# Systèmes Formels : l'automate fini (AF)

## 1) Avant d'étudier l'automate fini, revenons sur le séquenceur

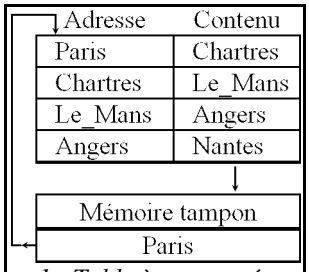
### 1.A) Le séquenceur = table à une dimension + registre tampon

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

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

À une table à une entrée, nous ajoutons une mémoire (un registre tampon) et nous obtenons un montage séquentiel, qui change d'état à chaque impulsion que nous lui fournissons (à chaque fois que nous commandons à la mémoire tampon de se charger).

Ce faisant nous changeons de registre de fonctionnement : à un montage combinatoire nous ajoutons la dimension temporelle. Ainsi nous obtenons un circuit séquentiel.




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

#### b) Expliquons le schéma ci-contre

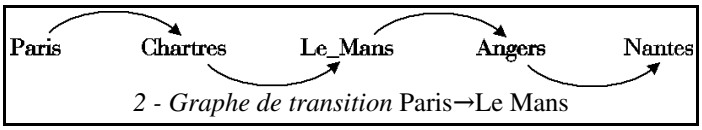
Dans cet exemple, la table reçoit en entrée l'adresse *Paris*. En conséquence, elle fournit en sortie la donnée *Chartres*, qui se présente en entrée du registre tampon. Quand nous lui en donnons l'ordre, cette mémoire enregistre ce nouvel état, qui sort à sa base et ensuite remonte, pour adresser à 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.

### 1.B) Graphe de transition

#### a) L'illustration ci-contre représente un graphe de transition

Nous pouvons utiliser un graphe de transition pour représenter les différents états par lesquels passe, transite le système.




*2 - Graphe de transition* Paris→Le Mans

Remarque : la notion de graphe est défini plus largement dans un paragraphe à suivre.

#### b) Étiqueter les flèches qui relient les états

Ce graphe s'obtient en dessinant des nœuds correspondant aux états par lesquels transite le système. Ensuite, nous les relions 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.

# 2) Rappels à propos de l'automate fini

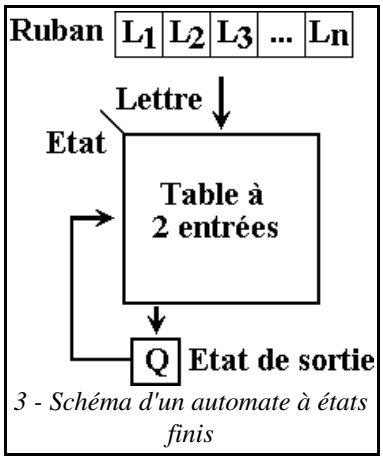
L'automate fini s'appelle aussi *automate à états finis*, et même *automate à états finis déterministe*.

## 2.A) 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.

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



*3 - Schéma d'un automate à états finis*

## 2.B) 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.

## 2.C) Description mathématique

Un automate fini déterministe A est un quintuplet $\{Q, \Sigma, \delta, q_0, F\}$ :
1) Q est un ensemble fini d'états[1].
2) $\Sigma$ 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 $\Sigma$ .
4) $q_0$ est l'état de départ du système.
5) F, inclus dans l'ensemble Q, est l'ensemble des états finals.

## 2.D) 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 quitte sa tour d'ivoire, il n'est plus solipsiste, il s'ouvre au monde et devient sensible[2].

Ainsi, placé devant une alternative, en fonction de son ruban, il doit prendre une décision. Il doit choisir sa destination entre deux états possibles. Illustrons ceci au moyen d'un graphe de transition.

---

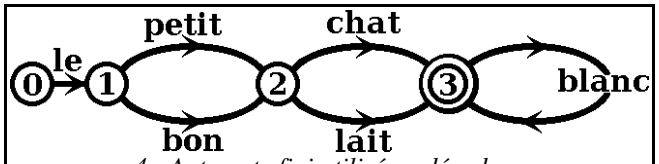
1 C'est d'ici que vient le nom de l'AF : automate à états finis (dont le nombre d'états est fini).

2 Symboliquement, il commence à percevoir l'extérieur au moyen de ses sens (vue, toucher, ouïe, goût, l'odorat).

# 3) Le graphe de transition

## 3.A) 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.

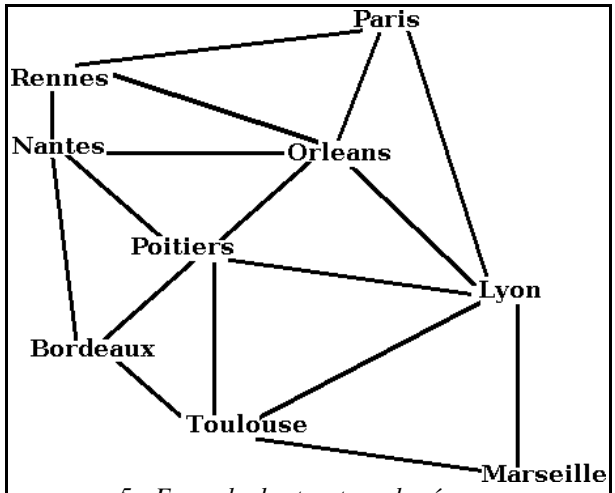


*4 - Automate fini utilisé en décodeur*

Voilà ci-dessus, un exemple, correspondant à un analyseur syntaxique, que nous détaillerons dans la suite de ce chapitre.

### a) Commençons d'abord par présenter la notion de *réseau*

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



*5 - Exemple de structure de réseau*

### b) 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 nous parlons plutôt de graphe.

### c) 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).

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

## 3.B) Une utilisation informatique du graphe de transition pour l'automate

### a) Introduction

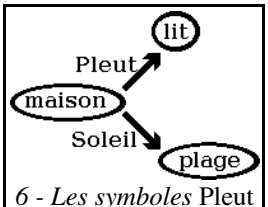
À la base, le graphe est un outil mathématique. En informatique, il s'avère pratique pour représenter le graphe de transition d'un automate.

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

### b) Étiqueter les flèches qui relient les états

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

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



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

Ensuite, nous les relions par des flèches. Avec l'automate fini, il devient facile d'étiqueter ces dernières, pour cela, nous prenons le symbole du ruban. Nous convenons d'écrire ce symbole en minuscules, mais en l'introduisant par une majuscule. Ex : *Pleut*, *Soleil*.

### c) 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'AF 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, nous focalisons seulement sur la patte d'oie de l'alternative.

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

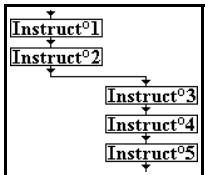
### d) Chez l'automate, les arcs sont orientés et étiquetés

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*.

# 4) Situons la puissance de traitement de l'automate fini

## 4.A) L'instruction de branchement de l'assembleur, le *GoTo* du basic ou du C[3]



*7 - Instruction de branchement*

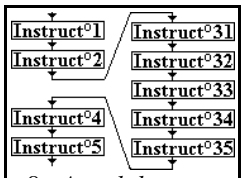
L'introduction du ruban permet à l'automate fini d'exécuter des instructions conditionnelles de branchement. Si nous représentons ses transitions d'état, nous constatons qu'ainsi, il peut parcourir des graphes présentant environ la même complexité que les organigrammes des langages non-structurés (basic, C, Fortran, Cobol).

Par exemple, dans l'illustration ci-contre, nous effectuons un branchement à l'instruction 3 qui se situe ailleurs dans la mémoire. Nous appelons cela une rupture de séquence. Dans le cours sur les tables à une entrée, nous avons vu que nous 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 avec l'automate, les ruptures du séquenceur, deviennent conditionnelles.

## 4.B) Mais pas 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 qui caractérise les langages structurés (pascal, C, lisp, python).

En effet, avec un automate, comme avec l'instruction *GoTo*, nous effectuons 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.



*8 - Appel de sous-programme*

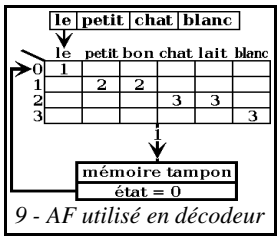
Au contraire, dans un langage structuré (illustration ci-contre), après l'instruction 2, le système effectue un appel de sous programme en se branchant à l'instruction 31. Ensuite, quand ce sous-programme est fini (après l'instruction 35), l'ordinateur ne continue pas en séquence, car le langage structuré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

## 4.C) 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.

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

# 5) Exemple d'automate fini pour décoder une phrase clé



*9 - AF utilisé en décodeur*

## 5.A) 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*.

## 5.B) D'abord, voyons quels sont les différents composants de cet automate fini

### a) 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 nous donnons, à la table, un état de départ (i.e. un état interne, et un 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*

### b) Le ruban externe

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

### c) 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*.

### d) L'état final ou état accepteur

L'état final, qui marque le succès, la reconnaissance de la phrase, est encore appelé *état accepteur*, car si l'automate s'y arrête, il accepte les données fournies sur le ruban.

Quand nous le dessinons sur un graphe, l'état accepteur est matérialisé par deux cercles concentriques. Précisons que, dans notre exemple, il correspond à *3*.

## 5.C) 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.

## 5.D) 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*.
- Donc, la table est adressée verticalement par *0* et horizontalement par *le*.
- Ainsi adressée, elle rend *1* qui devient le nouvel état. ‖ Donc le système passe à l'état *1*.
- 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*. ‖ C'est l'état final du système.

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

# 6) Étude de son fonctionnement, au moyen de son graphe de transition

## 6.A) Focalisons sur le graphe de transition de notre exemple d'automate

### a) 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.

### b) Rappel des conventions :

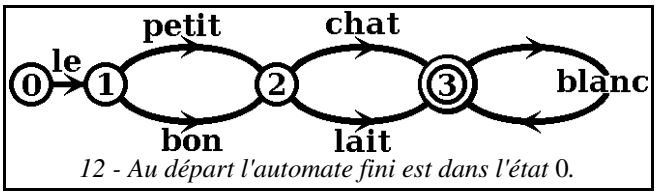
Pour cet automate, nous avons convenu que : l'état initial est *0*, et l'état final *3*.

## 6.B) 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.*

### a) Au départ :

Notre automate est dans l'état *0*.



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

### b) Les symboles en entrée

Ce sont les symboles du ruban. Ils constituent le groupe nominal à reconnaître, qui est traité séquentiellement : nous focalisons successivement sur la tête du ruban, i.e. le premier symbole du reste du groupe nominal à traiter.

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

Sur le graphe ci-dessus, nous repérons l'état actuel (interne) de l'automate : c'est *0.* Puis nous cherchons des flèches qui en sortent, permettant ainsi de quitter ce nœud : 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*, nous regardons s'il étiquette l'une d'elles. C'est OK !

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

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.

### d) L'état final/accepteur

Dans un automate, nous déclarons au choix, aucun, un ou des états finals/accepteurs.

Si l'automate réussit à arriver jusqu'à un état final, et s'il ne reste plus de symbole sur le ruban d'entrée, alors nous disons que le groupe nominal est reconnu, accepté : 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.

## 6.C) Sur le graphe de transition, analysons la phrase : *le bon chat blanc*

### a) Conventions de notation

Nous utilisons la notation suivante : *(symbole d'entrée, état interne)→nouvel_état*.

### b) Séquence des transitions d'état

Au départ, le système est dans l'état 0. Au sortir de l'état 0, nous trouvons 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 !

### c) 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 exemples de 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).

# 7) Automates finis non-déterministes et AF qui ne sont pas déterministes

## 7.A) Délaisser les aspects mathématiques et les AF non-déterministes

Dans ce cours d'IA pour une licence d'informatique, il n'est pas dans notre propos d'aller plus loin sur le plan mathématique en direction de la théorie des automates. Donc non ne traiterons, pas les automates finis non-déterministes.

Cependant nous traiterons une autre catégorie, qui est différente, mais qui s'appelle presque pareil : l'automate fini qui n'est pas déterministe.

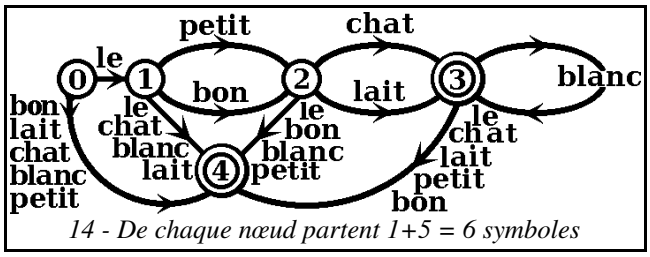
## 7.B) Un automate fini qui n'est pas déterministe

Si un informaticien s'attache à écrire un programme pour implémenter l'AF que nous utilisons pour reconnaître certains groupes nominaux, il risque d'écrire un programme qui plante quand nous lui demandons de reconnaître une séquence qui ne constitue pas un GN. Dans ce sens, nous pouvons dire que notre programme n'est pas déterministe, il peut arriver dans certains lieux d'où ils ne sortira pas : il ne terminera pas.

| | le | petit | bon | chat | lait | blanc |
|---|---|---|---|---|---|---|
| 0 | 1 | 4 | 4 | 4 | 4 | 4 |
| 1 | 4 | 2 | 2 | 4 | 4 | 4 |
| 2 | 4 | 4 | 4 | 3 | 3 | 4 |
| 3 | 4 | 4 | 4 | 4 | 4 | 3 |

*13 - Table de l'AF avec traitement des échecs/erreurs*

Or, en tant que programmeur notre travail est de traiter les erreurs et de les rapporter, remonter à l'utilisateur. Donc, nous allons traiter les cas d'échec en modifiant l'analyse de notre problème. Maintenant aussitôt que notre programme détecte une erreur de syntaxe, au lieu de le planter, nous le forçons dans l'état 4, qui devient l'état final de sortie en erreur.




*14 - De chaque nœud partent 1+5 = 6 symboles*

Ci contre nous obtenons cette table pour l'automate et le graphe correspondant. Le lecteur note que, dans ce cas, toutes les cases de la table de l'AF sont remplies. De même, si la taille de D, le domaine de l'AF est N, alors, chaque nœud du graphe possède N arcs sortants. Ici D= {le, petit, bon, chat, lait, blanc} et N=6.

# 8) 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 et simple. Elle constitue la base de ce cours : nous la retrouverons bientôt.

# Sommaire

Systèmes Formels : l'automate fini (AF)....1
1) Avant d'étudier l'automate fini, revenons sur le séquenceur....1
1.A) Le séquenceur = table à une dimension + registre tampon....1
a) Le registre tampon introduit la dimension temporelle : on passe au séquentiel ....1
b) Expliquons le schéma ci-contre....1
1.B) Graphe de transition....1
a) L'illustration ci-contre représente un graphe de transition....1
b) Étiqueter les flèches qui relient les états....1
2) Rappels à propos de l'automate fini....2
2.A) Fonctionnement....2
2.B) Simplicité....2
2.C) Description mathématique....2
2.D) Le ruban ouvre l'automate sur l'extérieur....2
3) Le graphe de transition....3
3.A) Quel est l'intérêt d'un graphe de transition....3
a) Commençons d'abord par présenter la notion de réseau....3
b) Définition d'un réseau/graphe....3
c) Définition d'un réseau orienté....3
3.B) Une utilisation informatique du graphe de transition pour l'automate....3
a) Introduction....3
b) Étiqueter les flèches qui relient les états....3
c) Utilisons ces conventions de représentation pour illustrer l'alternative....4
d) Chez l'automate, les arcs sont orientés et étiquetés....4
4) Situons la puissance de traitement de l'automate fini....5
4.A) L'instruction de branchement de l'assembleur, le GoTo du basic ou du C....5
4.B) Mais pas une instruction d'appel de sous-programme (langages structurés)....5
4.C) L'automate à pile pour parcourir des réseaux de transition récursifs....5
5) Exemple d'automate fini pour décoder une phrase clé....6
5.A) Introduction....6
5.B) D'abord, voyons quels sont les différents composants de cet automate fini....6
a) La table....6
b) Le ruban externe....6
c) La mémoire (ou encore registre) tampon ....6
d) L'état final ou état accepteur....6
5.C) Le fonctionnement de l'automate....6
5.D) Faisons tourner ce système à la main....6
6) Étude de son fonctionnement, au moyen de son graphe de transition....7
6.A) Focalisons sur le graphe de transition de notre exemple d'automate....7
a) Le graphe....7
b) Rappel des conventions :....7
6.B) Étude d'un exemple, pour illustrer l'utilisation d'un graphe de transition ?....7

a) Au départ : .....7
b) Les symboles en entrée.....7
c) Étude d'un exemple : comment effectuer graphiquement les transitions d'état ?.....7
d) L'état final/accepteur.....7
6.C) Sur le graphe de transition, analysons la phrase : le bon chat blanc.....8
a) Conventions de notation.....8
b) Séquence des transitions d'état.....8
c) En fait.....8
Voici les huit groupes nominaux reconnus par l'automate : .....8
Et voici deux exemples de groupes nominaux qui ne sont pas reconnus par l'automate :…8
7) Automates finis non-déterministes et AF qui ne sont pas déterministes.....9
7.A) Délaisser les aspects mathématiques et les AF non-déterministes.....9
7.B) Un automate fini qui n'est pas déterministe.....9
8) Conclusion.....9