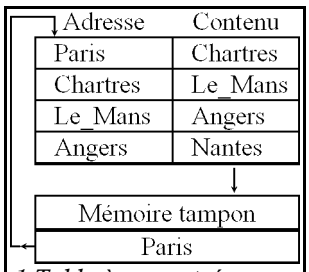
Systèmes Formels : l'automate fini (AF)

Avant d'étudier l'automate fini, faisons un retour sur le séquenceur

Le séquenceur est constitué d'une table à une dimension, utilisée avec un registre tampon

Le registre tampon introduit la dimension temporelle : on passe au séquentiel

À une table à une entrée, on ajoute une mémoire (un registre tampon) et on obtient un montage séquentiel, qui change d'état à chaque impulsion qu'on lui fournit (à chaque fois qu'on commande à la mémoire tampon de se charger). Ainsi on obtient un montage séquentiel.



1 Table à une entrée montée avec un registre pour construire un séquenceur

Expliquons le schéma ci-contre

Dans cet exemple, la table reçoit en entrée l'adresse *Paris*. En conséquence, elle fournit en sortie la donnée *Chartres*, qui se présente en entrée du registre tampon. Quand on lui en donne l'ordre, cette mémoire enregistre ce nouvel état, qui ensuite remonte, pour adresser à nouveau la table. Alors cette dernière reçoit *Chartres* en entrée, et donc fournit *Le_Mans* en sortie.

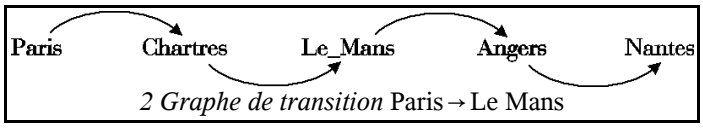
Le système ne changera que quand la mémoire tampon enregistrera cette nouvelle donnée. Ainsi, pas à pas, la séquence se déroule jusqu'à *Nantes*, état d'arrêt du système.

Graphe de transition

L'illustration ci-contre représente un graphe de transition

On peut utiliser un graphe de transition pour représenter les différents états par lesquels passe le système.

Remarque : les graphes sont définis plus largement dans un paragraphe à suivre.



2 Graphe de transition Paris→Le Mans

Étiqueter les flèches qui relient les états

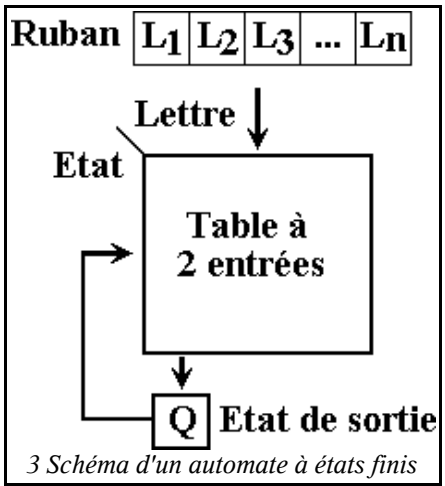
Ce graphe s'obtient en dessinant des noeuds correspondant aux états par lesquels transite le système. Ensuite, on les relie par des flèches, i.e. des liens orientés. Cependant, avec les séquenceurs, il est difficile de trouver avec quel symbole étiqueter ces dernières. En effet, il n'y a pas de raison d'utiliser l'état de départ plutôt que celui d'arrivée ; et, mettre les deux (départ-arrivée) devient lourd. Alors les flèches demeurent sans nom.

Rappels à propos de l'automate fini (automate à états finis)

Fonctionnement

Dans ce montage la dimension séquentielle apparaît en deux endroits :

1) D'abord, l'utilisation d'un tampon, pour reboucler la sortie de la table 2D sur une de ses entrées, introduit la dimension séquentielle dans le montage : 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 l'état courant du système. Immédiatement, cet état remonte pour adresser une des deux entrées de la table.



3 Schéma d'un automate à états finis

2) Au même moment, cette impulsion, fournie au ruban de symboles en entrée, le fait avancer d'un cran.

Simplicité

Selon le point de vue du synoptique de câblage, comme le séquenceur, l'automate fini présente une circuiterie extrêmement simple.

Cependant, il ne faut pas se tromper, selon le point de vue de l'IA, quand le nombre de symboles reçus en entrée augmente, la complexité (la taille) de sa table croît très vite. De plus, du point de vue *remplissage de table*, le nombre de programmations acceptées par cette dernière augmente encore plus vite.

Description mathématique

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.

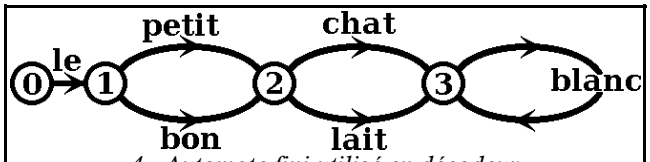
Le ruban ouvre l'automate sur l'extérieur

Selon un point de vue épistémologique, l'utilisation d'un ruban en entrée sert de capteur et ouvre l'automate sur l'extérieur. Ce dernier n'est plus solipsiste, il devient sensible au monde. Ainsi, placé devant une alternative, il doit prendre une décision. Il doit choisir sa destination entre deux états possibles. Illustrons ceci au moyen d'un graphe de transition.

Le graphe de transition

Quel est l'intérêt d'un graphe de transition

Voyons maintenant, de façon plus détaillée, une méthode graphique pour visualiser les changements d'états effectués par un automate : il s'agit du graphe de transition. Voilà ci-dessus un exemple, correspondant à un analyseur syntaxique, que nous détaillerons dans la suite de ce chapitre.



4 - Automate fini utilisé en décodeur

Commençons d'abord par la notion de réseau

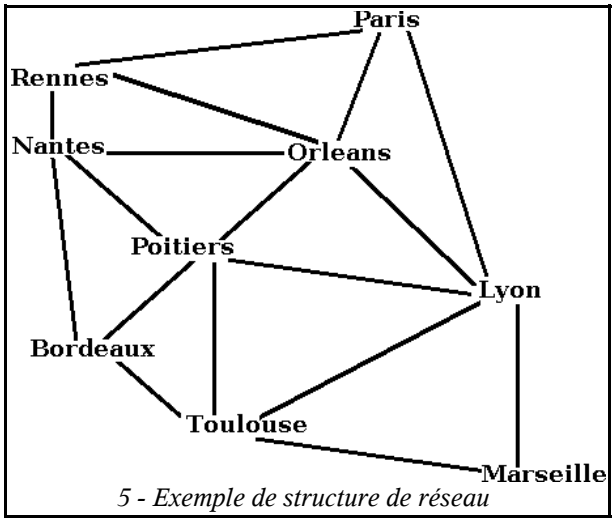
Pour mieux comprendre cette structure, énonçons quelques exemples classiques et quotidiens. Ce sont : le réseau routier, le réseau électrique et le réseau ferroviaire.

Définition d'un réseau/graphe

Un réseau est une structure constituée de nœuds (Dans le cadre du réseau ferroviaire : les gares), reliés par des arcs (Les voies de chemin de fer). Réseau et graphes sont équivalents, mais en mathématique on parle plutôt de graphe.

Définition d'un réseau orienté

Un réseau/graphe orienté est constitué de nœuds (Dans le cadre du réseau routier d'une ville : les places et carrefours), reliés au moyen d'arcs orientés par des flèches. (En effet, dans une ville, les trajets sont orientés, car parfois la circulation s'effectue en sens unique).



5 - Exemple de structure de réseau

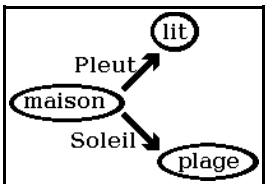
Dans un graphe de transition, les flèches qui orientent les arcs indiquent dans quel sens se font les transitions.

Le graphe de transition de l'automate : une utilisation informatique du graphe

Introduction

Un automate fini, qui parcourt un graphe de transition, constitue une illustration typique de la transition d'état.

Étiqueter les flèches qui relient les états



6 Les symboles Pleut *et* Soleil *étiquettent les transitions*

En informatique, le graphe de transition d'un automate s'obtient en deux temps :

D'abord on dessine les noeuds correspondant aux états par lesquels transite le système. Dans le dessin ci-contre ils sont écrits en minuscules, et représentés entourés par un ovale.

Ensuite, on les relie par des flèches. Avec l'automate fini, il devient facile d'étiqueter ces dernières, pour cela, on prend le symbole du ruban (Symbole en minuscules, introduit par une majuscule).

Utilisons ces conventions de représentation pour illustrer l'alternative

Avec l'alternative, le graphe de transition de l'automate fini se complexifie. Regardons l'exemple ci-dessus : il présente un embranchement (à partir de l'état 'maison').

Au départ, le système est dans l'état *maison*, l'A.F. observe le ruban pour savoir que faire. S'il perçoit le symbole *Pleut* alors il passe dans l'état *lit*. S'il perçoit le symbole *Soleil*, il passe dans l'état *plage*. Enfin, s'il ne perçoit aucun de ces deux symboles, il demeure sur place.

Remarque : le reste de la table de l'automate n'est pas précisé. En effet, on focalise seulement sur la patte d'oie de l'alternative.

état\ruban	***Pleut***	***Soleil***
maison	lit	plage
lit	.	.
plage	.	.

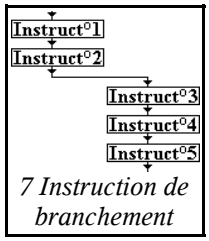
Chez l'automate, les arcs sont orientés et étiquetés

Quand le système est dans l'état *maison*, il se trouve placé devant les deux directions de l'alternative, et devant les deux branches d'une patte d'oie (devant l'entrée du delta d'un fleuve).

L'arc qui permet de transiter de l'état *maison* vers l'état *lit* est étiqueté par le symbole *Pleut* du ruban. Et celui qui permet de transiter de l'état *maison* vers l'état *plage* est étiqueté par le symbole *Soleil*.

Situons la puissance de traitement de l'automate fini

Comme une instruction de branchement de l'assembleur, ou le *GoTo* du basic ou du C[1]



7 Instruction de branchement

L'introduction du ruban permet à l'automate fini d'exécuter des instructions conditionnelles de branchement. Si on représente ses transitions d'état, on constate qu'ainsi, il peut parcourir des graphes présentant environ la même complexité que les organigrammes des langages non-structurés.

Par exemple, dans l'illustration ci-contre, on effectue un branchement à l'instruction 3 qui se situe ailleurs dans la mémoire. On appelle cela une rupture de séquence. Dans le cours sur les tables à une entrée, nous avons vu qu'on peut déjà exécuter ces ruptures au moyen d'un séquenceur. Dans ce cours sur les tables à deux entrées nous avons vu que l'automate fini autorise l'alternative ; ainsi ces ruptures peuvent, en plus, être conditionnelles.

Mais pas comme une instruction d'appel de sous-programme (langages structurés)

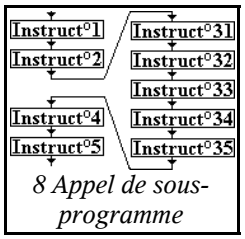
Cependant, il faut faire bien attention, l'automate récupère seulement la puissance de branchement d'un langage non-structuré, mais pas celle de l'appel de sous-programme (une des caractéristiques des langages structurés (pascal, C, lisp, python)).

En effet, avec un automate, comme avec l'instruction *GoTo*, on effectue un branchement *ailleurs*, mais, après l'instruction 5 (illustration ci-dessus), le programme continue en séquence, et le retour au programme appelant n'est pas fourni, c'est au programmeur de le gérer.

---

1 Dans 99,9% des cas l'instruction GoTo peut être évitée en C ! Attention, elle fait mauvais genre.

Au contraire, dans un langage structuré (illustration ci-contre), après l'instruction 2, le système effectue un appel de sous programme en se branchant à l'instruction 31. Ensuite, quand ce sous-programme est fini (après l'instruction 35), l'ordinateur ne continue pas en séquence, car le langage structuré génère le code pour retourner au programme appelant : l'ordinateur revient sur ses pas, et exécute ensuite l'instruction 4.

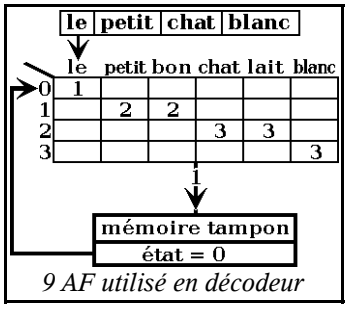


8 Appel de sous-programme

L'automate à pile pour parcourir des réseaux de transition récursifs

Puisque l'automate classique n'effectue pas de retour au programme principal, il existe un automate plus performant, dont la puissance de branchement est comparable à celle des appels de sous-programmes, c'est l'automate à pile. Il permet de traiter des réseaux de transition récursifs. Nous analyserons quelques-unes de ses applications un peu plus tard, mais seulement en étudiant les règles de production, ce qui simplifiera notre apprentissage.

Exemple d'automate fini pour décoder une phrase clé



9 AF utilisé en décodeur

Introduction

Étudions une application de l'automate fini utilisé pour reconnaître plusieurs phrases clés. En particulier, dans cet exemple, il reconnaîtra la séquence : *le petit chat blanc*.

Tout d'abord, voyons quels sont les différents composants de cet automate fini

La table

C'est une table à double entrée : elle est adressée verticalement par l'état de l'automate, et horizontalement par les symboles du ruban d'entrée : le, petit, bon, chat, lait, blanc. Elle est programmée par le remplissage de 6 de ses cases, au moyen des états 1, 2 et 3. Si on donne à la table un état de départ (état interne, symbole d'entrée) elle fournit l'état suivant de l'automate.

	le	petit	bon	chat	lait	blanc
0	1					
1		2	2			
2				3	3	
3						3

10 Table de l'AF décodeur

Le ruban externe

Il porte la phrase à reconnaître : *le petit chat blanc.*

La mémoire (ou encore registre) tampon

11 - Registre tampon de l'AF

Elle mémorise l'état dans lequel est le système. Au début du traitement, elle est initialisée à l'état de départ : *0*.

L'état final

Précisons que l'état final, qui marque le succès de la reconnaissance de la phrase, est *3*.

Le fonctionnement de l'automate

Deux index servent à adresser la table à deux entrées, ce sont l'état actuel de l'automate, et le premier symbole du ruban. Ainsi renseignée, la mémoire identifie le symbole qui se situe à l'intersection de la ligne et de la colonne adressées et le fournit en sortie. Ce futur état est présenté en entrée de la mémoire tampon, et il y entre lors de l'impulsion de transition.

Ensuite, à chaque impulsion le cycle recommence. Il se termine si l'automate parvient à un état terminal. Ici, il s'agit de l'état 3.

Faisons tourner ce système à la main

- Au départ l'état de l'AF. est *0*, et le premier mot (symbole) sur le ruban est *le*.
- Ainsi adressée verticalement par *0* et horizontalement par *le*, la table rend *1* qui devient le nouvel état.
- Adressée par 1 et *petit*, la table rend *2*. Donc le système passe à l'état *2*.
- Adressée par 2 et *chat*, la table rend *3*. Donc le système passe à l'état *3*.

- Adressée par *3* et *blanc*, la table rend *3*, qui est l'état final du système.

Cet automate fini a bien reconnu le groupe nominal *le petit chat blanc*.

Étude du fonctionnement de ce système, au moyen de son graphe de transition

Focalisons sur le graphe de transition de notre exemple d'automate

Le graphe

Le dessin ci-dessous représente un graphe car nous y trouvons bien les nœuds (les chiffres 0, 1, 2, 3), et les arcs orientés par des flèches. De plus, ces derniers sont étiquetés par des symboles (mots) : le, petit, bon, chat, lait, blanc.

Rappel des conventions :

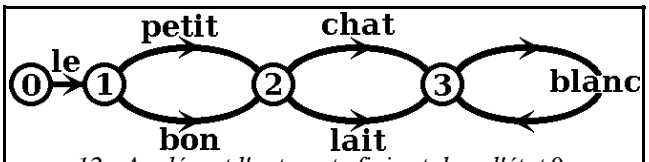
Pour cet automate, nous avons convenu que : l'état initial est '0', et l'état final '3'.

Étude d'un exemple, pour illustrer l'utilisation d'un graphe de transition ?

Dans cet exemple, le but est de reconnaître des groupes nominaux, au moyen d'un traitement graphique. Nous étudierons la reconnaissance du GN *le petit chat blanc.*

Au départ :

Notre automate est dans l'état *0*.



12 - Au départ l'automate fini est dans l'état 0.

Les symboles en entrée

Ce sont les symboles du ruban. Ils constituent le groupe nominal à reconnaître, qui est traité séquentiellement : on focalise successivement sur le premier symbole du reste du groupe nominal à traiter.

Étude d'un exemple : comment effectuer graphiquement les transitions d'état ?

Sur le graphe ci-dessus, on repère l'état actuel (interne) de l'automate : c'est *0.* Puis on cherche des flèches qui en sortent, permettant ainsi de quitter ce noeud : il existe une unique flèche étiquetée par *le*. Sachant que le symbole d'entrée est le premier mot du groupe nominal à traiter : ici *le*, on regarde s'il étiquette l'une d'elles. C'est OK !

Si c'est le cas, on quitte cet état *0* en suivant cette flèche jusqu'au prochain : *1*. Le système vient d'effectuer une transition. Et on recommence...

Si le groupe nominal n'appartient au langage à reconnaître, le système se bloque quelque part : l'ensemble des symboles sur le ruban d'entrée ne présente pas une des phrases clé à reconnaître. C'est l'échec de sa reconnaissance.

L'état final

Dans un automate, on déclare au choix, aucun, un ou des état finals.

Si l'automate réussit à arriver à un état final, et s'il ne reste plus de symbole sur le ruban d'entrée, alors on dit que le groupe nominal est reconnu : il appartient au langage reconnu par l'automate.

Sinon l'automate est bloqué, il s'arrête sur un état qui n'a pas été déclaré final. Ceci peut arriver, soit parce qu'il ne connaît pas le symbole d'entrée, soit parce que le ruban est entièrement lu.

Sur le graphe de transition, observons l'analyse de la phrase : *le bon chat blanc*

Conventions de notation

On utilise la notation suivante : " (symbole d'entrée, état interne) → nouvel_état ".

Séquence des transitions d'état

Au départ, le système est dans l'état 0. Au sortir de l'état 0, on trouve une flèche étiquetée par *le*.
Transition 1 : (le, 0) → *1*.
Transition 2 : (bon, 1) → *2*.

Transition 3 : (chat, 2) → *3*.
Transition 4 : (blanc, 3) → *3*.
À l'arrivée, le système est dans l'état *3* : l'automate a reconnu le groupe nominal !

En fait...

Voici les huit groupes nominaux reconnus par l'automate :

- *le bon chat blanc.*
- *le petit chat blanc.*
- *le bon lait blanc.*
- *le petit lait blanc.*
- *le bon chat.*
- *le petit chat.*
- *le bon lait.*
- *le petit lait.*

Et voici deux groupes nominaux qui ne sont pas reconnus par l'automate :

- *le chat noir* (l'automate demeure bloqué dans l'état 1).
- *le bon chien blanc* (l'automate demeure bloqué dans l'état 2).

Conclusion

Nous avons donc étudié un exemple très simple et très pur d'automate, et nous avons bien matérialisé les transitions d'état effectuées. Cette structure de transition d'état est naturelle, simple et elle est à la base de ce cours : nous la retrouverons bientôt.