état de départ du système, l'informaticien observe le monde réel, puis il analyse sa perception. Dans une démarche d'abstraction, il décrit l'état du monde analysé. Pour ce faire, dans le cadre de cette modélisation au moyen d'un séquenceur, il doit décrire l'état de départ du système en fournissant un unique symbole. Par exemple, dans la figure ci-dessous, il fournit la lettre *x*.



12 - État de départ et jeu de règles pour un séquenceur

Démarche d'abstraction : Décrire la base de règles

Ensuite, l'informaticien observe les transformations que subit le monde réel. Dans une démarche d'analyse et d'abstraction, il en décrit les règles de transformation (de réécriture) sous la forme :
ÉtatDuMondeAvantLaTransformation→ÉtatDuMondeAprèsLaTransformation.



13 séquenceur en prévision

Notez que la figure ci-contre à droite présente un exemple de base de règles pour un séquenceur irrégulier. Puisqu'elle se réduit à la forme *Symbole1→Symbole2*, elle peut aussi se mémoriser sous la forme d'une mémoire morte associative *Clé→Valeur* (figure de gauche).

La démarche de catégorisation n'est pas forcément exacte

Quand un informaticien décrit, modélise une situation au moyen d'un séquenceur, il fait aussi une démarche de catégorisation, i.e. une démarche qui n'est pas forcément rigoureuse. Il connaît (ou croit connaître) les différents comportements, i.e. les multiples trajets du système, et il essaie de les regrouper dans une catégorie. Mais rien ne garantit qu'il y parvienne. Certains traits particuliers du processus modélisé peuvent lui avoir échappé.

On retombe ici sur le problème du lien *signifiant→signifié*, qui correspond encore au problème que nous avons déjà évoqué, i.e. à celui du contenu du symbole qu'on utilise pour étiqueter une catégorie.

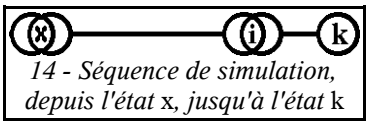
En conclusion, dans ce cours nous avons clairement dit que le processus formel d'application des règles dans les SF est rigoureux, par contre, nous constatons ici que l'étape de description/modélisation du monde réel peut pêcher.

Étape 3 - Transition d'état, par application de règle de réécriture

Ensuite, le système simule l'évolution de l'état du micro-monde en déroulant une séquence : tant que c'est possible, il applique la règle de réécriture permettant une transition, et fournit un nouvel état.

Tant qu'il trouve une règle dont la tête correspond à l'état du micro-monde, il l'applique, il efface l'état du micro-monde correspondant à cette tête de règle pour le remplacer par la queue de règle, et donc il obtient un nouvel état du système. Ainsi, à chaque procès, le symbole qui décrit l'état du micro-monde est remplacé.

Dans l'exemple ci-contre, partant de l'état *x*, il applique d'abord la règle $x \rightarrow i$, et obtient l'état *i* ; ensuite, en appliquant la règle $i \rightarrow k$, il obtient l'état *k*.



14 - Séquence de simulation, depuis l'état x*, jusqu'à l'état* k

Étape 4 - L'arrêt du système donne l'état d'arrivée du micro-monde

Si, le système arrive dans un état tel qu'aucune règle ne peut se déclencher, la situation obtenue correspond à une condition d'arrêt. La séquence de transitions s'arrête : le système stoppe. On obtient l'état d'arrivée du micro-monde : il est réduit à un symbole.

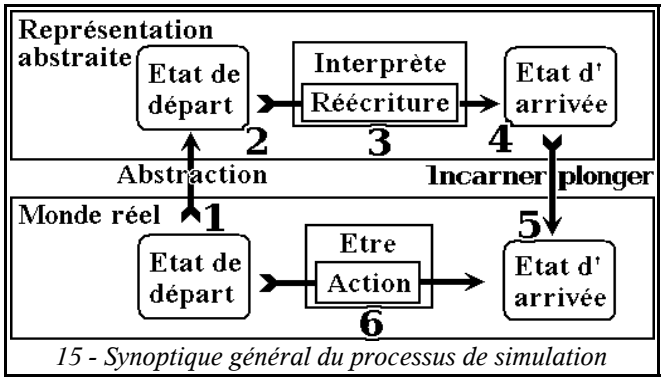
Dans l'exemple ci-dessus, plus aucune règle ne se déclenche, alors on est arrivé à l'état final *k*.

Étape 5 - Exploitation du résultat obtenu

Ainsi le traitement effectué fournit un symbole qui décrit l'état d'arrivée calculé du micro-monde.

Étape 6 - Vérification du calage du modèle

En comparant cet état du micro-monde obtenu par simulation avec celui qu'on peut observer dans la réalité, on peut se faire une idée de la qualité de la modélisation et, si nécessaire, recaler la simulation, i.e. réécrire les règles.



15 - Synoptique général du processus de simulation

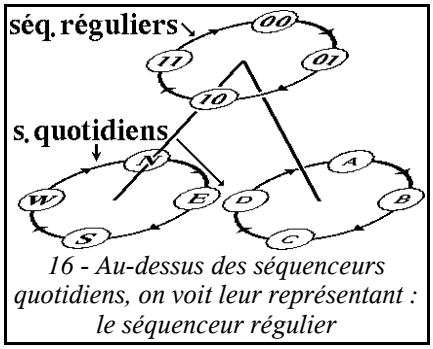
Remarque : ce recalage du modèle, n'est pas anodin ; il est très complexe et constitue une démarche nouvelle traitée plus longuement à la fin du cours, dans la partie à propos du traitement cognitif.

Application à un exemple de gestion de l'espace : vérification d'un itinéraire séquentiel

Maintenant nous décrivons une application de cette méthode, nous simulons, au moyen d'un séquenceur, la vérification d'un itinéraire. L'agent se pose la question suivante : *en commençant depuis ce point de départ, et en suivant le chemin, vais-je arriver au but que je me suis fixé ?*

Avant de commencer la vérification d'un itinéraire, l'agent doit d'abord avoir enregistré, dans une mémoire morte associative, la liste des étapes *départ→arrivée* qu'il connaît.

Ensuite, il simule : partant d'un point initial, il vérifie si, avec la connaissance qu'il possède du monde réel, il trouve l'itinéraire, i.e. il teste si, au bout de plusieurs dérivations, il peut atteindre un lieu but qu'il s'est donné.



16 - Au-dessus des séquenceurs quotidiens, on voit leur représentant : le séquenceur régulier

Simulation au moyen de séquenceurs réguliers ou quotidiens

Après avoir décrit comment effectuer une simulation au moyen des séquenceurs irréguliers, passons maintenant au séquenceurs réguliers.

Dans ce cours, en classant les séquenceurs au sein d'une arborescence, nous avons travaillé sur la relation $SR \leftrightarrow SQ$, et avons montré comment un séquenceur régulier peut représenter, être le père de plusieurs séquenceurs quotidiens.

Maintenant nous continuons de présenter le mécanisme de la simulation, mais nous travaillons dans le cadre plus organisé des séquenceurs réguliers plongés dans le monde quotidien.

Introduction : quantifier l'évolution du monde observé, deviner les valeurs de ses états futurs

De façon générale, une simulation modélise l'évolution du monde réel sous l'action des agents qui le peuplent. Or le travail de description de l'état de départ du monde demande d'y reconnaître des modèles déjà connus. Ainsi, au travers de cette recherche, il introduit déjà de la standardisation, i.e. de la régularité.

De plus, ici nous choisissons de travailler avec des séquenceurs réguliers, alors ce choix introduit encore plus d'ordre. Ainsi nous travaillons au niveau ultime de représentation abstraite pour décrire le monde : nous cessons d'utiliser des

symboles classiques pour traiter seulement des symboles numériques binaires. Nous en sommes arrivés à une démarche de quantification qui nous permettra de décrire le monde au moyen de nombres entiers.

Nous utilisons un séquenceur régulier numérique : partant du résultat de l'évaluation du système, en appliquant la formule récurrente qui caractérise la fonction de ce SR, nous représentons un histogramme, i.e. nous traçons l'évolution séquentielle de la courbe des quantifications effectuées sur le micro-monde.

Voyons maintenant cela en détail.

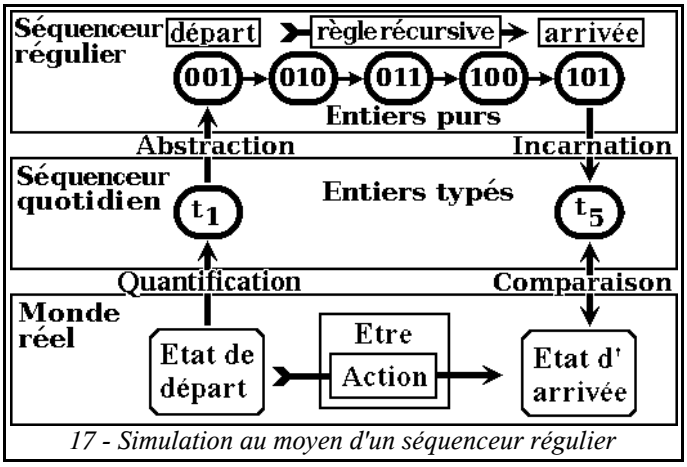
Le travail de modélisation consiste en une triple démarche d'abstraction

Dans une simulation effectuée au moyen des séquenceurs réguliers et quotidiens, la modélisation induit une triple démarche d'abstraction.

1) D'abord une démarche de modélisation

Correspondant à la démarche classique de simulation, on trouve d'abord un travail de modélisation : le cogniticien observe le monde réel et décrit l'état du micro-monde quotidien. Puisque il travaille avec un séquenceur, cette description se limite à la fourniture d'un unique symbole numérique qui quantifie son état. Note : il est typé car il appartient à la vie de tous les jours (temps, prix, difficulté, taille, surface, poids, production...).

Cette modélisation est effectuée sur des entiers : donc ici, cette simulation réalisée au moyen d'un séquenceur régulier, atteint une position ultime de régularité, et pour décrire l'état de départ du système elle utilise la forme numérique : au moyen d'une fonction d'évaluation du système, elle fournit une quantification typée. Dans l'exemple ci-contre, on peut voir l'entier typé t_1 (pour *valeur au temps 1*).



17 - Simulation au moyen d'un séquenceur régulier

2) Ensuite une démarche d'analyse : identifier le s. régulier correspondant, trouver la fonction récurrente

Le cogniticien observe les transformations que subit le monde réel. Dans une démarche d'analyse et d'abstraction, il recherche le séquenceur quotidien et la fonction récurrente correspondante, qui, à chaque cycle, quantifie l'évolution du monde.

En observant ce SQ, il identifie son représentant, son père ; ainsi il dispose du séquenceur régulier qui le modélise au moyen d'une fonction calculée sur des symboles très abstraits ; car évidemment, ici il obtient une fonction numérique sur des entiers binaires.

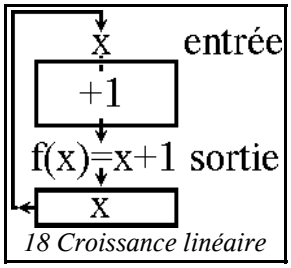
3) Enfin une démarche d'abstraction

Dans une démarche d'abstraction, il délaisse le type *t* du domaine et transmet, seulement au séquenceur régulier, l'entier non typé *1* et la fonction numérique à évaluer itérativement.

Transition d'état par application de règle de réécriture, effectuée au niveau du séquenceur régulier

Le système dispose maintenant des deux éléments nécessaires pour quantifier chaque étape de l'évolution de son domaine. Au niveau du séquenceur régulier, sachant la formule récurrente qui le caractérise et la valeur numérique initiale de départ, il effectue la simulation proprement dite : la transition d'états.

Ainsi, partant de cet entier *1*, valeur de l'état initial du micro-monde, l'informaticien simule son évolution en déroulant une séquence : tant que c'est possible, il applique itérativement au registre mémoire, la fonction numérique récurrente qui effectue le calcul, et fournit un nouvel état quantifié du micro-monde, i.e. une nouvelle valeur binaire.



18 Croissance linéaire

Ici, puisque nous sommes dans le cadre d'un traitement numérique, la chose se ramène à calculer les valeurs du système qui évolue à chaque impulsion de chargement du registre mémoire.

Dans l'exemple traité, on obtient la suite : *10, 11, 100, 101* ; qui constitue un histogramme des valeurs prises par le système.

L'arrêt du système

Puisque le système applique itérativement une fonction numérique récurrente, une fois lancé, il continue : il est très autonome et les conditions d'arrêt sont moins nombreuses. Alors, il faut aussi prévoir un regard extérieur qui à chaque cycle teste une condition d'arrêt pour stopper le processus. Dans l'exemple traité ci-dessus, le système s'arrête quand son état numérique est *101*.

On obtient l'état d'arrivée du micro-monde

Quand la séquence de transition s'arrête, le système stoppe. On obtient l'état d'arrivée quantifié du micro-monde : il est réduit à un symbole numérique : un nombre binaire.

Retour au réel du quotidien

Après avoir quantifié le comportement abstrait du modèle, nous retournons ce résultat final à ses origines, i.e. nous le plongeons dans le monde concret, en lui rajoutant le type que nous avions enlevé à l'étape de généralisation (temps, prix, difficulté, taille, surface, poids, production...).

Remarque : le test et le recalage du modèle relèvent de l'apprentissage

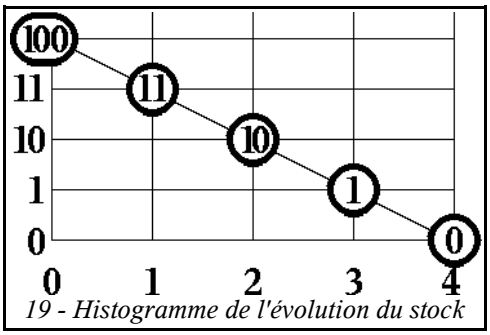
À ce stade, nous avons terminé le cycle de simulation proprement dit. Mais, ensuite, au vu du résultat calculé par la simulation, il se peut que nous passions à d'autres démarches : parfois, nous testons à nouveau le modèle, pour tirer des conclusions, afin de le recaler. Ces ajustements effectués sur lui, ne sont pas anodins. Ils constituent un fonctionnement nouveau en apprentissage, qui sera traité plus longuement à la fin du cours, dans la partie relative au traitement cognitif.

Exemple de traitement quantitatif : gestion de stock

Nous avons déjà décrit ce fonctionnement : dans un circuit logique combinatoire, nous câblons une fonction numérique. Le registre mémoire est initialisé à la valeur x_0 ; à chaque impulsion de l'horloge, il passe de l'état x à l'état suivant : x - k. Nous obtenons une ligne décroissante qui modélise la diminution du stock géré au fil du temps.

L'histogramme ci-contre montre la courbe obtenue pour les valeurs x_0=100 et k=1.

À l'issu de cette simulation, l'expérimentateur déduit qu'à l'étape 4 le stock est épuisé.



19 - Histogramme de l'évolution du stock

Application : Le séquenceur, déjà capable de prévoir le futur, et pourtant si simple

Après avoir présenté le séquenceur utilisé en simulation, ici nous le présentons dans une autre application : la prévision à long terme.

Le séquenceur est déjà capable de prévoir le futur

Quand le modèle servant à simuler est bien au point, on peut l'utiliser au sein d'agents capables d'anticiper. Dans un tel fonctionnement, quand la simulation dans le micro-monde s'arrête, le système fournit le futur état d'arrivée calculé du monde.

En le comparant avec le but visé, l'agent peut tirer des enseignements. Si la prévision est prometteuse, alors il décide de continuer l'action dans le monde réel, mais s'il considère qu'elle induit un problème, il peut décider de l'inhiber.

Par le biais de cette prévision à long terme, l'être vivant obtient une certaine emprise sur son environnement ; nous ne pouvons pas forcément conclure qu'il le maîtrise, mais nous pouvons déjà dire qu'il le subit moins.

Note : quel peut être l'intérêt d'une telle simulation ?

Le lecteur critique, qui connaîtrait le fonctionnement de la simulation classique en IA, pourrait formuler des réserves. En effet, le séquenceur ainsi utilisé en simulation fournit une information bien ténue à propos du monde simulé : elle est réduite à un unique symbole. Alors quel est son intérêt ?

Tout d'abord, le premier intérêt de ce séquenceur est qu'il est très simple. Il est déjà capable de prévoir le futur, pourtant, l'implémentation électronique ou informatique de ce processus si précieux demeure extrêmement légère. Elle se réduit à un registre temporaire rebouclé sur lui-même au moyen d'une mémoire morte associative ou d'une fonction numérique récurrente. Cependant il faut reconnaître que nous n'avons pas précisé comment implémenter une structure de contrôle. En effet, il nous reste à expliquer le départ et l'arrêt du traitement, i.e. comment sont implémentés l'initialisation et l'arrêt de notre montage.

On peut encore répondre à cette critique, en plaidant qu'en IA on utilise avec profit, une structure qui pourtant est encore moins performante. En effet, il existe une planification fournissant elle aussi un unique symbole, mais limitée à une profondeur unaire : elle prévoit seulement le futur proche. C'est l'anticipation immédiate, qui cependant présente un intérêt. Dans les SMA (système multi agents) elle évite à l'agent des dangers imminents et mortels : la chute dans un précipice, la menace d'un prédateur à l'affût... Alors si le séquenceur ne fournit lui aussi qu'une information réduite à un

symbole, par contre il prévoit à long terme, i.e. à profondeur N, à échéance de plusieurs étapes. Alors, il est donc encore plus intéressant que l'anticipation immédiate.

Enfin, il faut aussi mettre en avant le contexte. Certes cette planification au moyen d'un séquenceur est piteuse comparée à la planification classique de l'IA symbolique, mais elle trouve sa place dans un contexte de vie artificielle, quand des agents de deux espèces différentes sont en concurrence. En effet, le mutant qui le premier implémente cette nouvelle aptitude se trouve avantagé : il augmente ses chances de survie et de reproduction. Ainsi, il serait privilégié par le mécanisme de sélection naturel.

Conclusion : efficace et bon marché

En IA, et en particulier en VA, il existe des contextes où le séquenceur en simulation présente déjà bien des intérêts pour l'agent qui l'implémente.