

Automates à états finis et langages réguliers

**Rappels des notions essentielles
et plus de 170 exercices corrigés**

Yliès Falcone

Maître de conférences à
l'Université Grenoble Alpes

Membre du Laboratoire d'Informatique de Grenoble
et d'Inria Grenoble Rhône-Alpes

Jean-Claude Fernandez

Ancien Professeur à l'Université Grenoble Alpes

Ancien membre du laboratoire Verimag

DUNOD

Illustration de couverture :
© Anita Ponne – Adobe Stock

© Dunod, 2020
11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-081424-4

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, 2° et 3° a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

Table des matières

Avant-Propos	9
Introduction	13
1 Rappels et notations	19
1.1 Notions de logique, assertions	19
1.2 Quantificateurs	21
1.3 Notions ensemblistes	22
1.4 Entiers naturels	25
1.5 Ensembles définis inductivement	26
1.6 Raisonnements, techniques de preuve	26
2 Notions préliminaires	29
2.1 Résumé intuitif du chapitre	29
2.2 Les notions essentielles	29
2.3 Exercices	32
2.3.1 Mots et concaténation	32
2.3.2 Langages	33
2.3.3 Concaténation de langages	34
2.4 Indications pour résoudre les exercices	35
2.5 Solutions des exercices	37
2.5.1 Mots et concaténation	37
2.5.2 Langages	41
2.5.3 Concaténation de langages	42
3 Automates déterministes	45
3.1 Résumé intuitif du chapitre	45
3.2 Les notions essentielles	46
3.2.1 Définition et langage reconnu	46
3.2.2 Accessibilité et coaccessibilité	48
3.3 Exercices	49

3.3.1	Automate et langage reconnu	49
3.3.2	Automate reconnaissant un langage	50
3.3.3	Fonction de transition d'un automate	51
3.3.4	Langage à états ou non	53
3.3.5	Accessibilité et coaccessibilité	53
3.3.6	Automates et langages particuliers	53
3.4	Indications pour résoudre les exercices	54
3.5	Solutions des exercices	55
3.5.1	Automate et langage reconnu	55
3.5.2	Automate reconnaissant un langage	59
3.5.3	Fonction de transition d'un automate	66
3.5.4	Accessibilité et coaccessibilité	69
3.5.5	Automates et langages particuliers	70
4	Opérations sur les automates déterministes	71
4.1	Résumé intuitif du chapitre	71
4.2	Les notions essentielles	72
4.3	Exercices	74
4.3.1	Automate complet / complété	74
4.3.2	Automate complémentaire	74
4.3.3	Automate produit	75
4.4	Indications pour résoudre les exercices	76
4.5	Solutions des exercices	77
4.5.1	Automate complété	77
4.5.2	Automate complémentaire	81
4.5.3	Automate produit	85
5	Algorithmes sur les automates déterministes	89
5.1	Résumé intuitif du chapitre	89
5.2	Les notions essentielles	90
5.2.1	Successeurs et prédécesseurs d'un état	90
5.2.2	Parcours	90
5.2.3	Notions d'accessibilité et de coaccessibilité	93
5.2.4	Détection de cycles	97
5.2.5	Quelques problèmes de décision	98
5.3	Exercices	101
5.3.1	Complétion, complémentation et produit	101
5.3.2	Émonder un automate	102
5.3.3	Inclusion et égalité des langages reconnus par des automates	102
5.3.4	Autour de la notion de préfixe	103
5.4	Indications pour résoudre les exercices	103
5.4.1	Compléter un automate	103
5.4.2	Émonder un automate	104
5.4.3	Inclusion et égalité des langages reconnus par des automates	105
5.4.4	Autour de la notion de préfixe	105
5.5	Solutions des exercices	106

5.5.1	Compléter un automate	106
5.5.2	Émonder un automate	110
5.5.3	Inclusion et égalité des langages reconnus par des automates	112
5.5.4	Autour de la notion de préfixe	113
6	Minimisation d'automates déterministes	119
6.1	Résumé intuitif du chapitre	119
6.2	Les notions essentielles	120
6.2.1	Relation de distinguabilité entre états	121
6.2.2	Relation d'équivalence entre états d'un AFED	123
6.2.3	Minimisation	125
6.2.4	Tester l'équivalence entre deux AFED	125
6.3	Exercices	126
6.3.1	Équivalence et distinguabilité entre états	126
6.3.2	Minimisation	127
6.3.3	Équivalence et distinguabilité entre AFED	129
6.4	Indications pour résoudre les exercices	129
6.5	Solutions des exercices	130
6.5.1	Équivalence et distinguabilité entre états	130
6.5.2	Minimisation	130
6.5.3	Équivalence et distinguabilité	135
7	Automates non déterministes	143
7.1	Résumé intuitif du chapitre	143
7.2	Les notions essentielles	144
7.2.1	Définition et langage reconnu	144
7.2.2	Déterminisation des automates non déterministes	145
7.3	Exercices	146
7.3.1	Langage et AFEND	146
7.3.2	Déterminer un AFEND	148
7.3.3	Équivalence entre AFEND	149
7.3.4	Complexité	150
7.3.5	Algorithmes pour les AFEND	151
7.4	Indications pour résoudre les exercices	151
7.5	Solutions des exercices	153
7.5.1	Langage et AFEND	153
7.5.2	Déterminer un AFEND	158
7.5.3	Déterminer l'équivalence entre deux AFEND	161
7.5.4	Complexité	162
7.5.5	Algorithmes pour les AFEND	166
8	Automates non déterministes avec ϵ-transitions	167
8.1	Résumé intuitif du chapitre	167
8.2	Les notions essentielles	168
8.2.1	Définition et langage reconnu	168

8.2.2	Élimination des ϵ -transitions	168
8.2.3	Retour sur la fermeture de la classe des langages à états	170
8.3	Exercices	172
8.3.1	Trouver un automate	173
8.3.2	Fermeture de Kleene	175
8.3.3	Élimination des ϵ -transitions et déterminisation	175
8.3.4	Composition et transformation d' ϵ -AFEND	177
8.3.5	Algorithmes sur les ϵ -AFEND	179
8.4	Indications pour résoudre les exercices	179
8.5	Solutions des exercices	182
8.5.1	Trouver un automate	182
8.5.2	Fermeture de Kleene	186
8.5.3	Élimination des ϵ -transitions et déterminisation	189
8.5.4	Composition et transformation d' ϵ -AFEND	193
8.5.5	Algorithmes sur les ϵ -AFEND	201
9	Expressions régulières	205
9.1	Résumé intuitif du chapitre	205
9.2	Les notions essentielles	206
9.2.1	Syntaxe des expressions régulières	206
9.2.2	Sémantique des expressions régulières	206
9.2.3	Conventions de notation et précedence des opérateurs	207
9.2.4	Quelques propriétés : équivalence et simplification	207
9.2.5	Quelques problèmes de décision : équivalence et inclusion de langages dénotés	207
9.3	Exercices	208
9.4	Indications pour résoudre les exercices	210
9.5	Solutions des exercices	211
10	Théorème de Kleene	219
10.1	Résumé intuitif du chapitre	219
10.2	Les notions essentielles	220
10.2.1	Théorème de Kleene	220
10.2.2	Des automates vers les expressions régulières	220
10.2.3	Des expressions régulières vers les automates	224
10.3	Exercices	228
10.3.1	Calcul d'expressions régulières à partir d'automates	229
10.3.2	Calcul d'automates à partir d'expressions régulières	231
10.4	Indications pour résoudre les exercices	232
10.5	Solutions des exercices	233
10.5.1	Calcul d'expressions régulières à partir d'automates	233
10.5.2	Calcul d'automates à partir d'expressions régulières	241

11 Grammaires	247
11.1 Résumé intuitif du chapitre	247
11.2 Les notions essentielles	247
11.2.1 Définition des grammaires	247
11.2.2 Dérivations en une et plusieurs étapes	248
11.2.3 Langage généré par une grammaire	248
11.2.4 Classification de Chomsky des grammaires	248
11.3 Exercices	249
11.3.1 Générer des mots avec les grammaires	249
11.3.2 Langages et grammaires	250
11.4 Indications pour résoudre les exercices	251
11.5 Solutions des exercices	252
11.5.1 Générer des mots avec les grammaires	252
11.5.2 Langages et grammaires	252
12 Grammaires régulières	257
12.1 Résumé intuitif du chapitre	257
12.2 Les notions essentielles	257
12.2.1 Grammaires et automates	257
12.2.2 Grammaires et expressions régulières	260
12.3 Exercices	261
12.3.1 Des grammaires vers les automates	261
12.3.2 Des automates vers les grammaires	261
12.3.3 Des expressions régulières vers les automates et les grammaires	262
12.3.4 Grammaires non régulières	262
12.3.5 Composition de grammaires régulières	263
12.4 Solutions des exercices	263
12.4.1 Des grammaires vers les automates	263
12.4.2 Des automates vers les grammaires	264
12.4.3 Des expressions régulières vers les automates et les grammaires	269
12.4.4 Grammaires non régulières	269
12.4.5 Composition de grammaires régulières	271
13 Propriété de l'itération	273
13.1 Résumé intuitif du chapitre	273
13.2 Les notions essentielles	273
13.2.1 Lemme de l'itération	273
13.2.2 Constante d'itération	274
13.3 Exercices	275
13.3.1 Lemme de l'itération	275
13.3.2 Constante d'itération	275
13.4 Indications pour résoudre les exercices	275
13.5 Solutions des exercices	276
13.5.1 Lemme de l'itération	276
13.5.2 Constante d'itération	277

14 Démontrer la non-régularité	281
14.1 Résumé intuitif du chapitre	281
14.2 Les notions essentielles	281
14.2.1 Utilisation du lemme de l'itération	281
14.2.2 Utilisation des propriétés de fermeture	282
14.2.3 Une condition suffisante pour la non-régularité	283
14.3 Exercices	283
14.3.1 Lemme de l'itération	283
14.3.2 Fermeture des langages réguliers	284
14.3.3 Une condition suffisante pour la non-régularité	286
14.4 Indications pour résoudre les exercices	286
14.5 Solutions des exercices	287
14.5.1 Lemme de l'itération	287
14.5.2 Fermeture des langages réguliers	291
14.5.3 Une condition suffisante pour la non-régularité	294
A Modélisation et résolution de problèmes avec les automates	295
A.1 Exercices	295
A.2 Solutions des exercices	299
Bibliographie	311
Index	313

Avant-Propos

À qui s'adresse ce livre ?

Cet ouvrage s'adresse aux étudiants de premier cycle universitaire suivant un cursus incluant l'informatique (étudiants dans un parcours d'informatique intégral, de mathématiques appliquées ou de mathématiques). Il vise aussi bien les étudiants des IUT, de licences et de classes préparatoires aux grandes écoles. Peu de pré-requis sont nécessaires et correspondent à ceux d'un cours de la théorie des ensembles telle que enseignée en première année de l'enseignement supérieur. Les notions de cours nécessaires sont par ailleurs rappelées dans le premier chapitre.

Langages, automates et informatique

La théorie des langages et des automates revêt un aspect particulier dans l'apprentissage de l'informatique. C'est un sujet qui appartient à la branche fondamentale de l'informatique, tant les concepts abordés se retrouvent dans de nombreuses disciplines de l'informatique aussi bien théoriques qu'appliquées (par exemple la conception des processeurs, la compilation de programmes informatiques, la traduction automatique des langues naturelles, l'intelligence artificielle, la bio-informatique, la vérification de programmes embarqués critiques, la cybersécurité, etc.). Nous développons ce point dans l'introduction en donnant des exemples d'utilisation de cette théorie. La théorie des langages et automates est un enseignement incontournable dans tout cursus d'informatique, tant en France qu'à l'étranger.

Philosophie du livre

L'objectif de cet ouvrage est de conduire le lecteur vers la maîtrise des concepts et techniques de langage à états, ou langage rationnel. Il est orienté vers la pratique d'exercices. Il se veut être un outil accompagnant le travail des étudiants à travers à la fois des exercices simples d'application pour l'appropriation des notions et des exercices plus avancés pour la maîtrise des concepts. Chaque chapitre comporte un rappel des notions essentielles du cours

Chapitre par chapitre. Nous donnons une brève description de chaque chapitre.

Dans le chapitre 1, nous rappelons les notions de mathématiques nécessaires et suffisantes : notions de logique, d'ensembles et de techniques de démonstration. Il n'y a pas d'exercice associé à ce chapitre.

Dans le chapitre 2, nous introduisons les notions d'alphabet, qui est un ensemble fini de symboles, de mots et de langage. Un mot est une composition de symboles appartenant à un alphabet, et un langage un ensemble de mots. Un alphabet est un ensemble fini de symboles et la composition de symboles pour former un mot s'appelle concaténation.

Dans le chapitre 3, nous définissons la notion d'automate déterministe où les règles de transition sont définies par une fonction qui, étant donné un état et une action, détermine l'état suivant. On définit le langage associé à un automate : une configuration est un couple (état, mot) et un mot est accepté si, partant d'une configuration initiale (état initial, mot), on atteint une configuration terminale formée d'un état accepteur et du mot vide.

Dans le chapitre 4, nous définissons les opérations sur les automates permettant de calculer l'intersection, l'union ou le complément de langages associés à un automate.

Dans le chapitre 5, nous présentons les algorithmes sur les automates en lien avec les opérations présentées au chapitre précédent.

Dans le chapitre 6, partant de la notion d'équivalence (ou de distinguabilité) entre états, nous présentons la minimisation des automates, c'est-à-dire que, pour un automate donné, nous pouvons obtenir un automate avec un nombre minimal d'états qui reconnaît le même langage. À partir de la relation d'équivalence entre états, on obtient l'automate quotient, c'est-à-dire l'automate dont les états sont les classes d'équivalence de l'automate initial et la fonction de transition est celle induite par la fonction de transition initiale compatible avec la relation d'équivalence.

Dans le chapitre 7, nous introduisons la notion d'automates non déterministes, en remplaçant dans la définition des automates la fonction de transition par une relation de transition : ainsi, à partir d'un état et d'un symbole, on peut atteindre un ensemble d'états au lieu d'un seul état. Nous avons la propriété suivante : pour tout langage reconnu par un automate non déterministe, il existe un automate reconnaissant le même langage. Nous en déduisons une procédure de détermination de l'automate non déterministe.

Dans le chapitre 8, nous introduisons les automates non déterministes avec ϵ -transition et on a un résultat similaire au précédent, c'est-à-dire que, à chacun de ces automates, on a un automate déterministe reconnaissant le même langage.

Dans le chapitre 9, nous introduisons les expressions régulières qui constituent un autre formalisme pour dénoter des langages d'états finis, ou langages rationnels.

Dans le chapitre 10, nous étudions le théorème de Kleene, indiquant que la classe des langages rationnels est celle des langages reconnus par des automates d'états finis. Ainsi, pour chaque expression régulière, nous pouvons trouver un automate reconnaissant le langage dénoté par cette expression régulière, et inversement.

Dans le chapitre 11, nous introduisons les grammaires et en particulier les grammaires régulières. Nous étudions l'équivalence entre formalismes pour le cas des grammaires régulières dans le chapitre 12 : pour tout langage reconnu par un automate à nombre fini d'états, il existe une expression régulière reconnaissant le même langage (et réciproquement), de plus il existe une grammaire régulière reconnaissant le même langage (et réciproquement).

Le chapitre 13 est dédié au lemme de l'itération, qui est une condition nécessaire à la régularité d'un langage. Nous utilisons le lemme de l'itération et les propriétés de fermeture

des langages réguliers dans le chapitre 14 pour démontrer que certains langages ne sont pas réguliers, c'est-à-dire qu'il n'existe pas d'automate reconnaissant ces langages.

Dans l'Annexe A, nous abordons la modélisation de problèmes et leur résolution en utilisant les automates. Nous introduisons également la notion de propriété caractérisée par un langage. Une modélisation (sous forme d'automate) satisfait une propriété si le langage associé à l'automate est inclus dans le langage associé à la propriété.

Remerciements

Cet ouvrage est né après plusieurs années d'enseignement d'un cours à l'Université Grenoble Alpes sur le sujet. Dans ses formes préliminaires, l'ouvrage a bénéficié du retour de plusieurs collègues participant à l'enseignement (en particulier Saddek Bensalem, Yassine Lakhnech et Anne Rasse), que nous remercions. Nous remercions également les étudiants, doctorants et post-doctorants de l'Université Grenoble Alpes Théo Barollet, Alaa Ben Fatma, Tom Cornebize, Florian Gallay, Ali Kassem, Charlotte Lefèvre et Marius Monnier, qui ont participé à la relecture des chapitres. Leurs relectures ont permis de s'assurer que les solutions des exercices soient abordables et claires du point de vue de l'étudiant. Enfin, nous remercions l'Université Grenoble Alpes¹, Inria Grenoble Rhône-Alpes² et le Laboratoire d'Informatique de Grenoble³ pour les moyens et l'environnement de qualité mis à notre disposition facilitant notre travail quotidien.

Pour contacter les auteurs

Pour vos remarques ou rapports de coquilles, vous pouvez contacter les auteurs par email⁴.

1. www.univ-grenoble-alpes.fr

2. www.inria.fr/fr/centre-inria-grenoble-rhone-alpes

3. www.liglab.fr

4. prenom.nom@univ-grenoble-alpes.fr

Introduction

Pourquoi étudier les automates et la théorie des langages ?

Cet ouvrage traite des concepts et techniques de langage à états, ou langage rationnel. Les notions abordées ont une importance fondamentale en informatique. Elles sont utilisées dans de nombreux domaines, que ce soit en informatique théorique ou appliquée. Le livre fait le lien entre une certaine classe de langages informatiques et les automates à nombre fini d'états (aussi appelés par ailleurs automates à états finis ou automates d'états finis). La notion d'automates est très ancienne⁵ et a été utilisée dans les années 1950-1960 pour le calcul. Les premiers processeurs étaient constitués d'une **partie contrôle** et d'une **partie opérative**. La partie contrôle est un automate qui pilote la partie opérative pour réaliser par exemple des opérations arithmétiques et logiques ou pour faire des transferts mémoires. Par la suite, les automates ont été utilisés dans :

- la conception de matériel informatique,
- la reconnaissance de mots, l'analyse lexicale d'un compilateur,
- la reconnaissance de texte (formulaire internet, moteur de recherche, etc.),
- la conception et la vérification de protocoles de communication,
- la programmation de robots, de planification,
- la modélisation de propriétés et la vérification de programmes,
- la compression de textes,
- l'intelligence artificielle.

Même si les automates sont très anciens et s'ils sont beaucoup utilisés en pratique de nos jours, ils constituent toujours un sujet étudié en recherche, en tant que tel ou comme outil de résolution de problème.

5. fr.wikipedia.org/wiki/Automate_mecanique

Première rencontre avec les automates

Dans ce qui suit, pour abrégé, nous utilisons le terme automate pour automate à nombre fini d'états.

Un automate est représenté sous forme de graphe, dont les nœuds (cercles) représentent les états, les nœuds doubles les états accepteurs et les arcs (flèches) représentent les transitions ; voir figure 2. À l'intérieur du nœud, nous indiquons le nom de l'état. Un automate est caractérisé par un nombre fini d'états, un vocabulaire qui désigne des actions et des règles permettant de passer d'un état à un autre. Finalement, un automate est un quintuplet (ensemble d'états, un vocabulaire d'actions, un état initial, un nombre fini de règles de transition et un ensemble d'états accepteurs. D'autres automates, machines ne sont pas abordées dans ce livre : les automates à états finis avec vocabulaires d'entrée et de sortie comme les automates de Moore et de Mealy, les automates à piles en lien avec les grammaires hors contextes, les machines de Turing en lien avec les notions de calculabilité et de décidabilité.

Un langage est défini par un ensemble de symboles, de mots sur ces symboles et par une grammaire indiquant comment les mots peuvent être utilisés. Dans un compilateur, les mots du langage constituent un langage régulier et leur reconnaissance peut se faire par un automate à états fini.

Nous considérerons dans la suite quelques exemples d'automates.

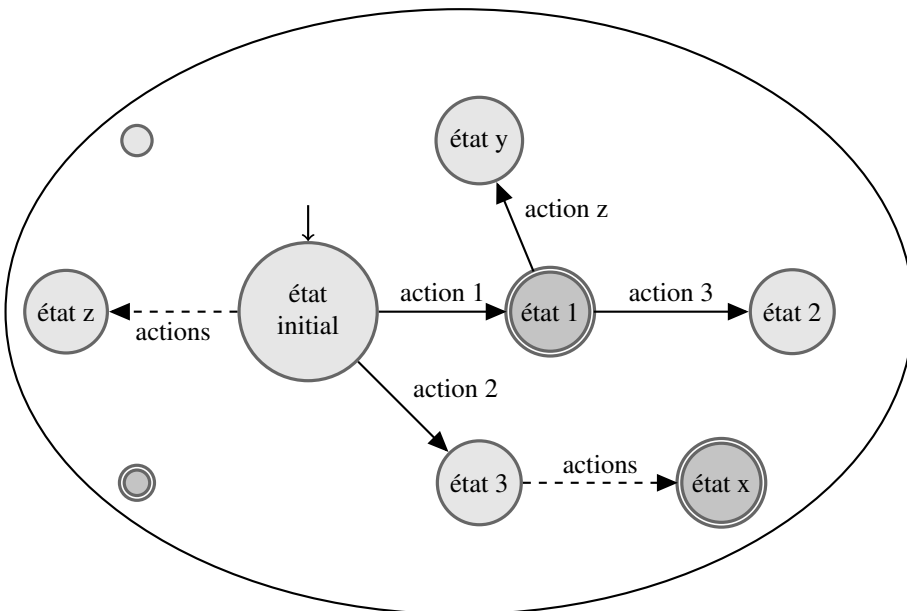


Figure 2 – Représentation schématique d'un automate.

Un automate reconnaissant la chaîne « abc »

Un premier exemple est un automate qui reconnaît une chaîne de caractères, mettons « abc ». Cet automate est représenté sur la figure 3. C'est un automate à quatre états : 0, 1, 2 et 3. Cet automate lit des mots (ou des phrases) sur l'alphabet latin. De l'état initial 0, par la lettre « a » on passe à l'état 1. De l'état 1, par « b », on va à l'état 2 et, de l'état 2, par la lettre « c », on va dans l'état 3, qui est un état accepteur. Tout ce qui n'est pas spécifié par l'automate conduit implicitement dans un état de rejet. Étant donné une séquence de symboles à lire, un automate lit chaque symbole, l'un après l'autre, dans l'ordre de la séquence. Au démarrage, avant de lire le premier symbole, un automate est dans son état initial.

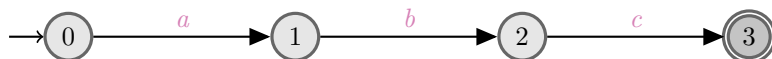


Figure 3 – Un automate lisant des mots sur l'alphabet latin et reconnaissant la chaîne « abc ».

Un automate reconnaissant les mots terminant par « ant »

Un second exemple d'automate est représenté dans la figure 4. Nous supposons pour simplifier que l'automate ne lit qu'un seul mot et doit déterminer si celui-ci termine par « ant », on dit dans ce cas que l'automate reconnaît ou accepte le mot. L'état initial **rien vu** est indiqué par une transition sans état de départ (flèche entrant dans le nœud **rien vu**). L'automate évolue d'état en état en fonction du symbole lu en entrée et de son état courant.

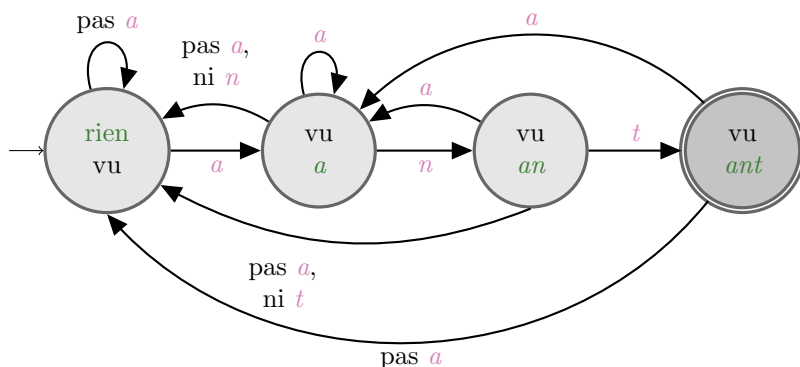


Figure 4 – Un automate lisant des mots sur l'alphabet latin et reconnaissant ceux qui terminent par « ant ».

- Depuis l'état **rien vu**, si le symbole lu est différent du symbole **a** alors l'automate reste dans l'état **rien vu**, sinon (si le caractère lu est **a**) alors l'automate va dans l'état **vu a**.
- Depuis l'état **vu a**, si le caractère lu est **a** alors l'automate reste dans l'état **vu a**, si le caractère lu est **n** alors l'automate va dans l'état **vu an**, sinon (si le caractère lu est ni **a** ni **n**), l'automate retourne dans l'état **rien vu**.

- Depuis l'état vu **an**, si le caractère lu est **a** alors l'automate retourne dans l'état vu **a**, si le caractère lu est **t** alors l'automate va dans l'état vu **ant**, sinon (si le caractère lu est ni **a** ni **t**), l'automate retourne dans l'état **rien** vu.
- Depuis l'état vu **ant**, si le caractère lu est **a** alors l'automate retourne dans l'état vu **a**, sinon (si le caractère lu n'est pas **a**) alors l'automate retourne dans l'état **rien** vu.

Si l'automate s'arrête dans l'état vu **ant** (qui est un état accepteur), alors l'automate accepte le mot. On remarque que les états de l'automate correspondent aux différentes situations dans la lecture caractère par caractère d'un mot par rapport au fait que ce mot termine par « ant ».

Des automates pour un protocole de transaction électronique

Nous modélisons une transaction électronique, avec comme entités un client, un marchand et une banque, inspiré de [10]; voir figure 5. Nous considérons un protocole très simplifié, mais qui nous permet d'illustrer sa vérification automatisée.

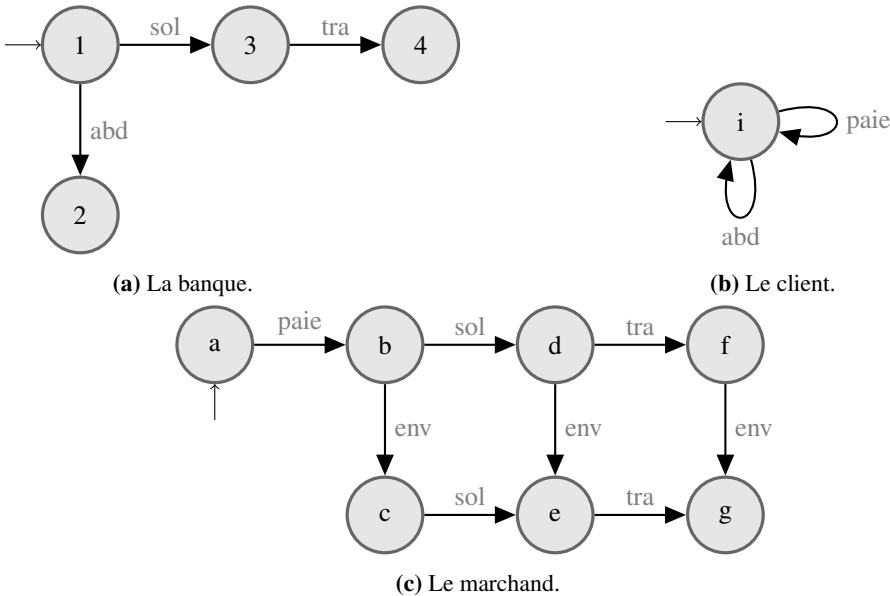


Figure 5 – Automates pour le protocole de transaction électronique.

Les participants de la transaction. Le client veut acheter une marchandise chez le marchand et la payer de manière électronique.

- Les actions du client sont « payer la marchandise » (action **paie**) ou « abandonner la transaction et récupérer son argent » (action **abd**).
- Le marchand peut envoyer la marchandise (action **env**) et solder le paiement (action **sol**). Ces deux actions peuvent être effectuées dans n'importe quel ordre.
- La banque peut transférer l'argent au marchand (action **tra**).