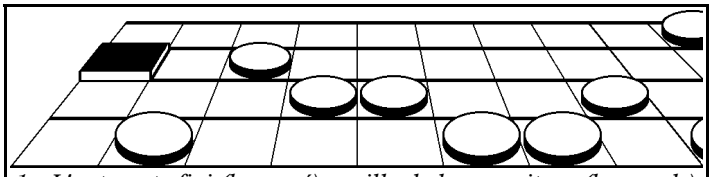
Exemple d'application de l'automate fini : Un agent réactif cueilleur

Étude d'une application concrète d'AF

Après avoir étudié l'automate fini (AF), maintenant nous étudions l'exemple d'une application concrète : l'implémentation d'un agent réactif cueilleur sur une route à 4 voies, au moyen d'un AF.



1 - L'automate fini (le carré) cueille de la nourriture (les ronds) distribuée aléatoirement sur une route à 4 voies

En marchant le long d'une route, un agent cueille de la nourriture distribuée aléatoirement

Description du monde

Le monde où l'agent recherche sa nourriture est simplifié au maximum : Il se réduit à une route à 4 voies. Selon un point de vue informatique, nous la représenterons par une grille de 4 lignes (les 4 voies) et de 10 colonnes (les 10 premiers kilomètres), ce qui donne 40 cases.

Description de la nourriture

Notez que la nourriture est distribuée régulièrement. Tous les kilomètres, une nourriture est posée dans une des voies choisie aléatoirement.

Graphiquement, cette distribution est matérialisée ainsi : Dans une case, la présence de nourriture est représentée par un 1 ; son absence, par un 0.

Finalement, sur la grille, dans chaque colonne, une seule case reçoit de la nourriture.

Se donner un exemple de monde

Afin de fixer les idées, voici donc les 10 premiers kilomètres de cette route à 4 voies avec laquelle nous allons travailler. Elle est donc découpée en 10 tronçons de 1 km, i.e. en 10 colonnes.

	km1	km2	km3	km4	km5	km6	km7	km8	km9	km10
ligne 1	0	0	0	0	0	0	1	0	0	0
ligne 2	0	0	0	1	0	0	0	0	0	0
ligne 3	1	1	0	0	0	0	0	1	1	0
ligne 4	0	0	1	0	1	1	0	0	0	1

2 Distribution de la nourriture le long des 10 premiers km de la route à 4 voies

Description générale de la simulation

En progressant, l'agent se déplace de colonne en colonne, et lorsqu'il avance d'un kilomètre, il peut monter d'une ligne, rester au même niveau, ou descendre.

Sur l'illustration de droite, il est dans la ligne 3 et le monde qu'il perçoit est matérialisé par un cône de vision. On comprend bien que, depuis une colonne de rang 'n' il perçoit la totalité de la colonne suivante, de rang (n+1).

Quand il réussit à passer sur une case où il trouve de la nourriture, il la mange, sinon il jeûne.

	km1	km2	km3	km4
ligne 1	0	0	·	0
ligne 2	0	0	·	1
ligne 3	1	1	a	0
ligne 4	0	0	·	0

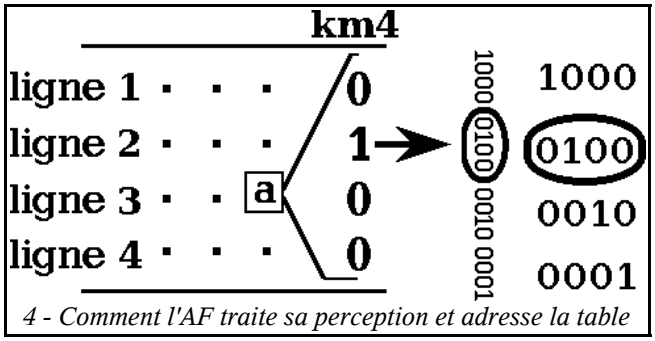
3 - Point de vue de l'automate au km3

Analysons plus finement le système

L'être perçoit le monde au moyen de son capteur de vision

Il scrute la route à 4 voies devant lui. On sait qu'à chaque kilomètre, une nourriture est distribuée. Ainsi il perçoit un capteur de 4 bits qui ne peut prendre que les 4 différents états suivants :

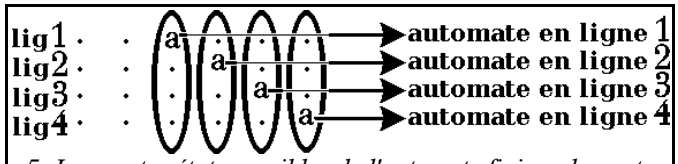
- 1000 Nourriture en ligne 1
- 0100 Nourriture en ligne 2
- 0010 Nourriture en ligne 3
- 0001 Nourriture en ligne 4.



4 - Comment l'AF traite sa perception et adresse la table

L'état de l'automate

Il est donné par la position de l'agent sur une des 4 voies de la route. Il peut donc être dans les lignes 1, 2, 3 ou 4, i.e. dans l'un des quatre états représentés ci-contre :



5- Les quatre états possibles de l'automate fini sur la route

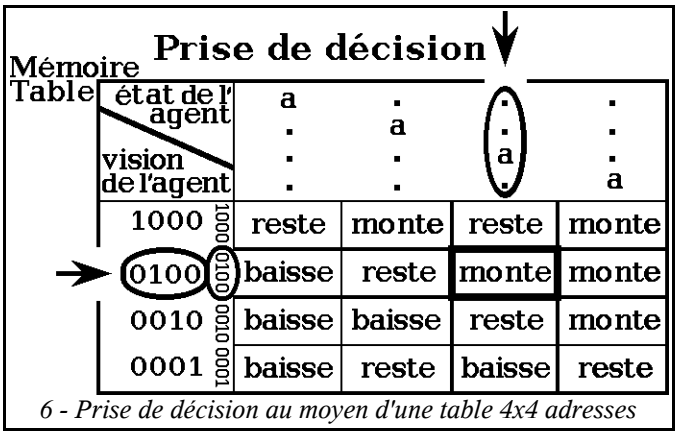
Comment l'automate prend-il une décision ?

L'automate peut se trouver dans 4 × 4 configurations différentes. Pour chacune de ces configurations, il doit décider que faire, c'est pourquoi il doit lire, dans une table de 16 adresses, l'action à effectuer. Elle est mémorisée dans chacune des cases de cette table.

Les seules actions qu'il peut exécuter sont :

Pour un déplacement de 1 km, il peut au maximum changer une fois de voie. Il peut donc :

- ↗ Monter d'une ligne,
- → Rester dans la même ligne,
- ↘ Baisser d'une ligne.



6 - Prise de décision au moyen d'une table 4x4 adresses

La solution repose sur une table à deux entrées

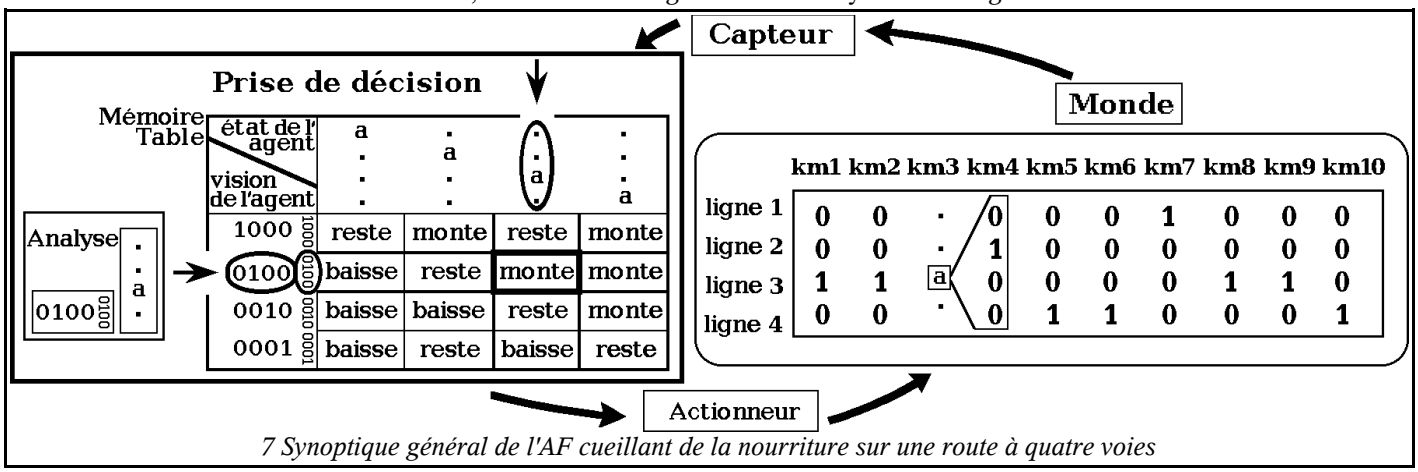
C'est une table de 4 × 4 = 16 cases, où chaque case contient une action choisie parmi les trois suivantes : monte, reste ou baisse.

Voici donc la solution que l'ordinateur devra exécuter. On vérifie avec un peu de bon sens que :

Si la nourriture est trop au-dessus : Il ne bouge pas.
Si la nourriture est juste au-dessus : Il monte de un cran.
Si la nourriture est au même niveau : Il ne bouge pas.
Si la nourriture est juste au-dessous : Il descend d'un cran.
Si la nourriture est trop au-dessous : Il ne bouge pas.

Agent ou automate

Dans le cadre de cette simulation, il faut bien distinguer entre deux systèmes : L'agent et l'automate.



7 Synoptique général de l'AF cueillant de la nourriture sur une route à quatre voies

L'agent seul

Notez que l'agent, pris tout seul, est réactif : Sa conduite est dictée par une table à deux entrées. Ainsi il ressort de la logique combinatoire.

Le système agent + grille

Mais le système agent + grille constitue un automate. Son état est caractérisé par la position de l'agent sur la route à 4 voies : Il ressort donc de la logique séquentielle.

Conclusion

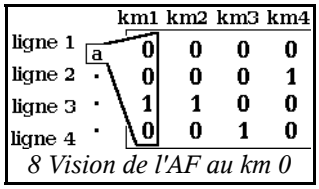
Avec ce petit automate nous touchons du doigt la notion très importante de transition d'état :
- Le système est dans un certain état interne.
- Il capte des variables extérieures provenant du monde qui l'entoure, et les analyse.

- Regardant cet état interne, et ces variables externes, il prend une décision d'action en se référant à une mémoire.
- Il agit suivant cette décision et modifie l'état du monde. Ainsi, il se retrouve lui-même dans un autre état.

Et le processus recommence…

Plus concrètement, traçons le parcours de l'agent sur les 10 km de cette route à 4 voies

Regardons fonctionner notre automate sur cet exemple



8 Vision de l'AF au km 0

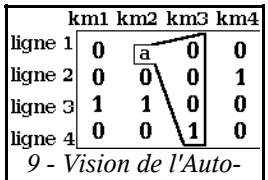
État de départ : L'agent est au km0

Choisissons d'introduire l'agent, verticalement au km0 (kilomètre 0), et horizontalement dans la ligne 1 : Son état de départ est donc celui dessiné ci-contre :

Analysons la première transition d'état (km0→km1)

La nourriture est dans la ligne 3, la vision de l'agent est "0010", la table indique l'action "reste" (dans la même voie), donc l'agent avance, il change de colonne et passe au km1, mais il ne change pas de ligne.

Analysons la deuxième transition d'état (km1→km2)



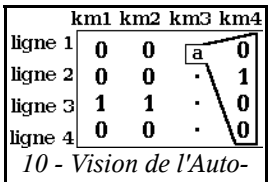
9 - Vision de l'Automate Fini au km 2

La deuxième transition d'état se déroule comme la première : L'agent avance au km2 sans changer de ligne.

Analysons la transition 3 (km2→km3)

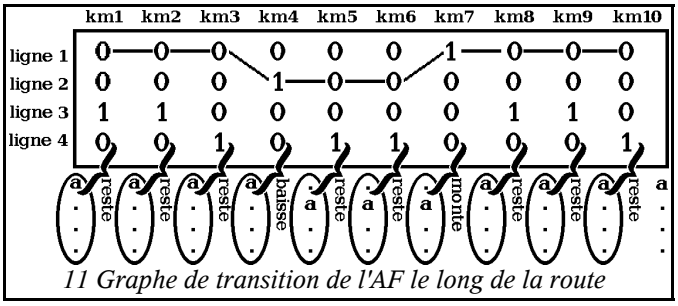
Cette fois-ci la nourriture est dans la ligne 4, la table indique de rester dans la même ligne. L'agent avance, encore sans changer de voie.

Analysons la transition 4 (km3→km4)



10 - Vision de l'Automate Fini au km 3

L'agent est au km3, il voit de la nourriture juste au-dessous de lui (0100). La table indique de passer dans la ligne au-dessous. Il descend à la ligne 2, et trouve de la nourriture : Il peut manger.



11 Graphe de transition de l'AF le long de la route

Graphe de transition

Ensuite, les kilomètres suivants du parcours se déroulent en appliquant classiquement les règles données par la table. Arrivé au km10, on obtient le graphe de transition ci-contre (à gauche).

Nombre de repas

Finalement, au cours de ce voyage de 10 km, l'agent mange deux fois.

Faire le lien entre cet agent cueilleur de nourriture et le cours d'I.A.

Introduction

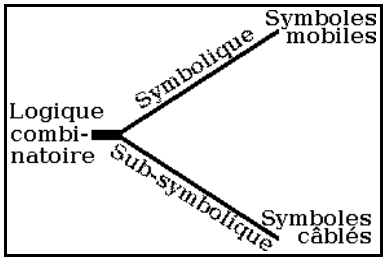
Avec cet agent qui cueille de la nourriture sur une grille de cases, nous avons étudié un modèle très simple d'interaction avec le monde. Le moment est venu d'aller plus loin dans nos considérations et de tirer des conclusions fructueuses pour notre cours d'I.A..

Situer cette simulation parmi les mouvances de l'I.A.

Introduction

Maintenant, c'est le moment de conforter notre savoir. Nous allons le mettre en application afin de classifier cet exemple. Ainsi ce travail sera pour nous l'occasion de conforter notre savoir.

Quelle représentation des connaissances



La table de décision, qui détermine l'action à effectuer

Elle est clairement symbolique car elle retourne les trois symboles suivants : monte, descend et reste.

La représentation du monde

Elle se fait au moyen d'une table qui retourne 1 ou 0. Ainsi, elle appartient à la logique binaire combinatoire, qui peut se situer entre les deux extrémités et peut ensuite basculer, soit vers le traitement sub-symbolique, soit vers le traitement symbolique.

Une représentation des connaissances pas si symbolique qu'il y paraît !

En effet, on pourrait se hâter de conclure que nous sommes en présence d'un système symbolique. Pourtant ce dernier n'est pas si clairement symbolique que ça ! Ceci pour deux raisons : d'abord une table peut facilement se réaliser au moyen d'un réseau de neurones (muni en sortie d'un étage de comparateurs pour fournir des niveaux logiques) ; et ensuite, on ne trouve pas dans ce cueilleur, un traitement lourdement symbolique, qui manipule des symboles mobiles[1] comme le fait par exemple un ordinateur, surtout au niveau de son bus de données.

Ce système est non-intentionnel

Si, d'abord nous focalisons uniquement sur la structure de l'agent, nous savons qu'il est seulement réactif, donc il n'est pas intentionnel. Ensuite, si nous considérons le système dans sa totalité, nous sommes en présence d'un automate qui change d'état quand l'agent avance. Mais nous sommes loin d'un agent qui force un drapeau interne qu'il ré-interprète ensuite pour déterminer son comportement. Le système entier, lui aussi, n'est pas intentionnel.

Système déterministe ?

Là encore, il faut mettre des nuances : si on se place du point de vue de l'agent, on peut penser qu'il est déterministe car son comportement est régit par une table remplie exhaustivement. Mais si on si on regarde les choses globalement, on constate, qu'en fait, il est victime d'un effet d'horizon : il ne voit pas plus loin que le bout de son nez, il ne possède pas une connaissance totale du monde. Ceci se traduit par un comportement qui n'est pas optimal. Dans ce cas on dit que l'agent exécute une *stratégie* (qui, par définition, est indéterministe) et non pas une tactique (qui, par définition, est déterministe).

Apprentissage ?

Non. Le système, tel que nous l'avons décrit, n'effectue pas d'apprentissage.

En conclusion

Ce système peut être à la fois symbolique et sub-symbolique, il est non déterministe et non intentionnel.

Prise de décision

Mémoire Table

état de l'agent / vision de l'agent	a . . .	. a . .	. . a .	. . . a
1000	reste	monte	reste	monte
0100	baisse	reste	monte	monte
0010	baisse	baisse	reste	monte
0001	baisse	reste	baisse	reste

12 - Prise de décision au moyen d'une table 4x4 adresses

Faire de l'apprentissage (dans le même cadre)

En restant dans le cadre de cet automate qui cherche de la nourriture sur une route à quatre voies, nous allons traiter un petit exemple d'apprentissage, ce qui constitue une approche confortable de cette démarche.

Effectuer un apprentissage en remplissant la table

Il est facile d'effectuer un apprentissage en remplissant la table 4 x 4, au moyen de la force brute de l'ordinateur. Pour faire cela, on teste la justesse de la programmation de la table en plaçant notre automate sur une route où de la nourriture est distribuée aléatoirement.

Voici la plus précisément la démarche d'apprentissage :

Un registre *NbreMaxDeRepas* mémorise le meilleur résultat obtenu, i.e. le plus grand nombre de repas effectués par l'agent lors du meilleur essai de comportement effectué jusqu'à maintenant.

---

1 En toute rigueur, au lieu de parler de *symbole mobile*, nous devrions plutôt dire *instance de symbole mobile*. Ainsi par exemple, au lieu d'énoncer : *le symbole 5*, qui est dans l'accumulateur, transite sur le bus de données et est rangé en mémoire ; nous devrions plutôt utiliser le terme *instance du symbole 5*.

On se donne une route à 4 voies où de la nourriture est distribuée aléatoirement. On la choisit suffisamment longue pour obtenir un lissage statique du résultat (de 300 à 1000 km).

Tant que :

On n'a pas essayé tous les remplissages possibles de la table...

Faire :

Générer le remplissage suivant.
Évaluer, sur la route à 4 voies, la qualité du comportement qu'il induit chez l'automate.
Si ce résultat est > que *NbreMaxDeRepas*
alors
1) Le conserver dans *NbreMaxDeRepas*.
2) Mémoriser le remplissage de la table
Fin de alors
Fin de Faire.

Conclusion :

Le système s'arrête quand toutes les remplissages possibles ont été testées.
Finalement on récupère le premier remplissage qui a donné le meilleur résultat.

Évaluer la combinatoire de la table

Attachons-nous maintenant à dénombrer les remplissages potentiels de cette table qui dicte le comportement de l'agent. Elle est de la forme 4x4 : elle est adressée par deux fois quatre données. Elle possède donc 16 cases. Chacune de ses cases peut retourner 3 valeurs : monte, descends et reste. Ainsi le nombre de ses remplissages possibles est de 3^{16}= 43 046 721.

Ne pas changer le cadre de cette simulation

Avant de passer à une démarche de quantification, précisons ce que ce titre signifie :
- L'agent conserve la même perception : Il ne voit que la case devant lui.
- L'agent conserve les mêmes actionneurs qui permettent seulement de monter, descendre et demeurer au même endroit : Il n'est pas capable de monter ou descendre plusieurs lignes à la fois.

Trouver une table plus efficace

Sans changer le cadre de l'exercice, on peut trouver une table plus efficace, que celle décrite au début du polycopié. La méthode existe, mais elle n'est pas précisée ici, car elle donne lieu à un exercice pour un devoir.

Travailler en apprentissage (en changeant le cadre)

Que peut-on changer au cadre ?

La faculté de mouvement

Quand il se déplace d'un km, le mouvement de translation de l'agent n'est plus limité à un déplacement unitaire : Il peut traverser plusieurs lignes à la fois. Pour commencer, disons qu'il peut traverser deux voies. Ainsi son mouvement s'étend à 5 possibilités : monter de deux voies, monter d'une voie, rester, descendre d'une voie et descendre de deux voies.

La perception

Jusqu'ici, nous avons pris un agent myope qui ne voit pas plus loin que le bout de son nez. Choisissons maintenant un automate qui perçoit mieux son monde. Par exemple, il peut voir les deux premiers kilomètres des voies devant lui.

Évaluer la combinatoire dans les différents cas où on change le cadre

Évaluer la combinatoire, si on change la faculté de mouvement

Si l'agent conserve la même faculté de vision, mais peut faire 5 actions : Monter de deux voies, monter d'une voie, rester, descendre d'une voie et descendre de deux voies ; alors il peut toujours être dans une des 4 voies. La table demeure de la forme 4 x 4, elle possède 16 cases, mais le nombre de ses remplissages possibles devient $5^{16} = 3^{16}$ x $(5/3)^{16}$. Ainsi la combinatoire initiale est multipliée par $(5/3)^{16} = 1{,}6667^{16} = 3544{,}7$.

Évaluer la combinatoire, si on change la perception

Si l'agent conserve les 3 mêmes moyens d'action (monter, rester et descendre), mais peut voir les deux premiers kilomètres ; alors il peut toujours être dans une des 4 voies. La table devient de la forme 8 voies x 4 voies, alors elle possède donc 32

cases et le nombre de ses remplissages possibles devient $3^{32} = 3^{16} \times 3^{16}$. Ainsi la combinatoire initiale est multipliée par $3^{16} = 43\ 046\ 721$. Ce n'est plus du tout pareil.

Évaluer la combinatoire si on change à la fois la faculté de mouvement et la perception

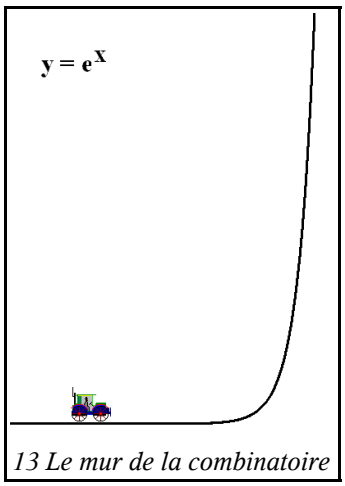
Si l'agent peut faire 5 actions et voir deux kilomètres devant, alors il peut toujours être dans une des 4 voies. La table devient de la forme 8 voies x 4 voies = 32 cases, alors le nombre de ses remplissages possibles devient $5^{32} = 5^{16} \times 5^{16} = 3^{16} \times (5/3)^{16} \times 5^{16} =$ Ainsi, comparée à cette de la solution initiale, la combinatoire est multipliée par $5^{16} \times (5/3)^{16} = 43\ 046\ 721 \times 3544{,}7$. Aïe, ça devient énOOOrme !

Conclusion

Ne boudons pas notre plaisir, l'exemple de cette application si simple nous a permis de toucher du doigt plusieurs domaines de l'IA. Tant que c'est possible, il faut en profiter, car les choses se compliquent rapidement. En effet, si on travaille avec des tables, il faut faire très attention : quand le nombre de paramètres d'entrée augmente, la combinatoire explose rapidement.

Le problème de l'explosion combinatoire

Nous venons donc de butter sur la combinatoire. il constitue un problème classique de l'IA. Mais ses conséquences sont désastreuses pour nous : la combinatoire augmente rapidement, au point que très vite, la force brute de l'ordinateur ne peut résoudre nos problèmes.



13 Le mur de la combinatoire

Des solutions au problème de l'explosion combinatoire

Dans la suite du cours, nous allons étudier au moins 3 solutions pour palier ce problème :

La modularité :

Éclater le traitement à effecteur vers un synoptique étagé en N couches verticales.

La programmation déclarative :

Introduire les variables, pour éclater la table en plusieurs règles.

Les langages objets :

Éclater le traitement à effectuer vers plusieurs sous-traitements.

Mais demain est un autre jour !