UNIVERSITÉ Grenoble Alpes

INF 302 : Langages et Automates

Travaux Dirigés

Univ. Grenoble Alpes
Licence Sciences et Technologies
Département Licence Sciences et Technologies

www.univ-grenoble-alpes.fr
dlst.univ-grenoble-alpes.fr

Année Académique 2020 - 2021

Table des matières

Introduction

Ce livret contient les exercices de l'Unité d'Enseignement (UE) INF 302 enseignée à l'Université Grenoble Alpes, France.

Information de contact

Voici les informations de contact en cas de question ou pour tout problème lié à l'UE :

- Pour des questions générales liées au cours, merci de contacter votre enseignant de cours.
- Pour des question techniques liées aux exercices, merci de contacter vos enseignants responsables des travaux dirigés.
- Pour des questions concernant ce document ou l'UE en général, merci de contacter Yliès Falcone.

Emails Voici les emails de vos enseignants.

- Saddek Bensalem, Cristian Ene, Yliès Falcone, Akshay Mambakam, Anne Rasse, Vincent Tavernier : `prenom.nom@univ-grenoble-alpes.fr`;
- Nicolas Basset : `nicolas.basset1@univ-grenoble-alpes.fr`;
- David Rouquet : `david.rouquet@tetras-libre.fr`;
- Alexandra Steinhilber : `alexandra.steinhilber@grenoble-inp.org`.

À propos des exercices et des chapitres de TD

- Le livret de travaux dirigés devrait contenir le nombre nécessaire et suffisant d'exercices pour le semestre. Pour l'examen final, vous êtes censés les avoir tous faits. Vos enseignants de travaux dirigés sont là pour vous aider sur les exercices que vous n'arriveriez pas à faire tout seul.
- Le rythme usuel est d'environ d'un chapitre par séance de TD. Le détail est ci-dessous :
  - chapitre 1 : 1 séance ;
  - chapitre 2 : 1 séance ;
  - chapitre 3 : 2 séances ;
  - chapitre 4 : 1 séance ;
  - chapitre 5 : 1 séance ;
  - chapitre 6 : 1 séance ;
  - chapitre 7 : 1 à 2 séances ;
  - chapitre 8 : 1 à 2 séances ;
  - chapitre 9 : 1 à 2 séances ;
  - chapitre 10 : 1 à 2 séance ;
  - chapitre 11 : 1 séance ;
  - chapitre 12 : 1 séance ;
  - chapitre 13 : 1 séance ;
  - chapitre 14 : 2 séances.
  - interrogations durant le semestre : 2 séances au total.
- Les exercices de difficulté 1 (♠) sont des exercices d'application directe du cours. Il est préférable de les préparer avant la séance de TD pour vous assurer que vous avez compris les concepts. Les exercices de difficulté 2 (♠♠) sont des exercices d'entrainement et indispensables, vous devriez les avoir tous traiter à l'issue de la séance de TD. Les exercices de difficulté 3 (♠♠♠) sont des exercices plus difficiles demandent plus de reflexion et seront traités en TD en fonction du temps disponible (et surtout une fois que les exercices de difficulté 2 sont maitrisés). Il est possible que l'examen contienne un ou deux exercices de ce niveau.

Aude : Automata demystifier

Aude est un logiciel en ligne fait par et pour les étudiants de l'UE. Il encourage le travail en autonomie des étudiants sur l'UE sur la plupart des concepts abordés durant l'UE. En particulier, vous pouvez obtenir la solution de certains exercices en utilisant Aude. Aude est disponible à l'adresse suivante :

https://aude.imag.fr

Vous aurez une présentation du logiciel durant le cours magistral.
Pour toute question ou rapport de bug concernant Aude, merci d'écrire à l'adresse

aude-contact@univ-grenoble-alpes.fr

Quelques conseils

Les conseils suivants peuvent sembler cliché et/ou naifs mais les suivre peut être un sérieux atout pour votre réussite à l'UE.

Soyez attentifs durant les cours. Votre objectif est d'avoir assimilé la plus grosse partie possible en sortant de l'amphi. Retravaillez le cours le soir même et venez au prochain amphi ou au prochain TD avec des questions sur ce que vous n'avez pas compris. Contrairement à une croyance durement établie, les séances de travaux dirigés ne sont pas faites pour comprendre le cours mais pour s'entrainer à faire les exercices mieux et plus rapidement.

Posez des questions durant les cours si vous avez le moindre doute sur une notion abordée. Si vous vous posez une question, plusieurs de vos camarades se posent probablement la même question.

Travaillez dur et régulièrement. Penser qu'il est possible d'assimiler le contenu de l'UE une semaine avant l'examen est illusoire. Pour la résolution d'un exercice, un bon algorithme consiste à essayer (sérieusement), dans un premier temps, de résoudre l'exercice seul. Si vous n'y arrivez pas, essayer de la résoudre avec un camarade. En dernier recours, faites appel à la solution ou à vos enseignants.

Ne vous perdez pas au milieu du semestre. Discuter avec vos camarades des concepts que vous ne comprenez pas et/ou contactez vos enseignants.

Contactez nous. N'hésitez pas ! Nous sommes généralement disponibles et sommes heureux de vous aider.

Ne jamais abandonner. Assez évident, mais ça va mieux en le disant.

CHAPITRE

1 Notions mathématiques de base

Ce chapitre est destiné à vous exercer sur les notions de base en mathématiques.

Exercice 1 () — Ensemble des sous-ensembles d'un ensemble - définition

Considérons un ensemble E de cardinal fini et $\mathcal{P}(E)$ l'ensemble de ses parties ou l'ensemble de ses sous-ensembles.

1. Rappeler la définition (formelle) de $\mathcal{P}(E)$.
2. Pour $E = \{1, 2, 3\}$, donner $\mathcal{P}(E)$.

Exercice 2 () — Ensemble des sous-ensembles d'un ensemble - propriétés

Prouver les propositions suivantes :

1. $\mathcal{P}(A) = \mathcal{P}(B)$ ssi $A = B$.
2. $\mathcal{P}(A \cup B) = \{X \cup Y \mid X \in \mathcal{P}(A) \wedge Y \in \mathcal{P}(B)\}$.
3. $\mathcal{P}(A \cap B) = \mathcal{P}(A) \cap \mathcal{P}(B)$.
4. En général, $\mathcal{P}(A \cup B) \neq \mathcal{P}(A) \cup \mathcal{P}(B)$.

Exercice 3 () — Relations et leur propriétés

1. Rappeler les définitions formelles mathématiques des éléments suivants : relation, fonction, application, relation réflexive, relation anti-réflexive, relation symétrique, relation antisymétrique, relation transitive, relation d'équivalence, classe d'équivalence.
2. Donner un exemple pour chacun des éléments mentionnés dans la question précédente.

Exercice 4 () — Une relation d'équivalence

Considérons la relation $R \subseteq \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$ définie comme suit :

$$\forall (a, b), (c, d) \in \mathbb{Z} \times (\mathbb{Z} \setminus \{0\}) : (a, b)\, R\, (c, d) \iff a \times d - b \times c = 0.$$

1. Prouver que R est une relation d'équivalence.
2. Donner ses classes d'équivalence.

Exercice 5 () — Preuves par récurrence

1. Prouver la proposition suivante :

$$\forall n \in \mathbb{N} : \sum_{i=0}^{n} i = \frac{n(n+1)}{2}.$$

2. Déduire que $1 + 3 + 5 + ... + (2n - 1) = n^2$.
3. Prouver la proposition suivante :

$$\forall n \in \mathbb{N} : \sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}.$$

Exercice 6 () — Entiers naturels définis par induction

Soit E un ensemble inductivement défini par les règles suivantes :

- **Règle de base** : $0 \in E$
- **Règle d'induction** : si $x \in E$, alors $\mathrm{s}(x) \in E$

1. Proposer une définition d'une fonction $+$ qui se comporte comme l'addition sur les entiers (où $\mathrm{s}(x)$ est l'entier après x). La fonction doit être définie par induction sur son premier argument.
2. Proposer une définition d'une fonction $*$ qui se comporte comme la multiplication sur les entiers. La fonction doit être définie par induction sur son premier argument.
3. Prouver les propriétés suivantes :
   — $\forall x \in E : x * 0 = 0 = 0 * x$,
   — $\forall x, y \in E : x * y = y * x$,
   — $\forall x, y, z : (x * y) * z = x * (y * z)$,
   — $\forall x, y, z : (x + y) * z = x * z + y * z$.

Exercice 7 () — Listes définies par induction

Soit $E(\Sigma)$ l'ensemble des listes dont les éléments sont dans l'ensemble Σ et qui est inductivement défini comme suit :

- **Règle de base** : $nil \in E(\Sigma)$,
- **Règle d'induction** : si $l \in E(\Sigma)$, $\mathrm{cons}(a, l) \in E(\Sigma)$, pour chaque $a \in \Sigma$, où cons est l'opérateur de concaténation d'un élément à une liste.

Soit $\Sigma = \{a, b\}$

1. Rappeler la définition de l'opérateur cons.
2. Donner une définition inductive de l'ensemble des listes qui contiennent le même nombre de a que de b.
3. Donner une définition inductive de l'ensemble des listes qui contiennent le même nombre de a que de b et qui commencent par des a suivis par des b. Entre les a, il ne doit pas y avoir de b.

CHAPITRE

2 Notions préliminaires

Exercice 8 (♠) — Concaténation de mots

En reprenant un des exemples utilisés pour illustrer la concaténation de mots en cours :
- La concaténation des mots 01 et 10 est le mot 0110.
- La concaténation du mot vide ϵ et du mot 101 est le mot 101.

1. Donner la définition formelle des applications correspondant à ces 5 mots et montrer que les deux affirmations ci-dessus sont cohérentes avec la définition de l'opération de concaténation vue en cours.

Exercice 9 (♠♠) — Longueur et nombre d'occurrences d'un symbole

Considérons un alphabet Σ.

1. En utilisant la définition non-inductive de mot comme une application, définir la fonction qui donne la longueur d'un mot.
2. En utilisant la définition non-inductive de mot comme une application, définir la fonction qui donne le nombre d'occurrences d'un symbole dans un mot.
3. Mêmes questions que précédemment en utilisant la définition inductive de mot.

Exercice 10 (♠♠) — Élément neutre de la concaténation

Considérons Σ un alphabet et ϵ_Σ le mot vide sur Σ.

1. En considérant la définition non-inductive de mot, montrer que le mot ϵ_Σ est l'élément neutre de la concaténation, c'est-à-dire $\forall u \in \Sigma^* : u \cdot \epsilon_\Sigma = \epsilon_\Sigma \cdot u = u$.

Exercice 11 (♠♠♠) — Puissance d'un alphabet

Rappelons que pour un alphabet Σ et Σ^k sont les ensembles de mots sur Σ respectivement de longueur égale à k.

1. Donner le cardinal de Σ^k.
2. Prouver que $\forall k \in \mathbb{N} : \Sigma^{k+1} = \Sigma^k \cdot \Sigma^1$.
3. Prouver que $\forall k \in \mathbb{N} : \Sigma^{k+1} = \Sigma^1 \cdot \Sigma^k$.
4. Démontrer le résultat sur la cardinalité de Σ^k.

Exercice 12 (♠♠♠) — Cardinal et concaténation

Considérons un alphabet Σ. Nous nous intéressons à la concaténation $L_1 \cdot L_2$ de deux langages L_1 et L_2 définis sur Σ.

1. Donner deux langages L_1 et L_2, avec $L_1 \neq \emptyset$ et $L_2 \neq \emptyset$, tels que $|L_1 \cdot L_2| < |L_1| \times |L_2|$.
2. Démontrer que $\forall L_1, L_2 \subseteq \Sigma^* : |L_1 \cdot L_2| \leq |L_1| \times |L_2|$.
3. Donner des conditions suffisantes pour que $|L_1 \cdot L_2| = |L_1| \times |L_2|$.

CHAPITRE

3 Automates à états finis déterministes

Rappels :
- AEFD : automate à états fini déterministe.
- Un automate est dit accessible (resp. co-accessible) si tous ces états sont accessibles (resp. co-accessibles).
- Un automate est dit émondé s'il est accessible et co-accessible.

Exercice 13 (♠) — Langage reconnu par un automate

Considérons l'alphabet $\{a, b\}$ et les AEFD dans la Figure 3.1.

1. Décrire en langage naturel les langages reconnus ces AEFD.
2. Comment les descriptions trouvées dans la question précédente seraient-elles modifiées si nous avions considéré l'alphabet $\{a, b, c\}$?

Exercice 14 (♠) — Représentation tabulaire d'un automate

Considérons l'alphabet $\{a, b\}$. Vous pourrez vous limiter à deux ou trois automates pour vérifier que vous avez compris le principe.

1. Donner la représentation tabulaire des AEFD dans la Figure 3.1.
2. Donner la représentation tabulaire des AEFD trouvés dans l'Exercice 15.

Exercice 15 (♠) — Automate pour un langage avec des contraintes sur le nombre de symboles

Considérons l'alphabet $\{a, b\}$. Pour chacun des ensembles suivants, donner un AEFD qui reconnaît cet ensemble de mots, si cela est possible.

1. L'ensemble des mots qui contiennent un nombre de a multiple de 2.
2. L'ensemble des mots qui contiennent un nombre de a multiple de 3.
3. L'ensemble des mots qui contiennent exactement n occurrences du symbole a, pour un certain $n \in \mathbb{N}$.
4. L'ensemble des mots qui contiennent moins de n de occurrences du symbole a, pour un certain $n \in \mathbb{N}$.
5. L'ensemble des mots qui contiennent plus de n de occurrences du symbole a, pour un certain $n \in \mathbb{N}$.
6. L'ensemble des mots qui contiennent autant de a que de b.
7. L'ensemble des mots tels que chaque bloc de 3 symboles consécutifs contienne (exactement) 2 occurrences du symbole a.

Exercice 16 (♠♠) — Automate pour un langage avec certains préfixes, suffixes ou facteurs

Considérons l'alphabet $\{a, b, c\}$. Rappelons que $\cdot$ désigne l'opérateur de concaténation entre mots (et langages). Pour chacun des langages suivants, donner un automate qui le reconnaît (si un tel automate existe).

1. L'ensemble des mots qui commencent par $a \cdot b$ ou $b \cdot c$.
2. L'ensemble des mots qui ne commencent ni par $a \cdot b$ ni par $b \cdot c$.
3. L'ensemble des mots qui contiennent $a \cdot b$.
4. L'ensemble des mots qui ne contiennent pas $a \cdot b$.
5. L'ensemble des mots qui terminent par $a \cdot b \cdot c$.
6. L'ensemble des mots qui ne terminent pas par $a \cdot b \cdot c$.

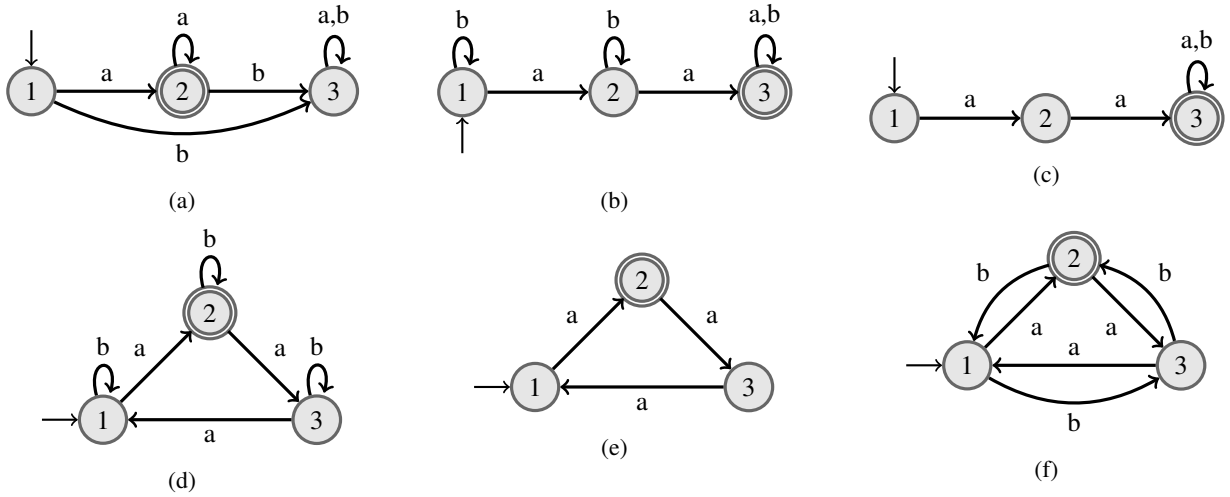


FIGURE 3.1 – Des automates à états finis déterministes

7. L'ensemble des mots de longueur supérieure ou égale à 2 et tels que l'avant-dernier symbole est b.

Exercice 17 (♠♠♠) — Fonction de transition et symbole puits

Considérons un AEFD $(Q, q_0, \Sigma, \delta, F)$ et $a \in \Sigma$ un symbole particulier tel que $\forall q \in Q : \delta(q, a) = q$.

1. Prouver que : $\forall n \in \mathbb{N} : \delta^*(q, a^n) = q$ où a^n est le mot formé en concaténant n a's (c'est-à-dire $\underbrace{a \cdot a \ldots a}_{n \text{ fois}}$).
2. Prouver que soit $\{a\}^* \subseteq L(A)$ soit $\{a\}^* \cap L(A) = \emptyset$.

Exercice 18 (♠♠♠) — Sous et sur-automate

Soit A un automate. On appelle sous-automate de A tout automate obtenu en enlevant un ou plusieurs états de l'ensemble des états de A, à l'exception de l'état initial, ou en supprimant une ou plusieurs transitions de la fonction de transition de A et sans changer la nature des états ni l'état initial. On appelle sur-automate de A tout automate obtenu en ajoutant un ou plusieurs états de l'ensemble des états de A, en ajoutant une ou plusieurs transitions à la fonction de transition de A (tout en préservant le déterminisme de l'automate) ; le sur-automate a le même état initial et son ensemble d'états accepteurs est un sur-ensemble de l'ensemble des états accepteurs de A.

1. Étant donné un AEFD $A = (Q, q_0, \Sigma, \delta, F)$, définir (formellement) à quelle condition un automate $A' = (Q', q'_0, \Sigma, \delta', F')$ est un sous-automate de A.
2. Démontrer que le langage reconnu par tout sous-automate de A est inclus dans le langage reconnue par A.
3. Démontrer que le langage reconnu par tout sur-automate de A contient le langage reconnue par A.

CHAPITRE

4 Opérations de composition d'automates déterministes

Dans ce chapitre, vous pouvez réutiliser les automates du chapitre précédent. Pour rappel, le langage accepté par un AEFD A est noté $L(A)$.

Exercice 19 (♠) — Complétude d'un automate

Considérons l'alphabet $\{a, b\}$ et les AEFD dans la Figure 3.1.

1. Déterminer quels automates sont complets.
2. Compléter les automates qui ne sont pas complets.
3. Compléter les automates en considérant l'alphabet $\{a, b, c\}$?

Exercice 20 (♠) — Trouver l'automate complémentaire

Considérons l'alphabet $\Sigma = \{a, b\}$. Pour chacun des automates suivants, donner un automate qui reconnait le complémentaire du langage reconnu.

1. L'automate représenté dans la Figure 4.1a.
2. L'automate représenté dans la Figure 4.1b.
3. L'automate représenté dans la Figure 4.1c.

Exercice 21 (♠♠) — Obtenir un automate par produit d'automates

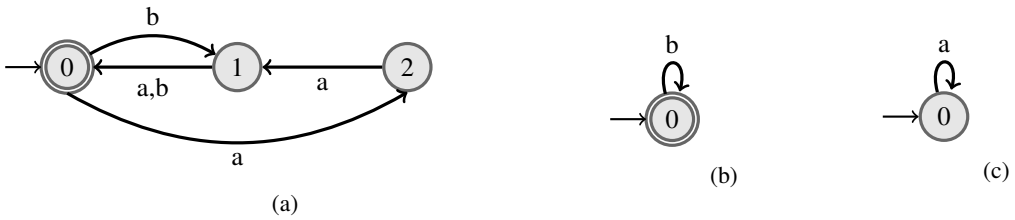
Pour les automates suivants, utiliser la construction de l'automate produit. Considérons l'alphabet $\Sigma = \{a, b\}$ et les automates trouvés dans l'Exercice 15. Pour chacun des langages suivants, donner un AEFD qui le reconnaît.

1. L'ensemble des mots qui contiennent un nombre de a multiple de 3 et multiple de 2.
2. L'ensemble des mots qui contiennent un nombre de a multiple de 2 et non multiple de 3.
3. Les langages complémentaires dans Σ^* de chacun des langages précédents.

Exercice 22 (♠♠) — Obtenir un automate par produit d'automates

Considérons l'alphabet $\Sigma = \{a, b, c\}$. Pour chacun des langages suivants, en ré-utilisant les automates de l'Exercice 16, donner un automate qui le reconnaît.

1. L'ensemble des mots qui commencent par $a \cdot b$ ou $b \cdot c$ et qui ne terminent pas par $a \cdot b \cdot c$.
2. L'ensemble des mots qui contiennent un nombre pair de c et qui ne contiennent pas $a \cdot b$.



(a) (b) (c)

FIGURE 4.1 – Des automates à états finis déterministes

Exercice 23 (♠♠♠) — Correction de l'opération de complétion

Nous souhaitons montrer que l'opération/l'algorithme de complétion est correcte, c'est-à-dire qu'elle ne change pas le langage de l'automate sur lequel elle est appliquée. Étant donné un AEFD A, $C(A)$ dénote l'automate résultant de l'application de l'opération de complétion à A.

1. Montrer que $L(C(A)) = L(A)$.

Exercice 24 (♠♠♠) — Correction de l'opération de complémentation

Nous souhaitons montrer que l'opération de complémentation est correcte, c'est-à-dire qu'elle produit bien un automate qui reconnaît le complémentaire de l'automate auquel elle a été appliquée (si l'automate de départ est complet). Nous considérons un AEFD A complet sur un alphabet Σ. Nous notons A^C l'automate obtenu en appliquant l'opération de complémentation sur A.

1. Montrer que $L(A^C) = \Sigma^* \setminus L(A)$.

Exercice 25 (♠♠♠) — Correction du produit d'automates

Soit $A = (Q^A, \Sigma, q_0^A, \delta^A, F^A)$ et $B = (Q^B, \Sigma, q_0^B, \delta^B, F^B)$ deux AEFD. L'objectif de cet exercice est de prouver que $L(A) \cap L(B) \subseteq L(A \times B)$.

1. Prouver que pour chaque $n \in \mathbb{N}$, pour chaque exécution $(q_0^A, u_0) \cdots (q_n^A, u_n)$ de A et $(q_0^B, u_0) \cdots (q_n^B, u_n)$ de B sur un mot commun u de longueur plus grande ou égale à n :

   $((q_0^A, q_0^B), u_0) \cdots ((q_n^A, q_n^B), u_n)$ est une exécution de $A \times B$.

2. Utiliser le résultat précédent pour prouver $L(A) \cap L(B) \subseteq L(A \times B)$.

CHAPITRE

5 Algorithmes et problèmes de décision sur les automates

Exercice 26 (♠) — Algorithmes pour déterminer la complétude

Considérons deux alphabets Σ et Σ' tels que $\Sigma' \subseteq \Sigma$. Nous rappelons qu'un automate sur un alphabet Σ est dit complet si sa fonction de transition est définie pour chaque symbole de Σ en chaque état.

1. Donner un algorithme qui détermine si un automate sur un alphabet Σ est complet.
2. Nous disons qu'un automate sur l'alphabet Σ est complet par rapport à l'alphabet Σ' si sa fonction de transition est définie en chaque état sur chaque symbole de Σ'. Donner un algorithme qui détermine si un automate sur l'alphabet Σ est complet sur l'alphabet Σ'.

Exercice 27 (♠) — Algorithmes pour complémenter

Considérons un alphabet Σ.

1. À partir de la définition de l'automate complémentaire vue en cours, donner un algorithme réalisant la négation d'un automate par rapport à Σ et produisant un automate reconnaissant le complémentaire du langage reconnu par l'automate passé en paramètre.

Exercice 28 (♠♠) — Algorithme pour calculer l'automate produit

Considérons un alphabet Σ.

1. À partir de la définition du produit d'automates vue en cours, donner un algorithme produisant un automate qui reconnaît l'intersection des langages des automates passés en paramètres. La construction de l'ensemble d'états de l'automate produit doit se faire « à la volée » c'est-à-dire en construisant les états de l'automate produit de manière paresseuse en parcourant simultanément les ensembles d'états des automates de départ.
2. Pour rappel, selon la définition d'automate produit vue en cours, si les automates passés en paramètres ont pour ensembles d'états Q_1 et Q_2, alors l'automate produit a pour ensemble d'états $Q_1 \times Q_2$ (avec $|Q_1 \times Q_2| = |Q_1| \times |Q_2|$). Donner des exemples d'automates tels que votre algorithme produise un automate dont le cardinal soit strictement inférieur à $|Q_1 \times Q_2|$.

Exercice 29 (♠♠♠) — Montrer l'inclusion entre langages

Considérons deux langages à états L et L' et les automates A_L et $A_{L'}$ reconnaissants L et L', respectivement.

1. En utilisant les opérateurs d'intersection et de complémentation entre langages, écrire une relation entre langages équivalente à $L \subseteq L'$.
2. Déduire de la question précédente, un algorithme utilisant A_L et $A_{L'}$ permettant de déterminer si $L \subseteq L'$.
3. Déduire de la question précédente, un algorithme utilisant A_L et $A_{L'}$ permettant de déterminer si $L = L'$.
4. Soit L_1 le langage des mots qui ne contiennent pas $abaa$ et qui contiennent un nombre pair de a. Soit L_2 le langage des mots qui ne contiennent pas aba et qui contiennent un nombre de a multiple de 4. Montrer que $L_2 \subseteq L_1$ en utilisant des AEFD qui reconnaissent ces langages.

Exercice 30 (♠♠♠) — Automates qui reconnaissent les langages prefixe-clos

Nous souhaitons montrer que le problème de déterminer si le langage reconnu par un automate est préfixe-clos est décidable.

1. Donner des exemples et des contre-exemples d'automates qui reconnaissent des langages préfixe-clos.

2. Caractériser par une condition les automates qui reconnaissent les langages préfixe-clos.
3. Donner un algorithme qui permet de décider si un langage défini par un AEFD est préfixe-clos.
4. Tester votre algorithme sur les automates de la première question.

CHAPITRE

6 Équivalence, distinguabilité et minimisation

Ce chapitre contient quelques exercices sur la minimisation et l'équivalence d'AEFD. Les deux chapitres suivants reviennent sur la notion de minimisation.

Exercice 31 (♠♠) — Minimiser des automates

Nous considérons les automates dans la Figure 6.1 sur l'alphabet $\Sigma = \{a, b\}$.

1. Minimiser l'AEFD dans la Figure 6.1a.
2. Minimiser l'AEFD dans la Figure 6.1b.

Exercice 32 (♠♠) — Déterminer l'équivalence ou la non-équivalence entre automates

Considérons les deux automates dans la Figure 6.2a et Figure 6.2b. Montrer que ces deux automates sont équivalents :

1. en utilisant la procédure basée sur la distinguabilité entre états ;
2. en utilisant la procédure basée sur la minimisation ;
3. en utilisant la procédure basée sur le produit d'automates.

Exercice 33 (♠♠) — Déterminer si un automate est minimal ou non

Nous considérons les deux automates dans la Figure 6.3. Nous souhaitons déterminer si ces automates sont minimaux. Pour chacun des automates (Figure 6.3a et Figure 6.3b), faire les questions suivantes.

1. Utiliser l'algorithme calculant les états distinguables pour déterminer la minimalité.
2. Utiliser l'algorithme calculant les états équivalents pour déterminer la minimalité. Montrer clairement les étapes de calcul. Lister les classes d'équivalences.
3. Vérifier que vous obtenez des résultats compatibles et donc les mêmes conclusions avec les deux méthodes.
4. Représenter l'automate minimal sous formes tabulaire et graphique.

TABLE 6.1 – Des AEFD en représentation tabulaire à minimiser. Les états sont en lignes, les symboles en colonnes. La flèche indique l'état initial, l'étoile indique un état terminal.

		a	b
→	0	0	1
	1	2	3
	2	2	3
*	3	2	4
	4	0	1

(a)

		a	b
→	1	3	8
*	2	3	1
	3	8	2
*	4	5	6
	5	6	2
	6	7	8
	7	6	4
	8	5	8

(b)

		a	b
→ *	0	1	3
	1	2	3
*	2	5	2
	3	4	1
*	4	5	4
	5	5	5

(c)

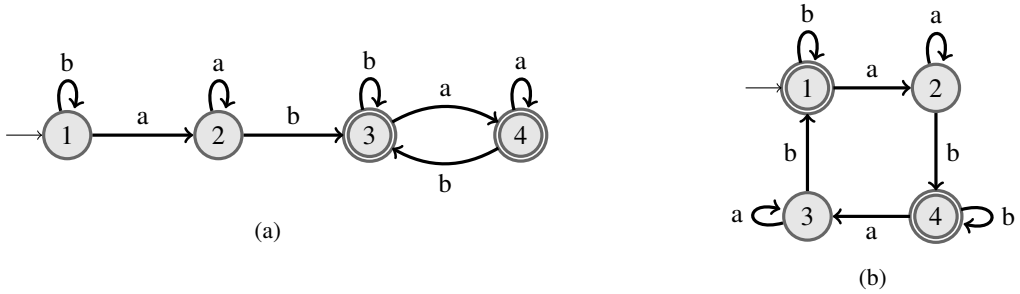


FIGURE 6.1 – Des AEFD en représentation graphique à minimiser.

FIGURE 6.2 – Des AEFD dont on veut déterminer l'équivalence.

FIGURE 6.3 – Des AEFD en représentation graphique à minimiser.

CHAPITRE

7 Automates à états finis non-déterministes

Rappel :
- AEFD : automate à états fini déterministe ;
- AEFND : automate à états fini non-déterministe.

Remarque : Lorsqu'on représentera un automate par sa table de transitions, nous utiliserons les conventions suivantes :
- l'état initial sera indiqué par une flèche,
- les états marqués d'une étoile sont accepteurs.

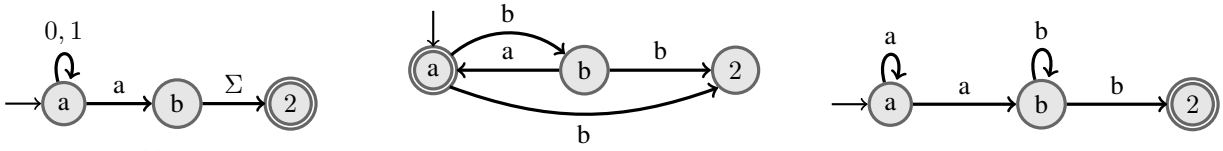


FIGURE 7.1 – Des automates à états finis non déterministes

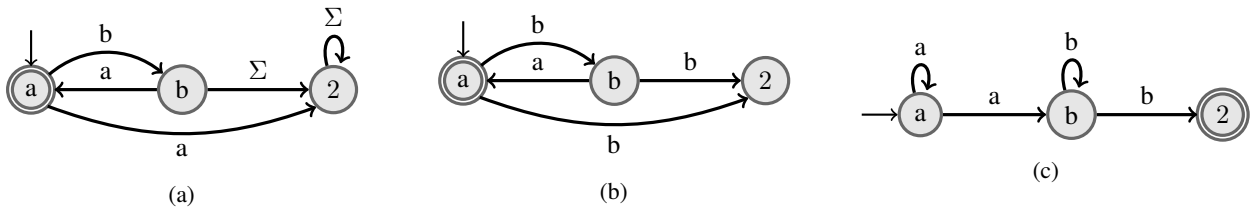


FIGURE 7.2 – Des automates à états finis non déterministes

Exercice 34 (♠) — Décrire le langage reconnu par un AEFND

Décrire en langage naturel chacun des automates représentés dans la Figure 7.1.

Exercice 35 (♠) — Donner un AEFND qui reconnait un langage

Nous considérons l'alphabet $\Sigma = \{a, b\}$. Nous définissons le langage L_i comme l'ensemble des mots de Σ^* tels que le i-ème symbole en partant de la fin des mots est le symbole a. (Le premier symbole d'un mot en partant de la fin est le dernier symbole.)

1. Définir formellement le langage L_i pour $i \in \mathbb{N}$.
2. Donner un AEFND qui reconnait L_1.
3. Donner un AEFND qui reconnait L_2.
4. Donner un AEFND qui reconnait L_3.

Exercice 36 (♠♠) — Déterminiser

Considérons les deux automates suivants :
- A l'AEFND défini par $(\{0, 1, 2, 3, 4, 5\}, \{a, b\}, 0, \Delta, \{4\})$ dont la relation de transition est définie par la Table 7.1a.
- B l'AEFND défini par $(\{0, 1, 2, 3, 4, 5\}, \{a, b\}, 0, \Delta, \{0, 3, 4\})$ dont la relation de transition est définie par la Table 7.1b.

TABLE 7.1 – Des automates à états finis non déterministes représentés sous forme tabulaire

	→ 0	1	2	3	4*	5
a	1,2,3,4,5	2,3	0,1,4	0	1	2
b		4	1,2,3	1,2,5		2,3,5

(a)

	→ 0*	1	2	3*	4*	5
a	1,2			5		
b		3	4			0

(b)

— C l'AEFND défini par $(\{1,2,3,4,5,6\},\{a,b\},1,\Delta,\{2\})$ tel que :

$$\Delta = \{(1,a,2),(2,a,3),(3,a,2),(2,a,4),(4,b,2),(2,b,5),(5,a,2),(2,b,6),(6,b,2)\}.$$

1. Déterminiser A et minimiser l'AEFD obtenu après déterminisation.
2. Même question pour l'automate B.
3. Même question pour l'automate C.

Exercice 37 (♠♠) — Déterminer l'équivalence

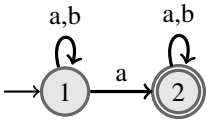
Soit $\Sigma = \{0,1\}$. Considérons les deux premiers AEFNDs représentés dans la Figure 7.2.

1. Quel est le langage reconnu par l'automate représenté dans la Figure 7.2a ?
2. Quel est le langage reconnu par l'automate représenté dans la Figure 7.2b ?
3. Montrer que ces deux automates sont équivalents.

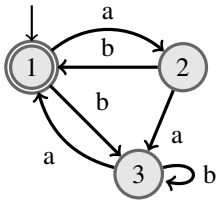
Exercice 38 (♠♠♠) — Nombre d'exécutions acceptées

Nous nous intéressons à calculer le nombre $\mathcal{N}(A,u)$ d'exécutions acceptées par un automate A pour un mot u donné en entrée. Rappelons que $|u|_a$ dénote le nombre d'occurrences du symbole a dans le mot u.

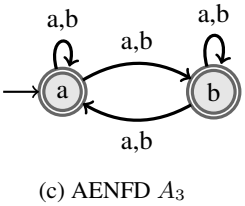
1. Considérons l'automate A_1 représenté dans la Figure 7.3a, lister les exécutions acceptées et associées au mot $abaa$. En déduire $\mathcal{N}(A_1,abaa)$.
2. Montrer que $\mathcal{N}(A_1,u) = |u|_a$.
3. Considérons l'automate A_2 représenté dans la Figure 7.3b, quelle est la valeur de $\mathcal{N}(A_2,u)$ pour un mot u. Justifier.
4. Considérons l'automate A_3 représenté dans la Figure 7.3c, quelle est la valeur de $\mathcal{N}(A_3,u)$ pour un mot u. Justifier.
5. Considérons deux AFENDs $Auto_1$ et $Auto_2$ sur l'alphabet Σ et l'AEFND $Auto$ résultant du produit pour intersection de $Auto_1$ et $Auto_2$. Montrer que $\forall u \in \Sigma^* : \mathcal{N}(Auto,u) = \mathcal{N}(Auto_1,u) \times \mathcal{N}(Auto_2,u)$.
6. Donner un automate A tel que $\forall u \in \Sigma^* : \mathcal{N}(A,u) = (|u|_a)^2$.



(a) AENFD A_1



(b) AENFD A_2



(c) AENFD A_3

FIGURE 7.3 – Des automates à états finis non déterministes

CHAPITRE

8 Automates à états finis non-déterministes avec ϵ-transitions

Rappel : ϵ-AEFND : automate à états fini non-déterministe avec ϵ-transitions.

Exercice 39 (♠) — Montrer qu'un langage est un langage à états

Soit L un langage à états sur un alphabet Σ et A un automate qui reconnait L.

1. En utilisant A, définir un automate qui reconnait $\{w \mid s \cdot w \in L\}$, l'ensemble des mots de L commençant par un symbole $s \in \Sigma$ et le supprimant, est un langage à états.
2. En utilisant A, définir un automate qui reconnait $\{w \mid w \cdot s \in L\}$, l'ensemble des mots non vide de L terminant par un symbole $s \in \Sigma$ et le supprimant, est un langage à états.
3. En utilisant A, définir un automate qui reconnait $\{w \cdot s \cdot s \mid w \cdot s \in L\}$, l'ensemble des mots obtenus un doublant la dernière lettre des mots non vide de L, est un langage à états.
4. En utilisant A, définir un automate qui reconnait $\{w_1 \cdot s \cdot w_2 \mid w_1, w_2 \in L\}$, l'ensemble des mots obtenu en insérant une occurence d'un symbole $s \in \Sigma$ dans un mot de L, est un langage à états.

Exercice 40 (♠) — Composition d'ϵ-AEFNDs

Soient $A_1 = (Q_1, q_1^0, \Sigma, \Delta_1, F_1)$ et $A_2 = (Q_2, q_2^0, \Sigma, \Delta_2, F_2)$ deux ϵ-AEFNDs et L_1, L_2 les langages reconnus par A_1 et A_2 respectivement.

1. Définir l'automate $A_\cup = (Q_\cup, q_\cup^0, \Sigma, \Delta_\cup, F_\cup)$ qui reconnaît $L_1 \cup L_2$.
2. Définir l'automate $A_\cdot = (Q_\cdot, q_\cdot^0, \Sigma, \Delta_\cdot, F_\cdot)$ qui reconnaît $L_1 \cdot L_2$.
3. Définir l'automate $A_* = (Q_*, q_*^0, \Sigma, \Delta_*, F_*)$ qui reconnaît L_1^*.
4. Définir l'automate $A_\cap = (Q_\cap, q_\cap^0, \Sigma, \Delta_\cap, F_\cap)$ qui reconnaît $L_1 \cap L_2$.

Exercice 41 (♠♠) — Élimination des ϵ-transitions et déterminisation

Considérons l'alphabet $\Sigma = \{a, b\}$ et l'ϵ-AEFND défini par $(\{0, 1, 2, 3, 4\}, \{a, b\}, 0, \Delta, \{4\})$ dont la relation de transition Δ est définie par la Table 8.1a. Pour les questions suivantes, utiliser la représentation tabulaire des automates.

1. Éliminer les ϵ-transitions.
2. Déterminiser l'automate obtenu à la question précédente.
3. Déterminiser l'automate initial en utilisant la méthode directe (combinaison de l'élimination des ϵ-transitions et déterminisation).

	0	1	2	3	4
ϵ	1, 3		4		
a		2		4	
b			1		3

(a)

	0	1	2	3	4
ϵ	1, 3	3	1		3
a		2			
b			4		

(b)

TABLE 8.1 – Des automates à états finis non déterministes avec ϵ-transitions

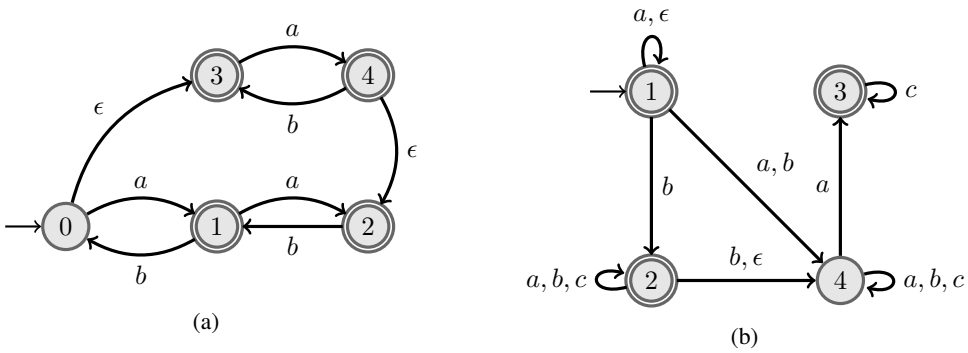


FIGURE 8.1 – Des automates à états-finis non déterministes avec ϵ-transitions

4. Vérifier que les automates obtenues aux deux questions précédentes (c'est-à-direen utilisant les deux méthodes) sont équivalents.

Exercice 42 (♠♠) — Élimination des ϵ-transitions et déterminisation

Soit l'ϵ-AEFND défini par $(\{0,1,2,3,4\},\{a,b\},0,\Delta,\{4\})$ dont la relation de transition Δ est définie par la Table 8.1b. Pour les questions suivantes, utiliser la représentation tabulaire des automates.

1. Éliminer les ϵ-transitions.
2. Déterminiser l'automate obtenu à la question précédente.
3. Déterminiser l'automate initial en utilisant la méthode directe (combinaison de l'élimination des ϵ-transitions et déterminisation).
4. Vérifier que les automates obtenues aux deux questions précédentes (c'est-à-dire en utilisant les deux méthodes) sont équivalents.

Exercice 43 (♠♠) — Élimination des ϵ-transitions et déterminisation

Considérons l'alphabet $\Sigma = \{a,b\}$ et les ϵ-AEFND représentés dans la Figure 8.1a et Figure 8.1b, définis sur Σ. Pour les questions suivantes, utiliser la représentation sous forme de graphes des automates.

1. Donner un mot accepté et un mot non-accepté par l'automate.
2. Éliminer les ϵ-transitions et déterminiser l'automate obtenu.
3. Minimiser l'automate obtenu à la question précédente.

Exercice 44 (♠♠♠) — Langage miroir

Soit Σ un alphabet. L'image miroir $R(u)$ d'un mot u est le mot que l'on obtient en lisant le mot u de droite à gauche (comme en Arabe ou en Hébreu). Plus précisément :

— $R(\epsilon) = \epsilon$,
— $R(u \cdot a) = a \cdot R(u)$, pour tout $u \in \Sigma^*, a \in \Sigma$.

Soit L un langage à états.

1. Prouver que $R(L) = \{R(u) \mid u \in L\}$ est un langage à états.

Exercice 45 (♠♠♠) — Fermeture de Kleene, union et inclusion

1. Soient L_1 et L_2 des langages. Montrer que si $L_1 \subseteq L_2$, alors $L_1^* \subseteq L_2^*$.
2. En déduire que pour tout langages L_1 et L_2, $L_1^* \cup L_2^* \subseteq (L_1 \cup L_2)^*$.
3. Montrer que, en général, $L_1^* \cup L_2^* \neq (L_1 \cup L_2)^*$.
4. Trouver des langages L_1 et L_2 tels que $L_1 \not\subseteq L_2$, $L_2 \not\subseteq$ àL_1 et $L_1^* \cup L_2^* = (L_1 \cup L_2)^*$

CHAPITRE

9 Expressions régulières

Exercice 46 (♠) — Simplifier des expressions régulières

Considérons l'alphabet $\Sigma = \{a, b\}$. Simplifier chacune des expressions régulières suivantes, c'est-à-dire trouver une expression régulière équivalente qui contient moins de symboles.

1. $\epsilon + a \cdot b + a \cdot b \cdot a \cdot b \cdot (a \cdot b)^*$.
2. $a \cdot a(b^* + a) + a \cdot (a \cdot b^* + a \cdot a)$
3. $a \cdot (a + b)^* + a \cdot a \cdot (a + b)^* + a \cdot a \cdot a \cdot (a + b)^*$.

Exercice 47 (♠♠) — Trouver des expressions régulières

Considérons l'alphabet $\Sigma = \{a, b, c\}$. Pour chacun des langages suivants sur Σ, donner une expression régulière qui le dénote.

1. L'ensemble des mots qui commencent par a et finissent par b.
2. L'ensemble des mots qui contiennent au moins trois occurrences du symbole b.
3. L'ensemble des mots qui contiennent au moins trois occurrences consécutives du symbole b.
4. L'ensemble des mots qui contiennent un nombre pair de a.
5. L'ensemble des mots qui contiennent un nombre impair de a.
6. L'ensemble des mots qui contiennent un nombre de a multiple de 3.
7. L'ensemble des mots qui ne contiennent pas le facteur $a \cdot a$.
8. L'ensemble des mots qui ne contiennent pas le facteur $a \cdot a \cdot b$.
9. L'ensemble des mots qui contiennent au moins 2 occurrences du symbole a, mais non consécutives.
10. L'ensemble des mots qui contiennent au moins 3 symboles et le troisième symbole est a.
11. L'ensemble des mots qui commencent et finissent par le même symbole.
12. L'ensemble des mots de longueur impaire.
13. L'ensemble des mots de longueur au moins 1 et au plus 3.

Exercice 48 (♠♠♠) — Équivalence ou non entre expressions régulières

Donner une preuve ou un contre-exemple pour les lois algébriques suivantes sur les expressions régulières :

1. $(\epsilon + R)^* \equiv R^*$.
2. $(\epsilon + R) \cdot R^* \equiv R^*$.
3. $(R + S)^* \equiv R^* + S^*$.
4. $(RS + R)^* R \equiv R(SR + R)^*$.
5. $(RS + R)^* RS \equiv (RR^* S)^*$.
6. $(R + S)^* S \equiv (R^* S)^*$.
7. $S(RS + S)^* R \equiv RR^* S(RR^* S)^*$.

Exercice 49 (♠♠♠) — Lemme d'Arden

Soient $E, F, X \subseteq \Sigma^*$ des langages.

1. Prouver que le langage $E^* F$ est une solution de l'équation $X = EX + F$.
2. Prouver que si $\epsilon \notin E$, alors $E^* F$ est la solution unique de $X = EX + F$ (Lemme d'Arden).

CHAPITRE

10 Théorème de Kleene

Exercice 50 (♠♠) — Automate vers expression régulière

Nous souhaitons calculer les expressions régulières associées aux automates dans la Figure 10.2.

1. Calculer les expressions régulières en suivant la méthode associant des équations linéaires aux états.
2. Calculer les expressions régulières en suivant la méthode associant des expressions régulières aux chemins.

Exercice 51 (♠♠) — Expression régulière vers automate

Soit $\Sigma = \{a, b\}$. Donner les ϵ-AEFND associés aux expressions régulières suivantes. Pour les trois premières expressions régulières, utiliser la méthode compositionelle de Thompson et la méthode par calcul des dérivées.

1. $a \cdot b$,
2. $a^* \cdot b$,
3. $(a+b)^* \cdot a^* \cdot b^*$,
4. $\big(a^* \cdot b + d \cdot c\big)^* \cdot \big(b^* \cdot d + a \cdot d\big)^*$,
5. $\big(a \cdot b + a^* \cdot b + c \cdot d\big) \cdot \big((c \cdot a^* + b \cdot d) \cdot (a \cdot b^* + a \cdot b \cdot d)\big)^*$.

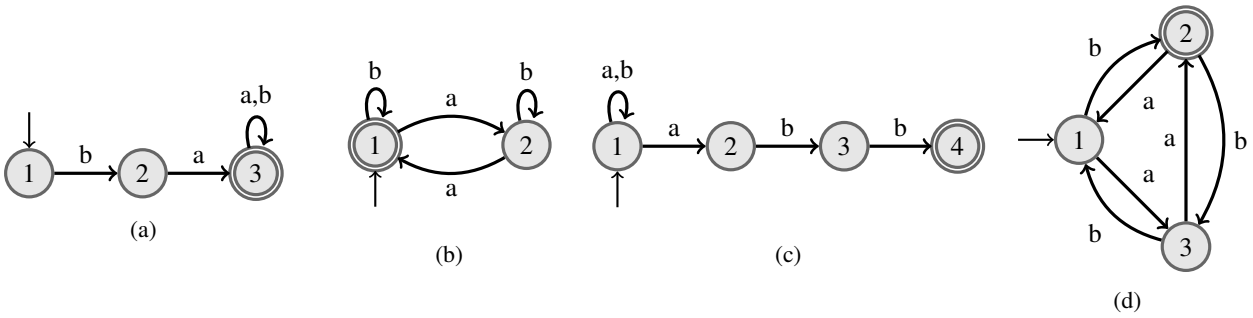
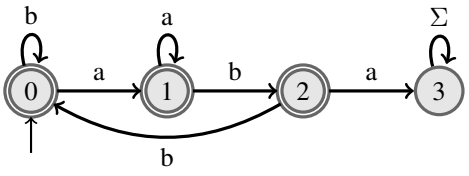
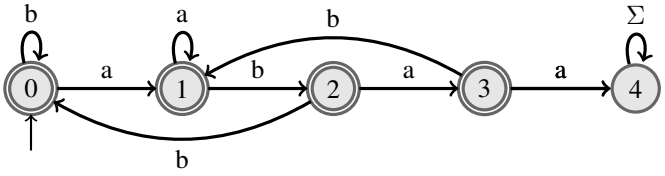


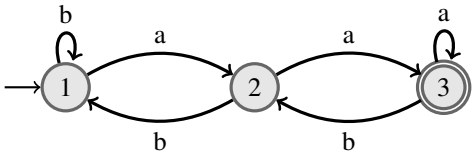
FIGURE 10.1 – Des automates pour le calcul d'expressions régulières



(a) Automate reconnaissant les mots ne contenant pas aba.



(b) Automate reconnaissant les mots ne contenant pas $abaa$.



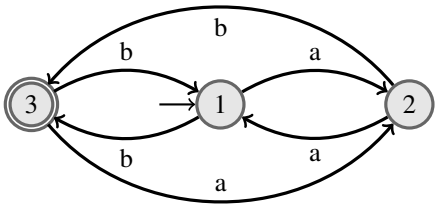


FIGURE 10.2 – Des automates pour le calcul d'expressions régulières

CHAPITRE

11 Grammaires

$$
\begin{array}{lcl}
S & \to & abS, \\
S & \to & bcS, \\
S & \to & bbS, \\
S & \to & a, \\
S & \to & cb
\end{array}
$$

FIGURE 11.1 – Règles de production de la grammaire pour l'Exercice 52.

Exercice 52 (♠) — Dérivation de grammaire

Considérons la grammaire $G = (\{a, b, c\}, \{S\}, S, P)$ où P est donné par les règles dans la Figure 11.1. Donner une dérivation de G pour les mots suivants :

1. $bcbba$
2. $bbbcbba$
3. $bcabbbbcb$

Exercice 53 (♠) — Mots générés par une grammaire

Considérons la grammaire $G = (V_T, V_{\mathrm{NT}}, S, P)$, où :

— V_T est l'ensemble des symboles terminaux suivants :

$$\{un, une, l', etudiante, etudiant, enseignante, enseignant, dutennis, duski, descours, faitdu, etudie, donne\}$$

— P est l'ensemble des règles de production suivantes :

$$
\begin{array}{lcllcllcl}
PH & \to & GN\,GV & GN & \to & AF\,NF \mid AM\,NM & GV & \to & VC \\
V & \to & fait \mid etudie \mid donne & NF & \to & etudiante \mid enseignante & NM & \to & etudiant \mid enseignant \\
AF & \to & une \mid l' & AM & \to & un \mid l' & C & \to & dutennis \mid duski \mid descours
\end{array}
$$

1. Donner quelques phrases générées par la grammaire.
2. Donner quelques phrases non générées par la grammaire.
3. Donner le nombre de phrases générées par cette grammaire. Il n'est pas requis que les phrases aient du sens.

Exercice 54 (♠♠) — Langage généré par une grammaire

Quels sont les langages générés par les grammaires de la forme $(\{a, b\}, \{S, A, B\}, S, P)$ avec les ensembles de règles de production suivants P :

1. $\left\{ \begin{array}{lcl} S & \rightarrow & AB, \\ A & \rightarrow & ab, \\ B & \rightarrow & BB \end{array} \right\}$;
2. $\left\{ \begin{array}{lcl} S & \rightarrow & AB \mid aA, \\ A & \rightarrow & a, \\ B & \rightarrow & b \end{array} \right\}$;
3. $\left\{ \begin{array}{lcl} S & \rightarrow & AB \mid AA, \\ A & \rightarrow & aB, \\ B & \rightarrow & b \end{array} \right\}$;
4. $\left\{ \begin{array}{lcl} S & \rightarrow & AA \mid B, \\ A & \rightarrow & aaA \mid aa, \\ B & \rightarrow & bB \mid B \end{array} \right\}$;
5. $\left\{ \begin{array}{lcl} S & \rightarrow & AB \mid aAb, \\ B & \rightarrow & bBa \mid \epsilon, \\ A & \rightarrow & \epsilon \end{array} \right\}$.

CHAPITRE

12 Grammaires régulières

Exercice 55 (♠♠) — Composition de grammaires

Soient L et L' des langages réguliers générés par les grammaires G et G' respectivement. On souhaite prouver qu'il existe des grammaires régulières qui génèrent les langages suivants

$$L \cup L' \qquad\qquad L \cdot L' \qquad\qquad (L)^*$$

1. À partir d'exemples de grammaires, montrer comment construire ces grammaires.
2. Généraliser. Donner les grammaires pour les langages demandés à partir de grammaires régulières quelconques. Expliquer la construction de ces grammaires.

Exercice 56 (♠♠) — Grammaire vers automate

Dans cet exercice, nous transformons des grammaires en automates équivalents.

1. Donner des automates qui reconnaissent les langages décrits par les grammaires données dans l'Exercice **??**, si cela est possible. C'est à dire pour les grammaires de la forme $(\{a, b\}, \{S, T, U\}, S, P)$ avec les ensembles de règles de production suivants P :
   — $\left\{ \begin{array}{lcl} S & \to & T \mid bSb, \\ T & \to & aT \mid \epsilon \end{array} \right\}$;
   — $\left\{ \begin{array}{l} S \to T \mid bS, \\ T \to aT \mid \epsilon \end{array} \right\}$.
2. Si cela est possible, donner des automates qui reconnaissent les langages décrits par les grammaires données dans l'Exercice 54. Si cela n'est pas possible, justifier.

Exercice 57 (♠♠) — Des AEFD et AEFND vers les grammaires

Donner des grammaires générant les langages reconnus par les AEFD donnés dans les figures suivantes.

1. Les AEFD dans la Figure 3.1.
2. Les AEFD dans la Figure 10.1.
3. Les AEFD non-minimaux dans la Figure 6.1.
4. Les AEFNDs dans la Figure 7.1.
5. Les AEFNDs dans la Figure 7.2.

CHAPITRE

13 Langages non réguliers et lemme de l'itération

Exercice 58 (♠) — Trouver la constante d'itération minimale

Considérons l'alphabet $\Sigma = \{0, 1\}$. Donner la constante d'itération minimale des langages réguliers sur Σ dénotés par les expressions régulières suivantes :

1. 0001^*.
2. 0^*.
3. 0^*1^*.
4. 1011.
5. $(01)^*$.
6. ϵ.
7. $(0 + 1)^*$.
8. 10^*1.

Exercice 59 (♠♠) — Trouver la constante d'itération minimale

Considérons l'alphabet $\Sigma = \{0, 1\}$. Donner la constante d'itération minimale des langages réguliers sur Σ dénotés par les expressions régulières suivantes :

1. $10(11^*0)^*0$.
2. $011 + 0^*1^*$.
3. $0^*1^+0^+1^*$.
4. $0^*1^+0^+1^* + 10^*1$.
5. 0^*101^*.
6. $0^*101^* + 101^*$.

Exercice 60 (♠♠♠) — Lien entre deux langages

Nous considérons l'alphabet $\Sigma = \{a, b\}$ et les langages
- $L_1 = \{u \cdot v \cdot w \mid u, w \in \Sigma^*, v \in \{aaa, bbb\}\}$, et
- $L_2 = \{(aab)^n \cdot (abb)^n \mid n \in \mathbb{N}\}$.

1. Est-ce que le langage L_1 est un langage à états ? Si oui, donner un automate reconnaissant L_1.
2. Remarquer une propriété sur la longueur des mots de L_2 ?
3. Donner les facteurs n'apparaissant pas dans L_2.
4. Déduire de la question précédente une relation entre L_1 et L_2.
5. Est-ce que le langage L_2 est un langage à états ? Donner une preuve.
6. Est-ce que le langage $L_1 \cup L_2$ est un langage à états ? Donner une preuve.

CHAPITRE

14 Démontrer la non-régularité d'un langage

Dans ce chapitre, nous considérons les alphabets $\Sigma_1 = \{a, b, c\}$ et $\Sigma_2 = \{a, b\}$. Dans les exercices suivants, il faut montrer que les langages proposés ne sont pas réguliers en utilisant le lemme de l'itération.

Exercice 61 (♠) — Utilisation du lemme de l'itération

Prouver que les langages suivants ne sont pas réguliers.

1. $\{a^n b^{n+1} \mid n \in \mathbb{N}\}$;
2. $\{a^n b^{2\times n} \mid n \in \mathbb{N}\}$;
3. $\{a^{2\times i} b^{2\times i} \mid i \in \mathbb{N}\}$;
4. $\{a^i b^j c^{i+j} \mid i, j \in \mathbb{N}\}$;

Exercice 62 (♠♠) — Utilisation du lemme de l'itération

Prouver que les langages suivants ne sont pas réguliers.

1. $\{w \in \Sigma_1^* \mid |w|_a = |w|_b + |w|_c\}$.
2. $\{a^{2\times i} \cdot (b \cdot c)^i \mid i \in \mathbb{N}\}$.
3. $\{w \cdot w \cdot w \mid w \in \Sigma_2^*\}$.

Exercice 63 (♠♠) — Régulier ou non régulier ?

Considérons l'alphabet $\Sigma = \{a, b\}$. Dire si les langages suivants sur Σ sont réguliers ou non. Justifier votre réponse en donnant un automate à états fini ou en utilisant le lemme de l'itération.

1. $\{a^n 1 b^n \mid n \in \mathbb{N}\}$
2. $\{a^n a^n \mid n \in \mathbb{N}\}$
3. $\{w \cdot w^R \mid w \in \Sigma^*\}$
4. $\{w \cdot u \cdot w^R \mid w \in \Sigma^*, u \in \Sigma^+\}$.

Exercice 64 (♠♠) — Utilisation des propriétés de fermeture pour montrer la non-régularité

Considérons un alphabet Σ. Dans cet exercice, il faut utiliser les propriétés de fermeture des langages réguliers et utiliser les résultats de non-régularité des autres exercices. Prouver que les langages suivants ne sont pas réguliers.

1. $\{u \in \Sigma^* \mid |u| \text{ est premier}\}$.
2. $\{u \in \Sigma^* \mid |u| \text{ est un carre}\}$.
3. $\{0^i 1^j 2^{i+j} \mid i, j \in \mathbb{N}\}$.
4. $\{0^{2\times i+1} 1^{2\times i+1} \mid i \in \mathbb{N}\}$.
5. $\{a^m b^k a^n \mid m, k, n \in \mathbb{N}, m \neq n\}$.
6. $\{w \in \Sigma^* \mid w \neq w^R\}$.

Exercice 65 (♠♠♠) — Correct ou incorrect

For chacune des propositions suivantes, dire si elle est correcte ou non. Si elle est correcte, donner une preuve. Si elle est incorrecte, donner un contre-exemple.

1. Si $A \cup B$ est régulier et A est régulier, alors B est régulier.
2. Si $A \cap B$ est régulier et A est régulier, alors B est régulier.
3. Si $A \cup B$ est non-régulier et A est non-régulier, alors B est non-régulier.
4. Si $A \cap B$ est non-régulier et A est non-régulier, alors B est non-régulier.

5. Si $A \cup B$ est non-régulier et A est régulier, alors B est non-régulier.
6. Si $A \cap B$ est non-régulier et A est régulier, alors B est non-régulier.
7. Si A est régulier et B est non-régulier, alors $A \cup B$ est non-régulier.
8. Si A est régulier et B est non-régulier, alors $A \cap B$ est non-régulier.

Exercice 66 (♠♠♠) — Utilisation du lemme de l'itération

Prouver que les langages suivants ne sont pas réguliers.

1. $\{a^i \mid i \text{ est un carré}\}$;
2. $\{a^i \mid i \text{ est un cube}\}$;
3. $\{a^i \mid i \text{ est une factorielle}\}$;
4. $\{a^i \mid i \text{ est premier}\}$.
5. $\{w \in \Sigma_2^* \mid |w|_a/|w|_b \in \mathbb{N}\}$.
6. $\{w \in \Sigma_2^* \mid |w|_a/|w|_b \in \mathbb{N} \text{ et est premier}\}$.

Exercice 67 (♠♠♠) — Un langage non-régulier qui satisfait le lemme de l'itération

Soit X un langage non régulier sur l'alphabet $\Sigma = \{a, b\}$. Nous considérons les langages suivants sur Σ :

— $A = \{aba\} \cdot \Sigma^*$
— $B = \overline{A}$
— $C = (\{aba\} \cdot X) \cup B$

1. Montrer que C satisfait le lemme de l'itération
2. Montrer que C est non régulier.

ANNEXE

A Modélisation et résolution de problèmes avec les automates

Exercice 68 (♠♠♠) — Code pour le contrôle l'accès

Considérons l'alphabet formé par les chiffres utilisés en notation décimale : $\Sigma = \{0, \ldots, 9\}$. Nous souhaitons modéliser des variantes d'automates qui permettent de reconnaitre un code sous la forme $x \cdot y \cdot z$ avec $x, y, z \in \Sigma$. Pour les questions suivantes, considérer des alternatives dépendant de la condition déterminant qu'un code a été entré. Par exemple, il est possible de considérer un bouton pour valider sa saisie ou que chaque séquence de trois chiffres correspond à une saisie.

1. L'automate ne laisse qu'une seule chance.
2. L'automate ne laisse que deux chances.
3. L'automate ne laisse que deux chances. L'automate permet d'annuler sa saisie grâce à un bouton spécial.
4. L'automate accepte dès que les derniers chiffres entrés correspondent au code.

Exercice 69 (♠♠♠) — Tennis

Au tennis les points sont comptés de la manière suivante : 15, 30, 40. Jusqu'à 40 les points sont comptés de façon incrémentale. Lorsqu'au moins un des deux joueurs atteint le score 40, si l'autre joueur a un score strictement inférieur et que le joueur a 40 marque un autre point, il remporte le jeu. Si les deux joueurs atteignent le score 40, alors le joueur remportant le point suivant obtient l'avantage. Si le joueur avec l'avantage marque un nouveau point il remporte le point. Sinon, si l'autre joueur marque un point, ils reviennent tous les deux au score de 40.

1. Donner un automate qui compte les points au tennis. On pourra utiliser l'état pour contenir le score et un alphabet à deux symboles, chaque symbole correspondant à la victoire d'un point.

Exercice 70 (♠♠♠) — Machine à café

Nous souhaitons modéliser une machine à boissons simplifiée et son interaction avec des clients. La machine sert plusieurs types de boissons café, thé et chocolat. Toutes les boissons ont le même prix : 1 jeton. La machine doit laisser à l'utilisateur le choix de la boisson. Elle doit délivrer une boisson lorsque l'utilisateur a acquitté le prix de la boisson et effectué son choix. La machine doit laisser au client la possibilité de récupérer ses jetons s'il n'y a pas encore confirmé son choix de boisson. L'utilisateur peut insérer des jetons, sélectionner une boisson, récupérer sa boisson, récupérer son jeton, demander l'annulation du service.

1. Déterminer les actions de l'utilisateur et modéliser son comportement à l'aide d'un automate.
2. Déterminer les actions de la machine et modéliser son comportement à l'aide d'un automate.
3. Comment obtenir le comportement global (observable) du système ? Décrire certains de ces comportements.
4. Nous souhaitons maintenant vérifier le modèle de la machine, en présence d'utilisateur. Supposons qu'une propriété soit représentée par un langage L_{prop} sur l'union des alphabets de la machine et de l'utilisateur. Nous considérons le langage L_{sys} des mots représentant les différentes exécutions de la machine en présence de l'utilisateur. Pour chacune des questions suivantes, donner une relation entre L_{prop} et L_{sys} qui est satisfaite si et seulement si la réponse à la question est affirmative.
   — Tous les comportements du système respectent la propriété.
   — Il est possible que le système réalise l'un des comportements de la propriétés.
   — Le système ne respecte pas la propriété.
5. Afin de detecter les potentielles erreurs de modélisation et en utilisant les algorithmes vus en cours, déterminer comment vérifier les propriétés suivantes sur le comportement global du modèle :

— Est-ce que le client peut obtenir une boisson sans avoir inséré de pièce ?
— Est-ce que le client peut, après avoir inséré des pièces et choisi une boisson, ne pas obtenir de boisson ?
— Est-ce que le client peut obtenir une boisson alors qu'il en a choisi une autre ?
— Est-ce que le système peut se bloquer ?

6. Changer les comportements du client et de la machine et revisiter les questions précédentes. Par exemple, pour la machine, vous pouvez complexifier le comportement en le rendant plus réaliste (par exemple en considérant des pièces au lieu de jetons ou alors en prenant en compte le rendu de monnaie) ou y introduire des failles ; pour le client vous pouvez considérer un client malicieux.

Exercice 71 (♠♠♠) — Une histoire de berger, de loup, de chèvre et de choux

Monsieur Berger B emmène un loup L, une chèvre C, et un choux X près d'une rivière et souhaite traverser avec un petit bateau. Le bateau est tellement petit que B peut entrer dans le bateau avec au plus un passager. Sans surveillance de B, L mange C et C mange X.
Nous souhaitons trouver comment B peut faire traverser la rivière à la compagnie sans perdre d'élément ?

1. Déterminer l'ensemble des états de l'automate qui représente les différentes situations des deux cotés de la rivière.
2. Donner un automate qui modélise la situation.
3. Utiliser l'automate trouvé à la question précédente pour trouver comment le problème peut être résolu de manière algorithmique.

Exercice 72 (♠♠♠♠) — Étrange planète

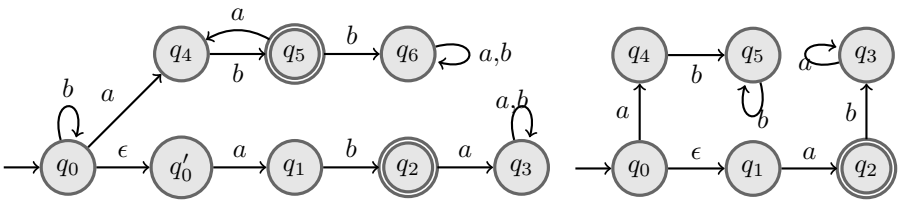
Nous reprenons la modélisation de l'étrange planète vue dans le cours d'introduction. Sur cette planète trois espèces cohabitent. Nous appellerons les espèces rouge, bleue et orange. Nous souhaitons modéliser l'évolution de la population de ces espèces selon leur mode de reproduction, qui suit les règles suivantes :
— 2 individus de deux *espèces différentes* peuvent s'unir ;
— la reproduction tue les deux individus ;
— la reproduction génère 2 individus de la troisième espèce.
Par exemple : 1 rouge et 1 bleue $\rightarrow$ 2 orange. Nous supposons également que :
— des individus peuvent s'unir juste après avoir été générés (pas de distinction adulte/enfant) ;
— les individus ne peuvent mourir que lors de la reproduction.

1. Trouver une symétrie dans les états pour un nombre d'individus fixé.
2. Exprimer une condition pour l'arrêt de l'évolution sur l'ensemble des individus (et non la liste des individus).
3. Proposer une nouvelle représentation de l'état basée sur les observations faites dans les deux questions précédentes.
4. Revisiter les automates du cours pour une planète avec 2, 3 et 4 individus.
5. Proposer un automate pour une planète avec 5 individus.
6. De manière générale, combien d'états l'automate d'une planète avec n individus contient-il (avec ou sans symétrie) ?
7. Pour un état courant donné, comment déterminer si :
   a) L'évolution s'arrêtera inévitablement.
   b) L'évolution ne peut pas s'arrêter.
   c) L'évolution peut s'arrêter.

Exercice 73 (♠♠♠♠) — Opacité d'un système

Dans cet exercice, nous nous intéressons à une propriété importante en sécurité des systèmes informatiques : l'*opacité*. Le contexte est le suivant. Nous supposons qu'un attaquant observe un système dont le comportement est modélisé par un AEFND avec ϵ-transitions. Les états accepteurs de l'automate représentent le "secret" : lors d'une exécution du système, l'attaquant ne doit pas être en mesure de savoir avec certitude que le système est dans un état secret. Si lors d'une exécution du système, l'attaquant est en mesure de déterminer que le système est dans un état secret, alors on dit que cette exécution *révèle le secret*. Un système est dit *opaque* s'il n'existe pas d'exécution qui révèle le secret. L'attaquant observe le système à travers une

"fenêtre d'observation" qui lui permet de voir toutes les transitions exceptées les ϵ-transitions. L'attaquant connait la structure de l'automate parfaitement.



1. Nous considérons le système représenté par l'automate de gauche ci-dessus. Lorsque l'attaquant observe a, b, ab, quels sont les états courants possibles du système ?
2. Dire si ce système est opaque.
3. Même questions avec le système modélisé par l'automate de droite ci-dessus.
4. Est-il possible à partir de l'automate modélisant le système, de construire un automate qui indique la connaissance de l'attaquant en fonction de son observation ?

Exercice 74 (♠♠♠) — Récipients

Nous considérons la situation où nous disposons d'une arrivée d'eau (supposée infinie) et deux récipients de 3 et 5 litres. Nous souhaitons utiliser un automate nous permettant de décrire (et trouver) comment obtenir précisément 4 litres dans le récipient de 5 litres. Il est uniquement possible de réaliser les actions suivantes :

— compléter le contenu de n'importe quel récipient jusqu'à sa contenance maximale (assurant ainsi que la quantité d'eau dans le récipient est égale à sa contenance) ;
— transvaser le contenu d'un récipient dans un autre.

Ces deux dernières opérations sont les seules permettant d'avoir une mesure précise de quantité d'eau dans les réservoirs.

1. Définir l'espace d'états de l'automate qui représente les différentes contenances des deux récipients.
2. Définir l'ensemble des états accepteurs de cet automate.
3. Définir l'alphabet de l'automate et la relation de transition entre états.
4. Supposons que cet automate ait été généré. Comment résoudre le problème initial ?
5. Comment résoudre le problème initial si on ne peut générer complètement l'automate apriori ?