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 2018 - 2019

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, Fréderic Prost, Anne Rasse : `prenom.nom@univ-grenoble-alpes.fr`
- Ali Kassem : `ali.kassem@inria.fr`.
- Mohamed Khatiri : `khatiri.med@gmail.com`.

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 serions heureux de vous aider.

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

Quelques remarques

- Le livret de travaux dirigés contient plus d'exercices qu'il n'est possible de faire pendant les séances de travaux dirigés du 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.
- Vous pouvez obtenir la solution de certains exercices en utilisant le logiciel Aude disponible à l'adresse suivante :

  `https://aude.imag.fr`

  Pour toute question ou rapport de bug concernant Aude, merci d'écrire à l'adresse

  `aude-contact@univ-grenoble-alpes.fr`

CHAPITRE

1 Notions préliminaires

Mots

Exercice 1 (♠♠) — Longeur et nombre d'occurences 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 2 (♠♠♠) — Mots définis par la droite ou par la gauche

Considérons un alphabet Σ. Soit $MOTS$ l'ensemble des mots que l'on peut définir avec la définition sous forme d'application. Soit $MOTS_D$ l'ensemble des mots que l'on peut définir avec la définition inductive des mots inductive à droite. De manière similaire, nous pouvons définir $MOTS_G$ l'ensemble des mots que l'on peut définir avec une définition inductive des mots à gauche : ϵ est un mot sur Σ, et si u est un mot sur Σ et a est un symbole dans Σ alors $a \cdot u$ est un mot sur Σ. Nous souhaitons montrer que ces définitions sont équivalentes, c'est-à-dire que les ensembles de mots ainsi définis sont les mêmes.

1. Démontrer que $MOTS_D = MOTS_G$.
2. Démontrer que $MOTS = MOTS_G$.

Concaténation de mots

Exercice 3 (♠) — 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 4 (♠♠) — Élément neutre de la concaténation

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

1. En considérant la définition 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 5 (♠♠♠) — Associativité de la concaténation

Nous considérons la la définition de mot sous forme d'application.

1. Montrer que l'opération de concaténation est associative.

Exercice 6 (♠♠♠) — Variantes de la concaténation

Nous considérons la définition inductive des mots. Nous souhaitons re-définir l'opération de concaténation $\cdot : \Sigma^* \times \Sigma^* \to \Sigma^*$ sur un alphabet Σ, de manière inductive. La concaténation de deux mots u et u' de Σ^* reste notée $u \cdot u'$.

1. Donner une définition inductive de l'opération de concaténation, par induction sur le premier opérande.
2. Donner une définition inductive de l'opération de concaténation, par induction sur le second opérande.
3. Montrer que le mot ϵ est l'élément neutre de l'opération de concaténation selon les deux définitions précédentes.
4. Montrer que l'opération de concaténation est associative, pour les deux définitions précédentes.

Langages

Exercice 7 (♠♠♠) — Puissance d'un alphabet

Rappelons que pour un alphabet Σ, Σ^k et $\Sigma^{\leq k}$ sont l'ensemble des mots sur Σ respectivement de longueur égale à k et de longueur inférieure ou 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 8 (♠♠♠) — Condition nécessaire et suffisante

Considérons Σ un alphabet et a un symbole de Σ. Considérons la proposition $\{a\} \cdot \Sigma^* = \Sigma^*$.

1. Montrer que cette proposition est toujours fausse.
2. Donner une condition *nécessaire* à la proposition.
3. Donner une condition *suffisante* à la proposition.
4. Donner, si cela est possible, une condition *nécessaire et suffisante* à la proposition.

Exercice 9 (♠♠♠) — Condition nécessaire et suffisante

Considérons Σ un alphabet et L un langage sur Σ. Considérons les propositions $L \cdot \Sigma^* = \Sigma^* \cdot L = L$ et $L \cdot \Sigma^* \neq \Sigma^*$. Pour chacune de ces propositions :

1. Donner une condition nécessaire à la proposition.
2. Donner une condition suffisante à la proposition.
3. Donner, si cela est possible, une condition nécessaire et suffisante à la proposition soit vraie.

Exercice 10 (♠♠♠) — Condition nécessaire et suffisante sur les langages

Considérons Σ un alphabet et a un symbole de Σ.

1. Donner une condition nécessaire et suffisante à la proposition suivante : $a \cdot \Sigma^* = \Sigma^*$.
2. Soit $L \subseteq \Sigma^*$ un langage défini sur Σ. Donner une condition nécessaire et suffisante à la proposition suivante : $L \cdot \Sigma^* = \Sigma^* L = L$.

Concaténation de langages

Exercice 11 (♠♠♠) — Concaténation de langages

La définition de concaténation de langage vue en cours est dédiée à un alphabet Σ donné. Nous nous intéressons à généraliser un peu cette définition. Considérons deux alphabets Σ_1 et Σ_2.

1. Définir un opérateur de concaténation de langage prenant en paramètre un mot sur Σ_1 et un mot sur Σ_2.

Exercice 12 (♠) — Concaténation avec le langage vide

Considérons un alphabet Σ et L un langage sur Σ. Nous nous intéressons à la concaténation de L avec le langage vide ($\emptyset$).

1. Démontrer que $L \cdot \emptyset = \emptyset \cdot L = \emptyset$.

Exercice 13 (♠♠♠) — 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

2 Automates à états finis déterministes (AEFDs)

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.

Automate et langage reconnu

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 AEFDs dans la Figure 2.1.
2. Donner la représentation tabulaire des AEFDs trouvés dans l'Exercice 17.

Exercice 15 (♠) — Langage reconnu par un automate

Considérons l'alphabet $\{a, b\}$ et les AEFDs dans la Figure 2.1.

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

Exercice 16 (♠♠♠) — Dénombrement d'automates

Considérons l'alphabet $\Sigma = \{a, b\}$.

1. Combien il y a-t-il d'AEFDs complets différents avec 2 états ? Avec 3 états ?
2. Mêmes questions avec les AEFDs en général.

Automate associé à un langage

Exercice 17 (♠) — 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.

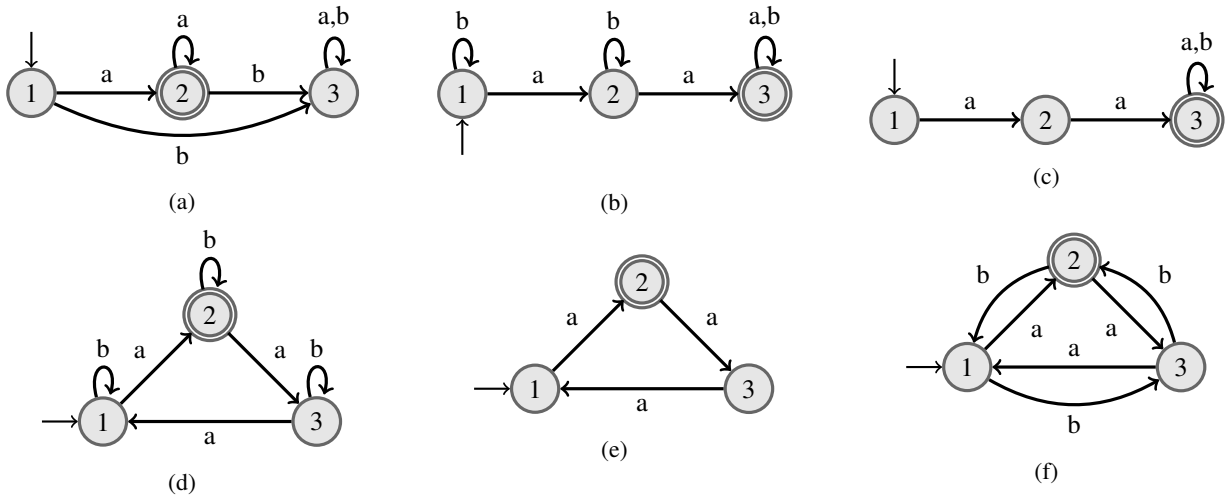


FIGURE 2.1 – Des automates à états finis déterministes

Exercice 18 (♠♠) — Automate pour un langage avec certains préfixes, suffixes ou facteurs

Considérons l'alphabet $\{a, b, c\}$. 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 ne terminent par $a \cdot b \cdot c$.
6. L'ensemble des mots qui ne se terminent pas par $a \cdot b \cdot c$.
7. L'ensemble des mots de longueur supérieure ou égale à 2 et tels que l'avant-dernier symbole est b.

Exercice 19 (♠♠) — Automate pour des langages avec des contraintes sur l'ordre des symboles

Considérons l'alphabet $\{a, b\}$. Pour les langages suivants, donner un automate reconnaisseur, si un tel automate existe.

1. L'ensemble des mots tels que a est toujours suivi de b.
2. L'ensemble des mots tels que a précède toujours b.
3. L'ensemble des mots tels que les occurrences du symbole a sont avant les occurrences du symbole b et les occurrences du symbole b sont avant les occurrences du symbole c.

Exercice 20 (♠♠) — Automates reconnaissant des entiers

Considérons l'alphabet $\Sigma = \{1, 2, \ldots, 9, 0\}$ et les entiers naturels notés en notation décimale usuelle.

1. Donner un AEFD qui reconnaît les entiers inférieurs à 245.

Exercice 21 (♠♠) — Automates reconnaissant des entiers multiples d'entiers

Considérons l'alphabet $\Sigma = \{1, 2, \ldots, 9, 0\}$ et les entiers naturels notés en notation décimale usuelle.

1. Donner des AEFDs qui reconnaissent les entiers multiples de 2, 3, 9, 10, 25, 50, 100, 250, 1000.
2. Donner des AEFDs qui reconnaissent les entiers multiples de 6 en bases 10, 2 et 4.
3. Donner des AEFDs qui reconnaissent les entiers multiples de 6 en base quelconque.
4. Donner un AEFD qui reconnait les entiers multiples de x en base y avec $x, y \in \mathbb{N}$.

Exercice 22 (♠♠) — Automate reconnaissant des horaires ou des dates

Considérons l'alphabet $\Sigma = \{1, 2, \ldots, 9, 0\}$.

1. Donner un AEFD qui reconnaît un horaire donné sous la forme $\mathrm{H_1H_2h} : \mathrm{M_1M_2m}$, avec $\mathrm{H_1, H_2, M_1, M_2} \in \Sigma$.
2. Donner un AEFD qui reconnait une date dans l'année donnée sous la forme $\mathrm{J_1J_2/M_1M_2}$ avec $\mathrm{J_1, J_2, M_1, M_2} \in \Sigma$.

Exercice 23 (♠♠) — Langage à états ou non

Pour chacun des langages suivants, dire si c'est un langage à états et justifier de manière informelle votre réponse.

1. $\{a^n \cdot b^n \mid n \in \mathbb{N}\}$.
2. $\{a^p \cdot b^q \mid (p, q) \in \mathbb{N}^2\}$.
3. $\{a^p \cdot b^q \mid p \geq q\}$.
4. $\{a^p \cdot b^q \mid p \geq q, q \geq 5\}$.
5. $\{a^p \cdot b^q \mid p \neq q\}$.
6. $\{a^p \cdot b^q \mid p = q\}$.
7. $\{a^p \cdot b^q \mid p = q \wedge p = 5\}$.
8. $\{a^p \cdot b^q \mid p < q\}$.
9. $\{a^p \cdot b^q \mid p \equiv q \mod 7\}$.
10. $\{a^n \cdot b \cdot a^n \mid n \in \mathbb{N}\}$.
11. $\{a^p \cdot b^q \mid p \geq q, q \leq 2017\}$.
12. $\{a^p \cdot b^q \mid p \geq q, q \geq 2017\}$.
13. $\{a^n \mid n \in$ n est un nombre premier$\}$.
14. $\{a^{2^n} \mid n \in \mathbb{N}\}$.
15. $\{b \cdot a \cdot b \cdot a^2 \cdot b \cdot a^3 \cdots b \cdot a^n \mid n \in \mathbb{N}\}$.
16. $\{w \cdot w \mid \in \Sigma^*\}$.
17. $\{w \cdot w \cdot w \mid w \in \Sigma^*\}$.
18. $\{u \in \Sigma^* \mid$ u est un palindrome$\}$.

Exercice 24 (♠♠♠) — Entiers en base 2 et divisibilité par 3

Nous considérons l'alphabet $\Sigma = \{0, 1\}$ des entiers représentés en binaire (c'est-à-dire en base 2). Les entiers dans cet exercice sont représentés en binaire.

1. Donner un automate qui accepte les entiers divisibles par 3 et représentés avec la convention big-endian, c'est-à-dire avec la lecture des bits de poids les plus forts en premier.
2. Donner un automate qui accepte les entiers divisibles par 3 et représentés avec la convention little-endian, c'est-à-dire avec la lecture des bits de poids les plus faibles en premier.

Fonction de transition d'un automate

Dans cette section, nous utilisons une définition **inductive** des mots (qui est équivalente à la définition de mots sous forme d'application vue en cours) :

— L'application $\epsilon : \emptyset \to \Sigma$ est un mot sur Σ. Elle est appelée le mot vide.
— Si u est un mot sur Σ et a est un symbole dans Σ alors $u \cdot a$ est un mot sur Σ

Exercice 25 (♠♠) — Fonction de transition et état puits

Considérons l'AEFD $(Q, q_0, \Sigma, \delta, F)$ et $q \in Q$ un état particulier de cet automate tel que $\forall s \in \Sigma : \delta(q, s) = q$.

1. Prouver par induction sur la longueur du mot d'entrée w que :

$$\forall w \in \Sigma^* : \delta^*(q, w) = q.$$

Exercice 26 (♠♠♠) — 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 27 (♠♠♠) — Fonction de transition étendue

À partir de la fonction de transition δ (opérant sur un état et un symbole), nous avons défini la fonction de transition étendue δ^* (son extension aux mots) comme la fermeture réflexive et transitive de δ.

1. Rappeler/proposer une définition inductive de cette fonction. Donner un argument justifiant le bon fondement de votre définition.
2. Prouver que : $\forall x, y \in \Sigma^*, \forall q \in Q : \delta^*(q, xy) = \delta^*(\delta^*(q, x), y)$.

Exercice 28 (♠♠♠) — Fonction de transition et état jumeaux

Considérons un AEFD $A = (Q, q_0, \Sigma, \delta, \{q_f\})$, et, supposons que pour chaque symbole $s \in \Sigma$ nous avons $\delta(q_0, s) = \delta(q_f, s)$. [1]

1. Prouver que, pour n'importe quel mot $w \neq \epsilon$, nous avons $\delta^*(q_0, w) = \delta^*(q_f, w)$.
2. Prouver que, si un mot non vide w est reconnu par$A$, alors w^k (le mot formé par k concaténations de w) est aussi dans $L(A)$ pour chaque entier strictement positif k :

$$\forall w \in \Sigma^* \setminus \{\epsilon\} : \big(w \in L(A) \Rightarrow \forall k \in \mathbb{N} \setminus \{0\} : w^k \in L(A)\big).$$

Automates et langages particuliers

Exercice 29 (♠♠♠) — Automate émondé

1. Montrer que à tout automate à états fini, on peut associer un automate émondé qui reconnaît le même langage.

Exercice 30 (♠♠♠) — Sous-automate et langage reconnu

Soit A un automate à états fini. On appelle sous-automate de A tout automate obtenu en enlevant un ou plusieurs états de l'ensemble des états de A ou en supprimant une ou plusieurs transitions de la fonction de transition de A.

1. Montrer que le langage reconnu par tout sous-automate de A est inclus dans $L(A)$.

Exercice 31 (♠♠♠) — Langages des préfixes, suffixes et facteurs

Soit L un langage sur un alphabet Σ. Pour rappel :

— l'ensemble des préfixes (des mots) de L est l'ensemble $\mathrm{Pref}(L) = \{u \in \Sigma^* \mid \exists u' \in \Sigma^* : u \cdot u' \in L\}$;
— l'ensemble des suffixes (des mots) de L est l'ensemble $\mathrm{Suf}(L) = \{u \in \Sigma^* \mid \exists u' \in \Sigma^* : u' \cdot u \in L\}$;
— l'ensemble des facteurs (des mots) de L est l'ensemble $\mathrm{Fact}(L) = \{u \in \Sigma^* \mid \exists u', u'' \in \Sigma^* : u'' \cdot u \cdot u' \in L\}$.

Supposons que L soit un langage à états.

1. Montrer que $\mathrm{Pref}(L)$ est également un langage à états.
2. Montrer que $\mathrm{Suf}(L)$ est également un langage à états.
3. Montrer que $\mathrm{Fact}(L)$ est également un langage à états.

1. Nous verrons dans un chapitre suivant que nous pouvons généraliser cette notion

CHAPITRE

3 Opérations de composition d'AEFDs

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

Automate complété

Exercice 32 (♠) — Complétude d'un automate

Considérons l'alphabet $\{a, b\}$ et les AEFDs dans la Figure 2.1.

1. Déterminer quels automates sont complets.
2. Compléter les automates qui ne sont pas complets.
3. Que se passe-t-il si nous complétons un automate déjà complet ?
4. Compléter les automates en considérant l'alphabet $\{a, b, c\}$?

Exercice 33 (♠♠♠) — Correction de l'opération de complétion

Nous souhaitons montrer que l'opération/l'algorithme de complétion est correcte, c'est-à-direqu'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 $L(C(A)) = L(A)$.

Automate complémentaire

Exercice 34 (♠♠) — Trouver l'automate complémentaire

Considérons les alphabets $\Sigma_1 = \{a, b, c\}$ et $\Sigma_2 = \{0, 1\}$.

1. Pour chacun des automates trouvés dans l'Exercice 17, donner un automate qui reconnaît le complémentaire dans Σ_1^* du langage reconnu par l'automate.
2. Pour chacun des langages de l'Exercice 18, donner un AEFD qui reconnaît son complémentaire dans Σ_1^* (si cela est possible).
3. Pour chacun des langages de l'Exercice 19, donner un AEFD qui reconnaît son complémentaire dans Σ_2^* (si cela est possible).

Exercice 35 (♠♠) — Trouver l'automate complémentaire

Considérons l'alphabet $\Sigma = \{a, b\}$. Donner un automate qui reconnait le complémentaire des langages reconnus par les automates suivants.

1. L'automate représenté dans la Figure 3.1a.
2. L'automate représenté dans la Figure 3.1b.
3. L'automate représenté dans la Figure 3.1c.

Exercice 36 (♠♠♠) — Correction de l'opération de complémentation

Nous souhaitons montrer que l'opération/l'algorithme de complémentation est correcte, c'est-à-direqu'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 $L(A^C) = \Sigma^* L(A)$.

Automate produit

Pour les automates suivants, utiliser la construction de l'automate produit.

Exercice 37 (♠) — Obtenir un automate par produit d'automates

Considérons l'alphabet $\Sigma = \{1, 2, \ldots, 9, 0\}$ et les entiers naturels notés en notation décimale usuelle. Vous pouvez ré-utiliser les automates trouvés dans l'Exercice 21.

1. Donner un automate qui reconnaît les nombres multiples de 6 en utilisant le produit entre l'automate reconnaissant les nombres multiples de 2 et l'automate reconnaissant les nombres multiples de 3.

Exercice 38 (♠♠) — Obtenir un automate par produit d'automates

Considérons l'alphabet $\Sigma = \{a, b\}$ et les automates trouvés dans l'Exercice 17.

1. Donner un AEFD qui reconnaît l'ensemble des mots qui contiennent un nombre de a multiple de 3 et multiple de 2.
2. Donner un AEFD qui reconnaît l'ensemble des mots qui contiennent un nombre de a multiple de 2 et non multiple de 3.
3. Pour chacun des langages précédents, donner un AEFD qui reconnaît le langage complémentaire du langage reconnu dans Σ^*.

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

Soit L un langage à états sur un alphabet Σ.

1. Montrer que l'ensemble des mots de L de longueur paire est un langage à états.
2. Montrer que l'ensemble des mots de L ne contenant pas un certain symbole $s \in \Sigma$ est un langage à états.

Exercice 40 (♠♠) — 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 18, 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 $a \cdot b$.

Exercice 41 (♠♠♠) — Correction du produit d'automates

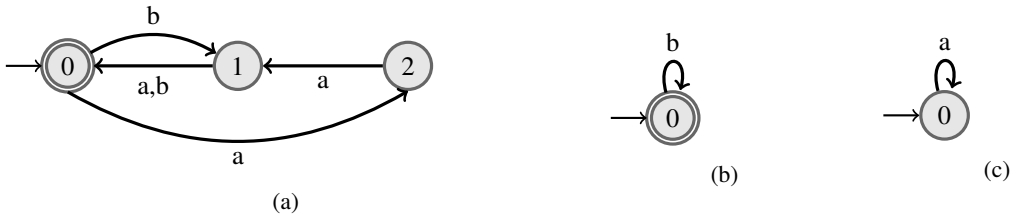


FIGURE 3.1 – Des automates à états finis déterministes

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 AEFDs. 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

4 Algorithmes et problèmes de décision sur les AEFDs

Exercice 42 (♠♠) — Algorithmes pour compléter

Considérons un automate sur un alphabet Σ.

1. À partir de la définition de l'automate complété vue en cours, donner un algorithme complétant un automate par rapport à un alphabet Σ' tel que $\Sigma \subseteq \Sigma'$.
2. Si nécessaire, adapter l'algorithme donné à la question précédente pour qu'il fonctionne avec un alphabet Σ' quelconque.

Exercice 43 (♠♠) — 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 44 (♠♠) — 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 45 (♠♠) — Algorithmes pour complémenter

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 état s 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 ensemble 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 46 (♠♠) — Déterminer si un automate est émondé

1. Donner un algorithme qui détermine si un automate est accessible.
2. Donner un algorithme qui détermine si un automate est co-accessible.
3. Donner un algorithme qui détermine si un automate est émondé. Cet algorithme pourra réutiliser les algorithmes trouvés aux questions précédentes.

Exercice 47 (♠♠♠) — 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 permettant de déterminer si $L \subseteq L'$.
3. Déduire de la question précédente, un algorithme permettant de déterminer si $L = L'$.

Exercice 48 (♠♠) — Montrer l'inclusion entre deux langages en utilisant leur automates

Considérons l'alphabet $\Sigma = \{a, b\}$.

1. Soit L_1 le langage des mots qui ne contiennent pas $abaa$ et qui contiennent un nombre pair de a. Donner un AEFD complet qui reconnaît L_1.
2. Soit L_2 le langage des mots qui ne contiennent pas aba et qui contiennent un nombre de a multiple de 4. Donner un AEFD qui reconnait L_2.
3. Montrer que $L_2 \subseteq L_1$ en utilisant les résultats de l'Exercice 47.

Exercice 49 (♠♠) — Émonder un automate

Soient Σ un alphabet. Donner des algorithmes qui résolvent les problèmes suivants.

1. Supprimer les états non accessibles et les transitions impliquant ces états (cad une transition telle que l'état de départ ou l'état d'arrivé soit un état supprimé).
2. Supprimer les états non co-accessibles et les transitions reliées à ces états.
3. Émonder un automate.

Exercice 50 (♠♠♠) — Reconnaitre les 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.

Exercice 51 (♠♠♠) — Plus grand sous-ensemble prefixe-clos d'un langage accepté

Nous considérons l'algorithme trouvé dans l'Exercice 50.

1. En s'inspirant de cet algorithme, donner un algorithme qui transforme un automate de telle manière à ce que le langage reconnu par l'automate produit soit le plus grand sous-ensemble préfixe-clos du langage reconnu par l'automate initial.

CHAPITRE

5 Équivalence, distinguabilité et minimisation

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

Équivalence et distinguabilité

Exercice 52 (♠♠) — Déterminer l'équivalence ou la non-équivalence entre automates

Considérons les deux automates dans la Figure 5.1a et Figure 5.1b. 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.

Minimisation

Exercice 53 (♠♠) — Minimiser des automates

Nous considérons les automates dans la Figure 5.2 sur l'alphabet $\Sigma = \{a, b\}$.

1. Minimiser l'AEFD dans la Figure 5.2a.
2. Minimiser l'AEFD dans la Figure 5.2b.

Exercice 54 (♠♠♠) — Trouver l'automate minimal reconnaissant un langage

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

1. Donner un AEFD minimal qui reconnaît le langage des mots qui ne contiennent pas aba.

Exercice 55 (♠♠) — Déterminer si un automate est minimal ou non

Nous considérons les deux automates dans la Figure 5.3. Nous souhaitons déterminer si ces automates sont minimaux. Pour chacun des automates (Figure 5.3a et Figure 5.3b), faire les questions suivantes.

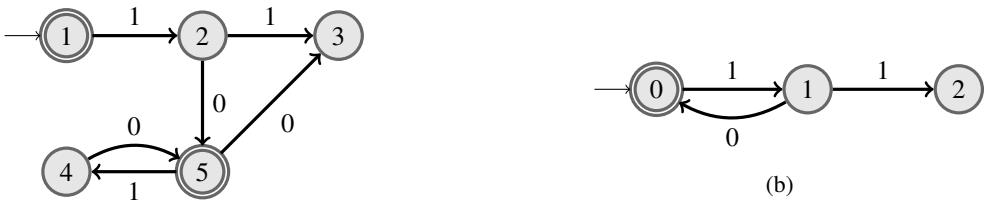


FIGURE 5.1 – Des AEFDs dont on veut déterminer l'équivalence.

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.

		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

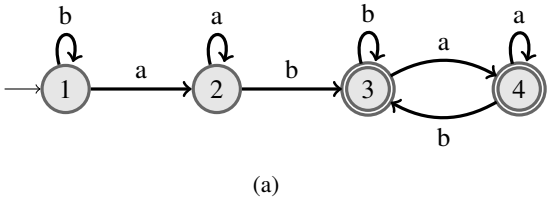
(c)

TABLE 5.1 – Des AEFDs 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.

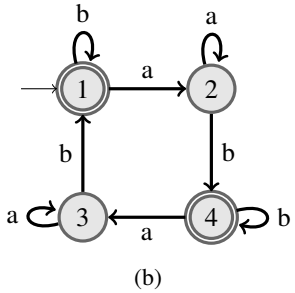
Exercice 56 (♠♠) — Déterminer si un automate est minimal ou non

Nous considérons les automates dans la Table 5.1. Nous souhaitons déterminer si ces automates sont minimaux. Pour chacun des automates (Table 5.1a et Table 5.1b) :

1. Utiliser l'algorithme calculant les états distinguables pour déterminer la minimalité.
2. Utiliser l'algorithme calculant les états équivalents pour déterminer la minimalité.
3. Vérifier que vous obtenez des résultats compatibles et donc les mêmes conclusions avec les deux méthodes.



(a)



(b)

FIGURE 5.2 – Des AEFDs en représentation graphique à minimiser.

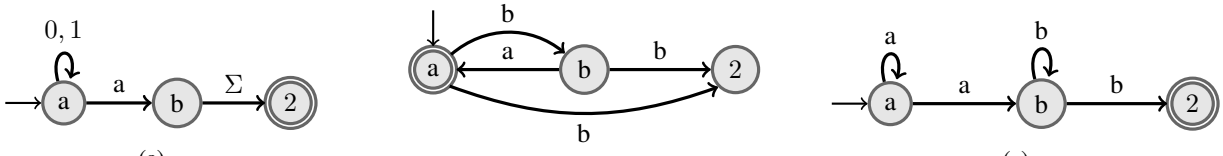
FIGURE 5.3 – Des AEFDs en représentation graphique à minimiser.

CHAPITRE

6 Automates à états finis non-déterministes

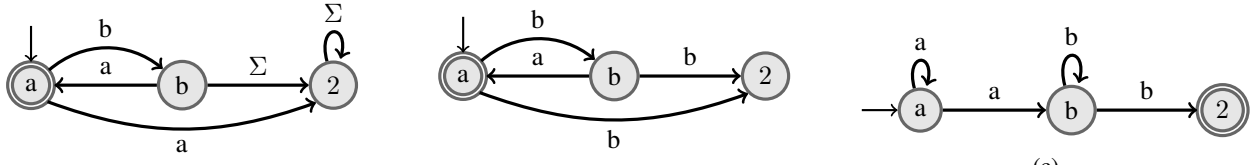
Rappel :
- AEFD : automate à états fini déterministe ;
- AEFND : automate à états fini non-déterministe.

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.



(a) (b) (c)

FIGURE 6.1 – Des automates à états finis non déterministes



(a) (b) (c)

FIGURE 6.2 – Des automates à états finis non déterministes

Langage et AEFND

Exercice 57 (♠) — Décrire le langage reconnu par un AEFND

Décrire en langage naturel chacun des automates représentés dans la Figure 6.1.

Exercice 58 (♠) — Donner un AEFND qui reconnaît un langage

Nous considérons l'alphabet $\{a, b\}$.

1. Donner un AEFND qui reconnait tous les mots de longueur supérieure ou égale à 2 et tels que l'avant-dernier symbole est b.

Exercice 59 (♠) — 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.)

	$\to 0$	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)

	$\to 0$*	1	2	3*	4*	5
a	1,2			5		
b		3	4			0

(b)

TABLE 6.1 – Des automates à états finis non déterministes représentés sous forme tabulaire

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 60 (♠♠♠) — Donner un AEFND qui reconnaît un langage

Soit A un AEFND reconnaissant un langage L. Pour chacun des langages suivants, donner un AEFND qui le reconnaît :

1. $\text{longueur}(L) = \{u \in \Sigma^* \mid \exists v \in L : |u| = |v|\}$.
2. $L^{1/2} = \{u \in \Sigma^* \mid \exists v \in \Sigma^* : |u| = |v| \wedge u \cdot v \in L\}$.

Exercice 61 (♠♠♠♠) — Donner un AEFND qui reconnaît les solutions d'équations

Pour $n \in \mathbb{N}$, soit $\widehat{n}$ l'ensemble des représentation en binaire de n où le bit de poids le plus faible est à gauche.
Par exemple :

— Pour $6 \in \mathbb{N}$: $011 \in \widehat{6}$ et $0110 \in \widehat{6}$.
— Pour $2 \in \mathbb{N}$: $010 \in \widehat{6}$ et $01 \in \widehat{2}$.

1. Donner un AEFND qui reconnaît les solutions de $y = 2x$ dans $\mathbb{N}$. Plus précisément, soit $\Sigma = \{0,1\} \times \{0,1\}$, nous cherchons un AEFND qui reconnaît $L \subseteq \Sigma^*$ tel que $u \in L$ si et seulement s'ils existent $x, y \in \mathbb{N}$ tels que :
   — $u = (x_1, y_1) \cdots (x_k, y_k)$,
   — $x_1 \cdots x_k \in \widehat{x}$,
   — $y_1 \cdots y_k \in \widehat{y}$, et
   — $y = 2 * x$.

Exercice 62 (♠♠♠) — Langages à états

La notion d'AEFND peut être généralisée en considérant que l'automate a un ensemble non vide d'états initiaux. Nous souhaitons montrer que ce type d'automate est équivalent aux AEFNDs classiques.

1. Adapter les notions de configuration, d'exécution et de langage accepté par ce type d'automates.
2. Indiquer comment montrer que ces automates sont équivalents aux AEFNDs.

Déterminiser un AEFND

Exercice 63 (♠♠) — Déterminiser

Soit A 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.

Exercice 64 (♠♠) — 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 6.1a.
— B l'AEFND donné par $(\{0,1,2,3,4,5\},\{a,b\},0,\Delta,\{0,3,4\})$ dont la relation de transition est définie par la Table 6.1b.

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

Exercice 65 (♠♠) — Déterminiser

Considérons l'AEFND représenté dans la Figure 6.2c.

1. Construire un AEFD équivalent par déterminisation.

Déterminer l'équivalence entre deux AEFNDs

Exercice 66 (♠) — Procédures pour déterminer l'équivalence

Nous considérons les procédures/algorithmes utilisés dans les chapitres précédents pour déterminer l'équivalence entre deux AEFDs.

1. Adapter les procédures/algorithmes aux AEFNDs.

Exercice 67 (♠♠) — Déterminer l'équivalence

Soit $\Sigma = \{0,1\}$. Considérons les deux premiers AEFNDs représentés dans la Figure 6.2.

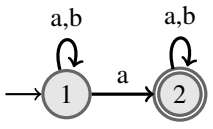
1. Quel est le langage reconnu par l'automate représenté dans la Figure 6.2a ?
2. Quel est le langage reconnu par l'automate représenté dans la Figure 6.2b ?
3. Montrer que ces deux automates sont équivalents.

Complexité

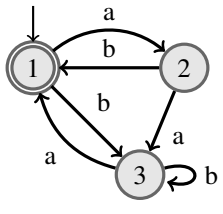
Exercice 68 (♠♠♠) — Nombre d'exécutions acceptées

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.

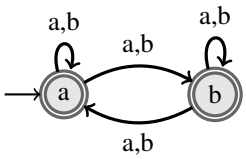
1. Considérons l'automate A_1 représenté dans la Figure 6.3a, lister les exécutions acceptées et associées au mot $abaa$. En déduire $\mathcal{N}(A_1, abaa)$.
2. Montrer que $\mathcal{N}(A_1,u) = |u|_a$.
3. Considérons l'automate A_2 représenté dans la Figure 6.3b, quelle est la valeur de $\mathcal{N}(A_2,u)$ pour un mot u. Justifier.
4. Considérons l'automate A_3 représenté dans la Figure 6.3c, quelle est la valeur de $\mathcal{N}(A_3,u)$ pour un mot u. Justifier.



(a) AENFD A_1



(b) AENFD A_2



(c) AENFD A_3

FIGURE 6.3 – Des automates à états finis non déterministes

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 $\forall u \in \Sigma^* : \mathcal{N}(A, u) = (|u|_a)^2$.

Exercice 69 (♠♠♠) — Complexité de la déterminisation avec le nombre d'états, exemples

Nous considérons les langages L_i tels que définis dans l'Exercice 59.

1. Donner le nombre d'états des AEFD reconnaissant L_1, L_2 et L_3. Vous pouvez utiliser un outil tel que Aude pour trouver l'AEFD qui reconnaît L_3.
2. Que pouvons-nous extrapoler pour le nombre d'états d'un AEFD reconnaissant L_n pour $n \in \mathbb{N}$?

Exercice 70 (♠♠♠) — Complexité de la déterminisation avec le nombre d'états, généralisation

Nous considérons l'algorithme de déterminisation vu en cours opérant par construction des sous-ensembles.

1. Au pire des cas, pour un AEFND avec n états, quel est le nombre d'états de l'AEFD équivalent obtenu par cet algorithme ?
2. Déterminer une condition sur l'exécution de l'algorithme pour que le pire cas se produise.
3. Déterminer le nombre de fois ou chaque opération de l'algorithme s'exécute dans le pire cas.
4. En supposant un temps d'exécution moyen de 1ms pour chaque opération de l'algorithme, donner une estimation du temps d'exécution de l'algorithme de déterminisation lorsqu'il est exécuté sur des AEFND de différentes tailles.

Algorithmes pour les AEFNDs

Exercice 71 (♠♠) — Algorithmes pour décider le déterminisme

Considérons deux alphabets Σ et Σ' tels que $\Sigma' \subseteq \Sigma$ et un AEFND défini sur Σ. Pour cet exercice, nous définissons la notion de déterminisme par rapport à un alphabet. Nous disons qu'un AEFND défini sur l'alphabet Σ est déterministe par rapport à Σ' si sa relation de transition est une fonction lorsqu'elle est restreinte à Σ'.

1. Donner un algorithme pour déterminer si un AEFND défini sur Σ est déterministe par rapport à Σ'.
2. Déduire de la question précédente un algorithme pour déterminer si un AEFND est un AEFD.

CHAPITRE

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

Rappel : ϵ-AEFND : Automate à États Fini Non-Déterministe avec ϵ-transitions.

Fermeture de Kleene

Exercice 72 (♠♠♠) — Idempotence de la fermeture de Kleene

Nous souhaitons montrer que la fermeture de Kleene est idempotente.

1. Montrer que : $\forall L \subseteq \Sigma^* : L^* = (L^*)^*$.

Exercice 73 (♠♠♠) — Fermeture de Kleene, union et inclusion

Soient L_1 et L_2 des langages.

1. Montrer que si $L_1 \subseteq L_2$, alors $L_1^* \subseteq L_2$.
2. En déduire que $L_1^* \cup L_2^* \subseteq (L_1 \cup L_2)^*$.
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 $L_1 \not\subseteq L_2$, $L_2 \not\subseteq L_1$ et $L_1^* \cup L_2^* = (L_1 \cup L_2)^*$

Exercice 74 (♠♠♠) — Fermeture de Kleene, union et concaténation

1. Montrer que : $\forall L_1, L_2 \subseteq \Sigma^* : (L_1 \cup L_2)^* = (L_1^* \cdot L_2^*)^*$.
2. Est-ce que : $\forall L_1, L_2 \subseteq \Sigma^* : (L_1 \cup L_2)^* = (L_1 \cdot L_2)^*$? Si oui, faire une preuve, sinon, donner un contre-exemple.
3. Déterminer une condition nécessaire et suffisante pour que $(L_1 \cup L_2)^* = (L_1 \cdot L_2)^*$.

Élimination des ϵ-transitions et déterminisation

Exercice 75 (♠♠) — 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 7.1a. Pour les questions suivantes, utiliser la représentation tabulaire des automates.

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

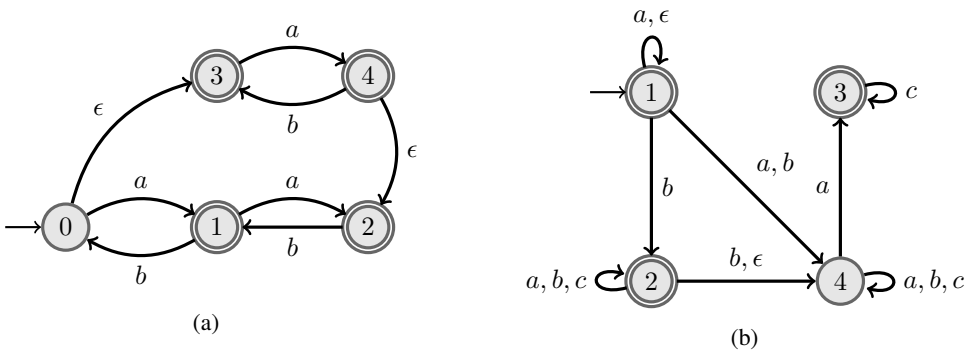


FIGURE 7.1 – Des automates à états-finis non déterministes avec ϵ-transitions

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-direen utilisant les deux méthodes) sont équivalents.

Exercice 76 (♠♠) — 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 7.1b. 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-direen utilisant les deux méthodes) sont équivalents.

Exercice 77 (♠♠) — Élimination des ϵ-transitions et déterminisation

Considérons l'alphabet $\Sigma = \{a,b\}$ et les ϵ-AEFND représentés dans la Figure 7.1a et Figure 7.1b, 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 78 (♠♠) — Élimination des ϵ-transitions et déterminisation

Soit $\Sigma = \{a,b,c\}$ et l'ϵ-AEFND dans la Figure 7.1b défini sur Σ :

1. Supprimer les ϵ-transitions.
2. Déterminiser l'automate obtenu à la question précédente.
3. Minimiser l'automate obtenu à la question précédente.

Composition et transformation d'ϵ-AEFNDs

Exercice 79 (♠♠♠) — Montrer qu'un langage est un langage à états

Soit L un langage à états sur un alphabet Σ.

1. Montrer que $\{w \mid s \cdot w \in L\}$, l'ensemble des mots de L commençant par un symbole $s \in \Sigma$ et le supprimant, est un langage à états.
2. Montrer que $\{w \mid w \cdot s \in L\}$, l'ensemble des mots non vide de L terminant par un symbole $s \in \Sigma$ et le supprimant, est un langage à états.
3. Montrer que $\{w \cdot s \cdot s \mid w \cdot s \in L\}$, l'ensemble des mots obtenus un doublant la dernière lettre des mots non vide de L, est un langage à états.
4. Montrer que $\{w_1 \cdot s \cdot w_2 \mid w_1, w_2 \in L\}$, 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 80 (♠♠♠) — 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 $L_1 \cap L_2$.
5. Démontrer que les compositions d'automates définies aux questions précédentes sont correctes, c'est-à-dire que $L(A_\cup) = L_1 \cup L_2$, $L(A_\cdot) = L_1 \cdot L_2$, $L_* = L_1^*$ et $L(A_\cap) = L_1 \cap L_2$.

Exercice 81 (♠♠♠) — 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 82 (♠♠♠) — Langage absolu

Un langage est dit absolu si le fait qu'un mot soit dans le langage dépend seulement des n dernières lettres de ce mot.

1. Prouver que tout langage absolu est un langage à états.
2. Prouver que les langages absolus sont fermés par union, intersection, et complémentation.
3. Prouver que les langages absolus ne sont pas fermés par concaténation et fermeture de Kleene.

Algorithmes sur les ϵ-AEFND

Exercice 83 (♠♠) — Algorithme pour calculer l'ϵ-fermeture d'un état

Soient Σ un alphabet.

1. Donner un algorithme qui calcule l'ϵ-fermeture d'un état d'un ϵ-AEFND passé en paramètre.

Exercice 84 (♠♠♠) — Algorithme pour déterminer le langage universel

Soient Σ et Σ' deux alphabets. Donner des algorithmes qui résolvent les problèmes suivants.

1. Déterminer si un ϵ-AEFND sur un alphabet Σ reconnaît le langage universel sur Σ.
2. Déterminer si un ϵ-AEFND sur un alphabet Σ reconnaît le langage universel sur Σ'.

CHAPITRE

8 Modélisation et résolution de problèmes avec les automates

Exercice 85 (♠♠♠) — 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 86 (♠♠♠) — 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 87 (♠♠♠) — 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 monaie) ou y introduire des failles ; pour le client vous pouvez considérer un client malicieux.

Exercice 88 (♠♠♠) — 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 89 (♠♠♠♠) — Étrange planète

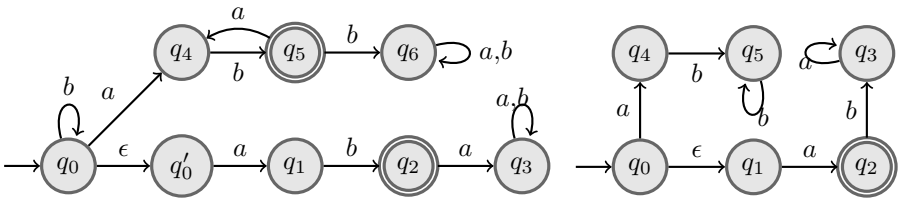
Nous reprenons la modélisation de l'étrange planète vue dans le cours d'introduction. Sur cette planète trois espèces cohabitent. Nous appellerons les espèces rouge, bleue et orange. Nous souhaitons modéliser l'évolution de la population de ces espèces selon leur mode de reproduction, qui suit les règles 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 90 (♠♠♠♠) — 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 91 (♠♠♠) — 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 ?

CHAPITRE

9 Expressions régulières

À propos des expressions régulières

Exercice 92 (♠) — 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 ocurrences 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 93 (♠♠) — Combien de mots d'une longueur donnée dans le langage

Considérons l'alphabet $\Sigma = \{a, b\}$. Pour chacune des expressions régulières suivantes, dire combien de mots de longueur 100 sont dans le langage qu'elles décrivent.

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

Exercice 94 (♠♠♠) — Preuves des identités classiques sur les expressions régulières

Les identités classiques entre expressions régulières vues en cours sont rappelées dans la Table 9.1.

1. Prouver chacune de ces identités.

Exercice 95 (♠♠♠) — Équivalence ou non entre expressions régulières

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

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

Expression régulière	Expression régulière équivalente	Remarque
e^*	$\epsilon + e \cdot e^*$	apériodicité
e^*	$\epsilon + e^* \cdot e$	apériodicité
$(\emptyset)^*$	ϵ	définition de l'opérateur de Kleene
$e + e$	e	idempotence
$(e^*)^*$	e^*	idempotence
e^*	$\epsilon + e \cdot e^*$	apériodicité
e^*	$\epsilon + e^* \cdot e$	apériodicité
$(\emptyset)^*$	ϵ	définition de l'opérateur de Kleene
$e + e$	e	idempotence
$(e^*)^*$	e^*	idempotence

TABLE 9.1 – Identités classiques entre expressions régulières

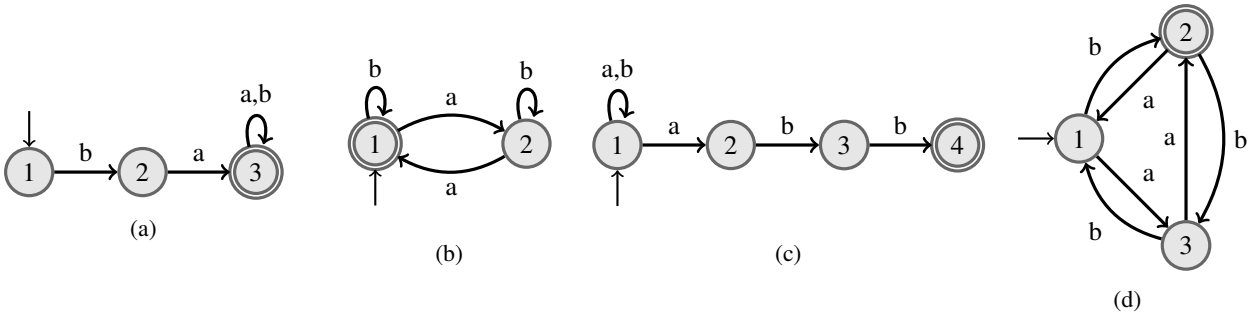


FIGURE 9.1 – Des automates pour le calcul d'expressions régulières

Exercice 96 (♠♠) — Taille d'une expression régulière et nombre d'états dans l'automate

Nous considérons l'alphabet $\Sigma = \{a, b\}$. Nous nous intéressons à la relation entre la taille d'une expression régulière, le nombre d'états de l'AEFND et de l'AEFD correspondants à cette expression.

1. Considérons l'expression régulière $E_n = \Sigma^* \cdot \Sigma^n$. Donner la forme générale de l'AEFND reconnaissant E_n.
2. Donner le nombre d'états des AEFDs reconnaissant E_1, E_2, E_3.
3. Extrapoler le nombre d'états de l'AEFD qui reconnait E_n.
4. Reprendre les questions précédentes avec l'expression régulière $E'_n = \Sigma^* \cdot b \cdot \Sigma^n$.
5. Comparer les tailles des AENFD et AEFD pour E_n et E'_n.

Exercice 97 (♠♠) — Simplifier des expressions régulières

Considérons l'alphabet $\Sigma = \{a, b\}$. Simplifier chacune des expressions régulières suivantes, c'est-à-diretrouver 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 98 (♠♠) — Expressions régulières étendues

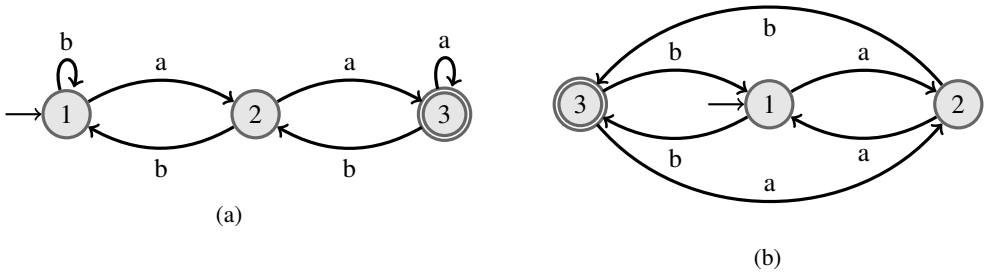


FIGURE 9.2 – Des automates pour le calcul d'expressions régulières

L'ensemble des expression régulières étendues est obtenu en ajoutant les constructions suivantes aux expressions régulières :

— Si e est une expression régulière sur Σ alors $\neg e$ est une expression régulière sur Σ.
— Si e et e' sont des des expressions régulières sur Σ, alors $e \cap e'$ est une expression régulière sur Σ.
— Si e est une expression régulière sur Σ alors $e^{\overset{+}{-}}$ est une expression régulière sur Σ.

La sémantique de ces opérateurs est défini comme suit :

— $L(\neg e) = \Sigma^* \setminus L(e)$.
— $L(e \cap e') = L(e) \cap L(e')$.
— $L(e^{\overset{+}{-}}) = L(e) \cdot L(e)^*$.

1. Donner les grandes lignes d'un algorithme qui transforme toute expression régulière étendue vers une expression régulière.

Exercice 99 (♠♠♠) — Fermeture par opération miroir

Sans utiliser les automates, montrer que l'ensemble des expressions régulières est fermé par l'opération miroir.

Calcul d'expressions régulières à partir d'automates

Exercice 100 (♠♠) — Automate vers expression régulière

Nous considérons les automates minimaux du Chapitre 5.

1. Pour chaque automate, donner une expression qui décrit le langage qu'il reconnait en utilisant la méthode associant des équation linéaires aux états.

Exercice 101 (♠♠) — Automate vers expression régulière

Nous souhaitons calculer les expressions régulières associées aux automates dans la Figure 9.2.

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

Exercice 102 (♠♠♠) — Lemme d'Arden

Soient $A, B, X \subseteq \Sigma^*$ des langages.

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

Calcul d'automates à partir d'expressions régulières

Exercice 103 (♠♠) — 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. $\big(a \cdot b + a^* \cdot b + c \cdot d\big) \cdot \big((c \cdot a^* + b \cdot d) \cdot (a \cdot b^* + a \cdot b \cdot d)\big)^*$.

CHAPITRE

10 Grammaires et grammaires régulières

Générer des mots avec les grammaires

Exercice 104 (♠) — Dérivation de grammaire

Considérons la grammaire $G = (\{a, b, c\}, \{S\}, S, P)$ où P est donné par les règles dans la Figure 10.1. Donner une dérivation de G pour les mots suivants :

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

Exercice 105 (♠) — 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.

Langages et Grammaires

Exercice 106 (♠) — Langage généré par une grammaire

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

1. $\left\{ \begin{array}{lcl} S & \to & T \mid bSb, \\ T & \to & aT \mid \epsilon \end{array} \right\}$;
2. $\left\{ \begin{array}{lcl} S & \to & T \mid bS, \\ T & \to & aT \mid \epsilon \end{array} \right\}$

Exercice 107 (♠♠) — Langage reconnu par une grammaire

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

FIGURE 10.1 – Règles de production de la grammaire pour l'Exercice 104.

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

Exercice 108 (♠♠) — 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 109 (♠♠) — Grammaire qui génère un langage

Pour chacun des langages suivants, donner une grammaire qui le génère et générer quelques mots. Indiquer si cette grammaire est régulière, linéaire à droite ou linéaire à gauche.

1. $\{a^n \cdot b^{2\times n} \mid n \in \mathbb{N}\}$;
2. $\{a^n \cdot b^n \mid n \in \mathbb{N}\}$;
3. $\{a^n \cdot b^m \mid n, m \in \mathbb{N}\}$;
4. $\{b \cdot a^n \cdot b \mid n \in \mathbb{N}\}$;
5. $\{a^n \cdot b \cdot c^n \mid n \in \mathbb{N}\}$;
6. $\{a^{n+1} \cdot b^n \mid n \in \mathbb{N}\}$;

Exercice 110 (♠♠) — Grammaire générant les expressions booléenes

Nous considérons des expressions booléennes simples formées par des variables, la composition par les opérateurs de disjonction, de conjonction, et de négation.

1. Donner une grammaire qui permet de générer des expressions booléennes telles que décrites ci-dessus.

Exercice 111 (♠♠) — Grammaire générant les commandes conditionnelles

Nous considérons un langage de programmation qui contient des commandes dans l'ensemble Σ_{stm}.

1. Donner une grammaire qui permet de générer des commandes conditionnelles de la forme `if ...then ...`.
2. Donner une grammaire qui permet de générer des commandes conditionnelles de la forme `if ...then ...else ...`.

Exercice 112 (♠♠) — Grammaire générant les expressions régulières

Nous considérons les expressions régulières, comme définies en cours.

1. Donner une grammaire qui permette de générer les expressions régulières.

Exercice 113 (♠♠) — Grammaire générant les chaînes de bits

Considérons l'alphabet $\Sigma = \{0, 1\}$ contenant les symboles utilisés dans la représentation binaire des entiers naturels. Nous considérons le langage formé par les entiers naturels pairs écrits en binaire et dans la notation big-endian : les mots dans ce langage terminent par un 0 et ne commencent pas par un 0 sauf pour le mot représentant l'entier naturel 0.

1. Donner une définition formelle de ce langage.
2. Écrire une grammaire linéaire à droite qui génère ce langage.
3. Écrire une grammaire linéaire à gauche qui génère ce langage.

Exercice 114 (♠♠) — Lien entre non-régularité d'un langage et d'une grammaire

Nous considérons le langage $\{a^n \cdot b^n \mid n \in \mathbb{N}\}$. Ce langage n'est pas régulier.

1. Essayer d'écrire une grammaire régulière qui génère ce langage et décrire le problème qui se pose.

Des grammaires vers les automates

Exercice 115 (♠♠) — Grammaire vers automate

Donner des automates qui reconnaissent les langages décrits par les grammaires données dans l'Exercice 106, 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 :

1. $\left\{ \begin{array}{lcl} S & \to & T \mid bSb, \\ T & \to & aT \mid \epsilon \end{array} \right\}$;

2. $\left\{ \begin{array}{l} S \to T \mid bS, \\ T \to aT \mid \epsilon \end{array} \right\}$.

Exercice 116 (♠♠) — Grammaire vers automate

1. Si cela est possible, donner des automates qui reconnaissent les langages décrits par les grammaires données dans l'Exercice 107. Si cela n'est pas possible, justifier.

Des automates vers les grammaires

Exercice 117 (♠♠) — Des AEFDs vers les grammaires

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

1. Les AEFDs dans la Figure 2.1.
2. Les AEFDs dans la Figure 9.1.
3. Les AEFDs non-minimaux dans la Figure 5.2.

Exercice 118 (♠♠) — Des AFENDs vers les grammaires

Donner les grammaires générants générants les langages reconnaissant par les AEFNDs donnés dans les figures suivantes.

1. Les AEFNDs dans la Figure 6.1.
2. Les AEFNDs dans la Figure 6.2.

Exercice 119 (♠♠) — Des automates vers les grammaires

Considérons les AEFNDs dans la Figure 6.1 et Figure 6.2 et les AEFDs correspondants obtenus dans le Chapitre **??**.

1. Donner la grammaire correspondant aux AEFDs.
2. Comparer la grammaire obtenu à la question précédente aux grammaires obtenues dans l'Exercice 118.

Des expressions régulières vers les automates et les grammaires

Exercice 120 (♠♠) — Des AEFNDs vers les grammaires

Pour chacun des langages suivants décrits par les expressions régulières, donner un AENFD qui reconnait le langage et utiliser l'AEFND pour obtenir une grammaire générant ce langage.

1. $a \cdot b$,
2. a^*,
3. $a^* \cdot b$,
4. $(a+b)^*$,
5. $(a+b)^*$,
6. $(a+b)^* \cdot a^* \cdot b^* \cdot a$,
7. $a + (a+b)^* ab^*$
8. $\emptyset \cdot b^* + (a \cdot \emptyset)^*$.

CHAPITRE

11 Langages non réguliers et lemme de l'itération

Lemme de l'itération

Dans cette section, 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 121 (♠) — 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 122 (♠♠) — 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. $\{a^{2\times i}(b \cdot c)^i \mid i \in \mathbb{N}\}$.
3. $\{w \cdot w \cdot w \mid w \in \Sigma_2^*\}$.

Exercice 123 (♠♠♠) — Utilisation du lemme de l'itération

Prouver que les langages suivants ne sont pas réguliers.

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

Exercice 124 (♠♠♠♠) — Utilisation du lemme de l'itération

Prouver que les langages suivants ne sont pas réguliers.

1. $\{a^i b^j \mid i$ et j sont premiers entre eux$\}$;

Exercice 125 (♠♠) — 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. $\{a^n b^n \mid n \in \mathbb{N}\}$
3. $\{w \cdot w^R \mid w \in \Sigma^*\}$
4. $\{w \cdot u \cdot w^R \mid w \in \Sigma^*, u \in \Sigma^+\}$.

Exercice 126 (♠♠♠) — 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.

Constante d'itération

Exercice 127 (♠♠) — Trouver la constante d'itération minimale

Considérons l'alphabet $\Sigma = \{0, 1\}$. Donner la constante d'itération minimale des langages réguliers sur Σ suivants :

1. $(01)^*$
2. $10(11^*0)^*0$
3. 1011
4. $011 + 0^*1^*$
5. $0^*1^+0^+1^*$
6. 10^*1
7. $0^*1^+0^+1^* + 10^*1$
8. 0^*101^*
9. $0^*101^* + 101^*$
10. ϵ
11. 0001^*
12. 0^*1^*

Fermeture des langages réguliers

Exercice 128 (♠♠♠) — 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 129 (♠♠♠) — Utilisation des propriétés de fermeture pour montrer la non-régularité

Considérons l'alphabet $\Sigma = \{a, b\}$. En utilisant les résultats de l'Exercice 125, prouver que les langages suivants sont non réguliers.

1. $\{a^m b^k a^n \mid m, k, n \in \mathbb{N}, m \neq n\}$
2. $\{w \in \Sigma^* \mid w \neq w^R\}$

Exercice 130 (♠♠) — Utilisation de la condition suffisante de différentiabilité pour montrer la non-régularité

Considérons un alphabet Σ. Dans cet exercice, il faut utiliser la condition suffisante de différentiabilité d'une infinité de mots pour montrer la non-régularité.

1. Prouver que $\{u \cdot a^{|u|} \mid u \in \Sigma^*\}$ n'est pas un langage régulier.

Exercice 131 (♠♠) — 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.

1. Prouver que $\{u \in \Sigma^* \mid |u| \text{ est premier}\}$ n'est pas un langage régulier.
2. Prouver que $\{u \in \Sigma^* \mid |u| \text{ est un carre}\}$ n'est pas un langage régulier.
3. Prouver que $\{0^i 1^j 2^{i+j} \mid i, j \in \mathbb{N}\}$ n'est pas un langage régulier.
4. Prouver que $\{0^{2\times i+1} 1^{2\times i+1} \mid i \in \mathbb{N}\}$ n'est pas un langage régulier.

Exercice 132 (♠♠♠) — 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.

ANNEXE

A Notions mathématiques de base

Ce chapitre est destiné à vous exercer sur les notions de base en mathématiques. Ce chapitre ne sera pas corrigé durant les TDs à moins que vous prépariez les exercices et que vous le demandiez à votre enseignant de TD.

Ensembles, relations

Exercice 133 ()

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 134 ()

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 135 ()

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 136 ()

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.

Preuves par induction

Exercice 137 ()

Prouver que pour chaque entier naturel $n > 1$:

1. n a un diviseur premier ;
2. si n n'est pas premier, il a un diviseur premier p tel que $p \leq \sqrt{n}$;

3. il existe un nombre premier strictement plus grand que n.

Exercice 138 ()

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 139 ()

1. Prouver la proposition suivante : $\forall n \in \mathbb{N}, \forall i \in \mathbb{N} : (n \geq 1 \wedge 1 \leq i \leq n) \implies n \leq i(n - i + 1)$
2. Déduire que $\forall n \in \mathbb{N} : n \geq 1 \implies \log n! \leq n \log n \leq 2(\log n!)$.

Définitions inductives

Exercice 140 ()

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 141 ()

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.