

Théorie des langages

Pierre Berlioux, Mnacho Echenim et Michel Lévy

Année 2017-2018

Table des matières

I	Introduction aux langages formels	11
1	Définitions de base	13
1	Prérequis sur les relations	13
1.1	Définitions	13
2	Mots et langages	16
2.1	Mots	16
2.2	Langages	19
2	Automates finis	21
1	Introduction	21
2	Automates finis et langages	23
3	Automates finis sans ϵ -transition	26
4	Automates finis déterministes	28
5	Construction des automates déterministes	30
6	Machines de Mealy et de Moore	33
3	Automates finis déterministes minimaux	37
1	Construction de l'automate minimal	37
2	Relation d'équivalence entre les états d'un automate	39
3	Unicité de l'automate minimal	42
3.1	Propriétés de la catégorie des automates	43
4	Pour aller plus loin : le théorème de Myhill-Nérode	45
4	Expressions régulières	49
1	Définition des expressions régulières	49
2	Tout langage régulier est reconnu par un automate fini	50
3	Tout langage reconnu par un automate fini est un langage régulier	53
4	Système d'équations associé à un automate fini	56
5	Propriétés de fermeture des langages réguliers	59
1	Le lemme de l'étoile	59
2	Opérations préservant la régularité d'un langage	62
2.1	Opérations ensemblistes	62

2.2	Substitutions régulières	63
2.3	Homomorphismes et homomorphismes inverses	65
6	Systèmes de réécriture, grammaires	67
1	Systèmes de réécriture	67
1.1	Définitions	67
2	Grammaires	70
2.1	Définitions	70
2.2	Une définition équivalente des grammaires sous-contexte	71
2.3	Quelques propriétés	71
2.4	Exemples	74
2.5	Exercices	75
7	Grammaires hors-contexte	79
1	Propriété fondamentale (décomposition des dérivations hors-contexte)	79
2	Dérivations indicées dans une grammaire hors-contexte	80
2.1	Dérivations indicées gauches	81
2.2	Dérivations gauches	83
3	Décomposition des dérivations indicées gauches	84
4	Arbre de dérivation	85
5	Ambiguïté	88
II	Notions avancées	91
8	Définition de langages par induction	93
1	Exemples de définitions inductives de langages	93
2	Théorèmes du point fixe	94
2.1	Ordre faiblement complet	94
2.2	Ordre complet	95
2.3	Treillis finis	97
3	Langages algébriques	98
4	Équivalence entre langages algébriques et hors-contextes	102
5	Transformation de systèmes d'équations	103
5.1	Substitution simple	104
5.2	Substitution générale	105
9	Transformations des grammaires hors-contexte	109
1	Rappels du chapitre précédent	109
1.1	Grammaires et équations	109
1.2	Des substitutions, qui préservent les langages	110
1.3	Treillis fini et calcul de points fixes	111
2	Réduction d'une grammaire hors-contexte	111

2.1	Calcul de l'ensemble des symboles productifs	111
2.2	Calcul de l'ensemble de l'ensemble des symboles accessibles	112
2.3	Construction d'une grammaire réduite	113
3	Élimination des ε -règles	114
4	Élimination des 1-règles	117
5	Forme normale de Chomsky	118
6	Grammaires hors-contexte récursives	120
7	Élimination des règles directement récursive à gauche	121
8	Élimination de la récursivité à gauche	122
9	Forme normale de Greibach	126
10	Conditions d'unicité des systèmes d'équation	129
1	Système strict	129
2	Conditions nécessaires et suffisantes d'unicité	130
3	Problèmes de décision	134
3.1	Le problème «un système est-il riche ?» est indécidable	134
	Système riche : rappel	134
3.2	Indécidabilité du problème «un système est-il riche ?»	134
	Le problème «un système a-t-il une seule solution ?» est relativement décidable	136
III	Annexes	137
A	Rappels sur l'induction	139
1	Compléments sur les relations d'ordre	139
2	Induction bien fondée	141
3	Induction structurelle	142
B	Éléments de corrections	145
1	Langages, relations	145
2	Systèmes de réécriture, grammaires	149
3	Grammaires hors-contexte	151

Introduction

Le terme "langage" désigne d'une part les langues naturelles (sous leur forme parlée et écrite), d'autre part les systèmes de notations (les formalismes) utilisés dans diverses sciences comme les mathématiques, la logique, la chimie... et en particulier l'informatique (les langages de programmation par exemple). Tout langage comprend un ensemble d'objets élémentaires, que l'on peut composer pour constituer des unités qui ont un sens. La composition des objets élémentaires est le plus souvent un simple enchaînement linéaire (une concaténation) ; les unités qui ont un sens sont alors des suites finies d'objets élémentaires. Cette opération peut se répéter à plusieurs niveaux. Ainsi dans une langue naturelle écrite, un mot est une suite de lettres (niveau lexical) ; une phrase est une suite de mots (niveau syntaxique) ; un texte est une suite de phrases (niveau du discours).

La définition d'un langage à un niveau donné est la définition :

1. de l'ensemble des objets élémentaires (le *vocabulaire* ou l'*alphabet* du langage du niveau considéré),
2. de l'ensemble des suites d'objets élémentaires ayant un sens (la *syntaxe*),
3. des sens de ces suites (la *sémantique*). La définition d'un langage utilise elle-même un langage (souvent appelé *métalangage*), qui peut être une langue naturelle ou un formalisme.

Le langage défini est parfois utilisé comme métalangage.

Exemple 1 : le français écrit.

Niveau lexical (ou lexicographique) : les objets élémentaires sont les lettres de l'alphabet, les unités ayant un sens des mots. Un dictionnaire comme le Robert donne une définition lexicographique du français : énumération des mots, définition (en français) des sens de chaque mot. Un dictionnaire français-anglais donne aussi une définition lexicographique du français, les sens des mots français étant définis par leurs traductions en anglais. Au niveau *morphologique*, les mots sont considérés comme des suites d'objets élémentaires qui sont eux-mêmes des suites de lettres appartenant à des catégories grammaticales comme préfixe, suffixe, racine, désinence, etc.

Niveau syntaxique : les objets élémentaires sont les mots et les signes de ponctuation, les unités ayant un sens les phrases. Une grammaire comme le "bon usage" de Grevisse donne une définition du français au niveau syntaxique : elle définit les phrases correctes et leur sens au moyen de règles exprimées en français, avec des termes techniques comme "adjectif", "substantif", "mas-

culin", "subordonnée", "relative", etc. La définition des phrases correctes sans référence à leurs sens constitue l'aspect *formel* (ou syntaxique proprement dit) de la grammaire.

La définition du sens des phrases constitue la *sémantique*. Notons qu'au niveau syntaxique les objets élémentaires (les mots) ont un sens, et que le sens d'une phrase est fonction du sens des mots qui la composent. Il en est de même au niveau morphologique, où le sens d'un mot est fonction des sens des objets élémentaires (préfixes, racines, etc) qui le composent. Par contre au niveau lexical, les lettres n'ont pas de sens, et le sens d'un mot est arbitraire.

Exemple 2 : le français parlé.

Un mot (ou une phrase) prononcé est une suite continue de sons. L'alphabet phonétique définit un ensemble de notations de sons élémentaires tel que l'on puisse considérer tout mot (ou toute phrase) prononcé comme une suite discrète de tels objets élémentaires.

Exemple 3 : un langage de programmation (Ada, C, Java, etc.)

Les objets élémentaires sont les symboles de base (lettres de l'alphabet, chiffres, caractères spéciaux) dont la liste est donnée explicitement. Les unités ayant un sens sont les programmes. Ceux-ci sont définis par des diagrammes syntaxiques (ou des formes normales de Backus) complétées par des conditions supplémentaires (par exemple sur les déclarations des identificateurs) qui sont souvent exprimées en langue naturelle. Le sens des programmes est lui aussi souvent défini en langue naturelle. Une telle définition est un mélange de définitions formelles (les diagrammes syntaxiques) et de définitions informelles (les paragraphes en langue naturelle). On peut aussi donner des définitions formelles (c'est-à-dire n'utilisant pas la langue naturelle) des langages de programmation. Par exemple le sens d'un programme peut-être défini en lui associant une formule mathématique définissant la fonction qu'il calcule, ou un programme équivalent dans un autre langage de programmation; ainsi un compilateur définit le sens d'un programme par traduction dans un langage machine. On peut encore définir formellement la sémantique d'un langage au moyen d'un interpréteur de ce langage, le sens d'un programme est alors l'ensemble de ses exécutions pour toutes les données possibles. L'interpréteur est un programme qui peut être écrit dans un autre langage de programmation.

Notons que le mot "formel" signifie couramment "exprimé en langage mathématique", ou plus généralement "exprimé dans un formalisme" (c'est-à-dire dans un langage artificiel rigoureusement défini), "informel" signifiant "exprimé en langue naturelle" (c'est ainsi qu'il faut comprendre les termes "définition formelle", "définition informelle"); cependant on entend par *langage formel* un langage considéré comme un ensemble de suites d'objets élémentaires, abstraction faite du sens de ces suites. Cet aspect formel de la définition d'un langage est aussi appelé aspect syntaxique et doit être complété par la définition du sens des unités du langage, ce qui constitue l'aspect sémantique.

L'objet de ce cours est l'étude de certains formalismes utilisés pour la définition syntaxique de langages artificiels comme les langages de programmation ou les systèmes de notations mathématiques et logiques. Ces formalismes sont aussi utilisés pour étudier les langues naturelles,

mais ces applications ne seront pas développées.

Ce polycopié est une *introduction à la théorie des langages formels*. Un langage formel est un ensemble de séquences (ou *chaînes*) d'objets élémentaires, qui selon le langage défini, et le niveau auquel on se place, peuvent être des lettres, des mots, des symboles, des caractères, etc. Tous ces ensembles d'objets élémentaires ont en commun le fait d'être finis, et sont appelés *vocabulaire* (ou *alphabet*). Les définitions formelles ainsi que les opérations de base sur les chaînes et les langages sont définies au chapitre 1, où sont aussi données des propriétés élémentaires de la notion de relation, qui joue un rôle important dans la suite. La notion de langage comme ensemble de chaînes est trop générale : un langage doit de plus satisfaire des conditions de reconnaissabilité pour pouvoir être traité par des machines physiques (cas des langages de programmation), ou abstraite (cas des systèmes de notations logiques), ou être compris et énoncé par une personne (cas des langues naturelles, des systèmes de notations mathématiques, des langages de programmation évolués, etc). Le premier sous-ensemble de langages étudié est l'ensemble des *langages réguliers*. Ces langages peuvent être décrits grâce à un formalisme simple : les *automates finis*. Les langages réguliers et les automates finis sont étudiés dans les chapitres 2 à 5.

Le chapitre 6 introduit la notion de système de réécriture, qui joue un rôle fondamental dans la définition des machines abstraites comme les machines de Turing, et des automates et grammaires utilisés pour la reconnaissance et la génération des langages formels. Les grammaires les plus utilisées sont les grammaires hors-contexte, qui sont étudiées au chapitre 7.

Enfin les annexes contiennent des rappels mathématiques sur les relations d'ordre, les preuves par induction... ainsi que des éléments de correction des exercices.

Nb. Ce document est en constante évolution et certainement encore truffé de coquilles. Tout retour permettant son amélioration sera le bienvenu. Les auteurs remercient en particulier Nicolas Peltier, Cyril Lorenzetto, Aurélie Violette et Vincent Lefoulon pour leur relecture et contribution aux corrections des exercices du document.

Première partie

Introduction aux langages formels

Chapitre 1

Définitions de base

1 Prérequis sur les relations

1.1 Définitions

Définition 1.1.1 Soit E un ensemble. On note $\mathcal{P}(E)$ l'ensemble des sous-ensembles de E . Autrement dit, pour tout ensemble A , on a $(A \in \mathcal{P}(E))$ si et seulement si $(A \subset E)$. \diamond

Définition 1.1.2 On appelle *relation* (binaire) sur un ensemble E toute partie de $E \times E$. Etant donné une relation ρ sur E et deux éléments x et y de E , on dit que ρ est *vraie* pour le couple (x, y) si et seulement si $(x, y) \in \rho$. Dans la suite, on écrira souvent $x\rho y$ au lieu de $(x, y) \in \rho$. \diamond

Exemple 1.1.3 Pour tout ensemble E , \emptyset est la relation partout fausse, $E \times E$ est la relation partout vraie sur E . On note ι_E , ou ι s'il n'y a pas d'ambiguïté, la relation identité sur E , c'est-à-dire la partie de $E \times E$ définie par $\iota = \{(x, x) \mid x \in E\}$. La relation ι pourra aussi être dénotée par le symbole '='. \clubsuit

Proposition 1.1.4 L'ensemble des relations binaires sur E est $\mathcal{P}(E \times E)$.

Définition 1.1.5 On dit qu'une relation ρ sur E est :

- *réflexive* si $\forall x \in E, x\rho x$;
- *symétrique* si $\forall x, y \in E, x\rho y$ implique $y\rho x$;
- *antisymétrique* si $\forall x, y \in E$, si $x\rho y$ et $y\rho x$, alors $x = y$;
- *transitive* si $\forall x, y, z \in E, x\rho y$ et $y\rho z$ impliquent $x\rho z$. \diamond

Définition 1.1.6 Une relation réflexive et transitive est une *relation de préordre*. Une relation de préordre qui est symétrique est appelée une *relation d'équivalence*. Une relation de préordre qui est antisymétrique est appelée une *relation d'ordre partiel*. Une *relation d'ordre (total)* est une relation d'ordre partiel telle que $\forall x, y \in E, x\rho y$ ou $y\rho x$. \diamond

Définition 1.1.7 Une *relation fonctionnelle* sur E (ou fonction partielle de E dans E) est une relation ρ sur E telle que $\forall x, y, z \in E$, si $x\rho y$ et $x\rho z$ alors $y = z$. Une *fonction totale* (une *application*) de E dans E est une relation fonctionnelle ρ sur E telle que pour tout x de E il existe un y de E tel que $x\rho y$. \diamond

Les relations étant les parties de $E \times E$, on définit de la manière usuelle le complémentaire $\bar{\rho}$, la réunion $\rho \cup \sigma$ et l'intersection $\rho \cap \sigma$ de relations ρ et σ sur E .

Définition 1.1.8 (Opérations ensemblistes) Etant données deux relations ρ et σ , on définit les opérations $\rho \cup \sigma$, $\rho \cap \sigma$ et $\bar{\rho}$ de la façon suivante :

- $(x, y) \in \rho \cup \sigma$ si et seulement si $(x\rho y$ ou bien $x\sigma y)$;
- $(x, y) \in \rho \cap \sigma$ si et seulement si $(x\rho y$ et $x\sigma y)$;
- $x\bar{\rho}y$ si et seulement si $(x, y) \notin \rho$.

Etant données deux relations ρ et σ sur E , on dit que ρ implique σ (que σ contient ρ) si et seulement si pour tout $x, y \in E$, si $x\rho y$ alors $x\sigma y$. \diamond

Exercice 1.1.9 Soit $E = \{a, b, c, d, e\}$, et considérons la relation ρ définie par :

$$\rho = \{(a, a), (a, b), (a, c), (b, d), (b, e)\}.$$

Construire les relations suivantes :

1. La plus petite relation réflexive contenant ρ .
2. La plus petite relation symétrique contenant ρ .
3. La plus petite relation transitive contenant ρ .

Exercice 1.1.10 Soit E un ensemble fini quelconque. On rappelle qu'une partition de E est un ensemble $\{A_1, \dots, A_n\}$ tel que :

- pour tout $i = 1, \dots, n$, A_i est non-vide,
- $\bigcup_{i=1}^n A_i = E$,
- pour tout $i, j = 1, \dots, n$, si $i \neq j$ alors $A_i \cap A_j = \emptyset$.

Soit ρ une relation d'équivalence sur E . Pour un élément $x \in E$, la classe d'équivalence de x est l'ensemble $[x]_\rho = \{y \in E \mid x\rho y\}$.

1. Montrer que si $y \in [x]_\rho$, alors $[x]_\rho = [y]_\rho$.
2. Montrer que l'ensemble des classes d'équivalence des éléments de E forme une partition de E .

Exercice 1.1.11 Prouver que $(\rho$ implique $\sigma)$ si et seulement si $(\bar{\rho} \cup \sigma = E \times E)$.

Définition 1.1.12 (Composition de relations) La composition des relations sur E est l'opération sur $\mathcal{P}(E \times E)$ définie par de la façon suivante :

$$\forall \rho, \sigma \in \mathcal{P}(E \times E), \rho \cdot \sigma = \{(x, y) \in E \times E \mid \exists z \in E, (x, z) \in \rho \text{ et } (z, y) \in \sigma\}. \quad \diamond$$

Exercice 1.1.13 Reprendre la relation ρ définie dans l'Exercice 1.1.9. Construire la relation $\rho \cdot \rho$.

Remarque La composition des relations est associative et admet comme élément neutre la relation identité ι . Autrement dit, $\langle \mathcal{P}(E \times E), \cdot, \iota \rangle$ est un monoïde (voir la Définition 1.2.15).

Exercice 1.1.14 Montrer qu'une relation ρ sur E est transitive si et seulement si $\rho \cdot \rho \subset \rho$.

Définition 1.1.15 (Itérations d'une relation) On définit pour toute relation ρ sur E et tout entier $n \geq 0$ la relation ρ^n par :

- $\rho^0 = \iota$
- $\forall n > 0, \rho^n = \rho \cdot \rho^{n-1}$. ◇

On définit enfin pour toute relation ρ les relations ρ^* et ρ^+ par :

$$\rho^* = \bigcup_{n \geq 0} \rho^n \text{ et } \rho^+ = \bigcup_{n > 0} \rho^n.$$

On vérifie immédiatement que, pour tout $n > 0$, $x\rho^n y$ si et seulement si il existe une suite x_1, \dots, x_{n+1} d'éléments de E tels que $x_1 = x, x_{n+1} = y$ et pour tout i tel que $1 \leq i \leq n, x_i \rho x_{i+1}$. Il est également clair que :

- $x\rho^* y$ si et seulement si $x = y$ ou $x\rho^+ y$
- $x\rho^+ y$ si et seulement si il existe un entier $n > 0$ et une suite x_1, \dots, x_{n+1} d'éléments de E tels que $x_1 = x, x_{n+1} = y$ et pour tout $i, 1 \leq i \leq n, x_i \rho x_{i+1}$.

Définition 1.1.16 On appelle *fermeture transitive* d'une relation ρ sur E la plus petite relation transitive sur E contenant ρ , c'est-à-dire la relation sur E qui contient ρ , qui est transitive et qui est incluse dans toute relation transitive contenant ρ . ◇

Proposition 1.1.17 Pour toute relation ρ sur un ensemble E , ρ^+ est la fermeture transitive de ρ .

PREUVE. Il est clair que ρ^+ contient ρ et est transitive. Soit maintenant σ une relation transitive contenant ρ . Si $x\rho^+ y$, alors il existe x_2, \dots, x_n , avec $n > 0$, tels que $x\rho x_2 \dots x_n \rho y$. Comme σ contient ρ , on a $x\sigma x_2 \dots x_n \sigma y$ et, σ étant transitive, $x\sigma y$. ■

Remarque La proposition ci-dessus prouve donc que la fermeture transitive d'une relation existe toujours et est unique.

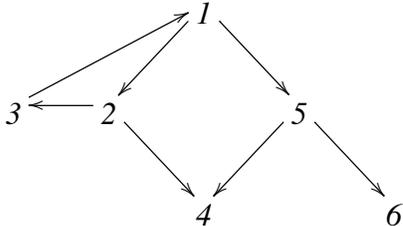
Définition 1.1.18 On appelle *fermeture transitive et réflexive* d'une relation ρ la plus petite relation transitive et réflexive contenant ρ . ◇

Proposition 1.1.19 Pour toute relation ρ sur un ensemble E , ρ^* est la fermeture transitive et réflexive de ρ .

PREUVE. exercice. ■

Exercice 1.1.20

1. Soit ρ la relation sur l'ensemble $\{1, 2, 3, 4, 5, 6\}$ définie par le graphe de la figure ci-dessous. Tracer les graphes des relation ρ^+ et ρ^* .



2. Définir un algorithme qui, étant donnée une relation sur un ensemble fini, calcule sa fermeture transitive.

2 Mots et langages

2.1 Mots

Définition 1.2.1 (Vocabulaire) On appelle *vocabulaire* (ou *alphabet*) un ensemble fini quelconque. Les éléments d'un vocabulaire sont appelés *lettres*, *caractères* ou *symboles*. On note $|V|$ (ou bien $\#V$) le nombre d'éléments d'un vocabulaire V . \diamond

N'importe quel ensemble fini peut représenter un vocabulaire ; ainsi, les ensembles suivants sont des vocabulaires :

- $V_1 = \{a, b\}$ (pour construire des mots avec les lettres a et b),
- $V_2 = \{0, 1, \dots, 9\}$ (pour construire des mots représentant des nombres),
- $V_3 = \{\underline{\text{procedure}}, _ , (, \underline{\text{begin}}, \underline{\text{end}}\}$ (pour déclarer des procédures).

Définition 1.2.2 (Mot) On appelle *mot* (ou *chaîne*, ou *phrase*) sur un vocabulaire V toute suite finie a_1, a_2, \dots, a_n d'éléments de V . Lorsque V est uniquement constitué de symboles simples (lettres, chiffres), on note usuellement un mot en écrivant en séquence, sans séparateur, les lettres qui la composent.

Par convention, il existe un mot qui ne contient aucun élément : la *chaîne vide* ; cette dernière est notée ε . \diamond

Exemple 1.2.3 aba est un mot sur le vocabulaire $\{a, b\}$.

$\underline{\text{début}}x := 1 \underline{\text{fin}}$ est un mot sur le vocabulaire $\{\underline{\text{début}}, \underline{\text{fin}}, :=, 1, x\}$, qu'il est préférable d'écrire en utilisant l'espace comme séparateur : $\underline{\text{début}} x := 1 \underline{\text{fin}}$. \clubsuit

Remarque Un vocabulaire V peut être un ensemble fini de chaînes, comme dans le second exemple ci-dessus. L'écriture des chaînes sur V sans séparateurs entre les éléments de V peut alors produire des ambiguïtés. Par exemple, si $V = \{a, aa\}$, il faut un séparateur pour distinguer les deux chaînes a,aa et aa,a .

Définition 1.2.4 (Longueur d'un mot) On appelle *longueur* d'un mot le nombre d'éléments de la suite le définissant. La longueur d'un mot x sera notée $|x|$. \diamond

Exemple 1.2.5 Avec les exemples ci-dessus, on a :

- $|aba| = 3$,
- $|\underline{\text{début}}x := 1\underline{\text{fin}}| = 5$. \clubsuit

Par convention, le mot vide est l'unique chaîne de longueur 0.

Définition 1.2.6 Etant donné un vocabulaire V , on note V^n , où n est un entier positif ou nul, l'ensemble des chaînes sur V de longueur n . \diamond

Proposition 1.2.7 Pour tout vocabulaire V , on a $V^0 = \{\varepsilon\}$: le mot vide est un mot sur tout vocabulaire.

Remarque Par convention, on pose $V^1 = V$: les chaînes de longueur 1 sont identifiées aux lettres.

Exercice 1.2.8 Montrer que V^n contient $|V|^n$ éléments (on pourra utiliser la convention $0^0 = 1$).

Définition 1.2.9 (Ensembles de chaînes sur un vocabulaire) Etant donné un vocabulaire V , on note V^* l'ensemble des chaînes sur V , et V^+ l'ensemble des chaînes non vides sur V . Autrement dit, $V^* = \bigcup_{i \geq 0} V^i$ et $V^+ = \bigcup_{i > 0} V^i$. \diamond

Propriété 1.2.10

- Si $V \neq \emptyset$ alors V^* est infini.
- $\emptyset^* = \{\varepsilon\}$.

Définition 1.2.11 (Concaténation) La *concaténation* est l'opération qui associe à deux mots x et y le mot noté $x.y$ ou xy , défini comme suit : si $x = a_1a_2 \cdots a_n$ et $y = b_1b_2 \cdots b_p$ alors $x.y = c_1c_2 \cdots c_{n+p}$, où $c_i = a_i$ pour $i = 1, \dots, n$ et $c_{n+i} = b_i$ pour $i = 1, \dots, p$. \diamond

Par exemple, la concaténation des mots aba et cab donne le mot $abacab$.

On vérifie facilement que la concaténation est une opération associative admettant le mot vide comme élément neutre. Soit :

- $\forall x, y, z \in V^*$, $x.(y.z) = (x.y).z$
- $\forall x \in V^*$, $x.\varepsilon = \varepsilon.x = x$.

Définition 1.2.12 (Sous-mots) Etant donné un mot w sur un vocabulaire V , un mot u est un *sous-mot* de w s'il existe $x, y \in V^*$ tels que $w = xuy$. Le sous-mot u est un *préfixe* de w si $x = \varepsilon$; un *suffixe* de w si $y = \varepsilon$. \diamond

Exemple 1.2.13 Le mot $abba$ admet les sous-chaînes $\varepsilon, a, b, ab, bb, ba, abb, bba, abba$. Il existe deux occurrences de chacune des sous-chaînes a et b . Les préfixes de $abba$ sont ε, a, ab, abb et $abba$; ses suffixes sont ε, a, ba, bba et $abba$. \clubsuit

Exercice 1.2.14

1. Déterminer les nombres maximal et minimal de sous-chaînes, de préfixes et de suffixes d'un mot de longueur donnée.
2. Montrer que toute chaîne sur $\{a, b\}$, de longueur supérieure ou égale à 4, admet deux occurrences consécutives d'une même sous-chaîne non vide.

Pour aller plus loin

Définition 1.2.15 (Monoïde) On appelle *monoïde* un ensemble muni d'une opération associative admettant un élément neutre. Un monoïde est représenté par un triplet $\langle E, \cdot, i \rangle$, où E est l'ensemble considéré, " \cdot " est l'opération associative sur E et $i \in E$ est l'élément neutre de cette opération (l'identité). \diamond

Exemple 1.2.16 Comme exemples de monoïdes, citons $\langle \mathbb{N}, +, 0 \rangle$ et $\langle \mathbb{N}, \cdot, 1 \rangle$, où $+$ et \cdot dénotent respectivement l'addition et la multiplication des entiers. \clubsuit

Proposition 1.2.17 Pour tout vocabulaire V , $\langle V^*, \cdot, \varepsilon \rangle$ est un monoïde ; dans la suite, nous mentionnerons le monoïde V^* pour signifier le monoïde $\langle V^*, \cdot, \varepsilon \rangle$.

Proposition 1.2.18 Soit V un vocabulaire quelconque. Tout élément de V^* est de la forme $a_1 \cdot \dots \cdot a_n$, où pour tout $i = 1, \dots, n$, $a_i \in V$.

Cette propriété distingue le monoïde V^* par exemple du monoïde $(V \cup \{a\})^*$, avec $a \notin V$. Elle distingue également le monoïde $\{a, b\}^*$ par exemple du monoïde commutatif M obtenu en postulant que l'opération de concaténation est commutative : les deux suites ab et ba d'éléments de V définissent alors le même élément de M .

V^* est le seul monoïde construit à partir de V satisfaisant cette propriété : on dit que V^* est le *monoïde libre sur V* .

Définition 1.2.19 (Homomorphismes de monoïdes) Etant donnés $M = \langle E, \cdot, i \rangle$ et $M' = \langle E', \circ, i' \rangle$ deux monoïdes, on appelle homomorphisme de M dans M' toute application h de E dans E' telle que :

- $h(i) = i'$
- $\forall x, y \in E, h(x \cdot y) = h(x) \circ h(y)$.

Un isomorphisme de monoïdes est un homomorphisme bijectif de monoïdes. \diamond

Si h est un homomorphisme de V^* dans un monoïde quelconque M , alors h est défini par sa valeur pour chaque lettre a de V . En d'autres termes, si h et h' sont deux homomorphismes de V^* dans M tel que $\forall a \in V, h(a) = h'(a)$, alors h et h' sont identiques. Il en résulte immédiatement la propriété suivante :

Propriété 1.2.20 Soit M un monoïde quelconque et f une application d'un vocabulaire V dans M . Alors il existe un homomorphisme unique h de V^* dans M qui prolonge f , c'est-à-dire tel que $\forall a \in V, h(a) = f(a)$.

Considérons maintenant un monoïde L satisfaisant la propriété P suivante :

Soit M un monoïde quelconque et f une application d'un vocabulaire V dans M ; alors il existe une application *injective* g de V dans L et un homomorphisme unique h de L dans M tel que $\forall a \in V, h(g(a)) = f(a)$.

L'application g définit une représentation des lettres de V par des éléments de L et l'homomorphisme h étend à tous les éléments de L l'application f . On peut montrer que deux monoïdes satisfaisant la propriété P sont isomorphes.

Le monoïde V^* satisfait la propriété P et tout monoïde L satisfaisant P est alors isomorphe à V^* . On peut donc utiliser la propriété P pour donner une définition algébrique de la notion de monoïde libre sur un vocabulaire V . Cette notion est alors définie à un isomorphisme près (c'est-à-dire à la représentation près des chaînes sur V ; ceci était implicite dans la première définition).

Exercice 1.2.21 *Montrer que l'application "longueur d'un mot" est un homomorphisme de monoïde de $\langle V^*, \cdot, \varepsilon \rangle$ dans $\langle \mathbb{N}, +, 0 \rangle$.*

Cette application est un isomorphisme si $\#V = 1$: si V ne contient qu'un élément, $\langle \mathbb{N}, +, 0 \rangle$ est le monoïde libre sur V car on peut alors représenter sans ambiguïté les chaînes sur V par leurs longueurs.

Définition 1.2.22 On pose, pour toute chaîne x sur un vocabulaire V ,

- $x^0 = \varepsilon$
- pour tout entier $n > 0$, $x^n = x.x^{n-1} = x.x \dots x$ avec n facteurs. ◇

Proposition 1.2.23 *Comme pour tout monoïde multiplicatif, on a pour tout n, m les propriétés :*

- $x^{n+m} = x^n . x^m$
- $x^{nm} = (x^n)^m$

2.2 Langages

Définition 1.2.24 On appelle *langage* sur un vocabulaire V tout sous-ensemble de V^* . ◇

Exemple 1.2.25

- Un langage de programmation est un langage sur l'ensemble de ses symboles de base.
- L'ensemble des phrases de la langue française est un langage sur l'ensemble des mots du français ; l'ensemble des mots du français est lui-même un langage sur l'ensemble des lettres de l'alphabet.
- L'ensemble des identificateurs est un langage sur $\{A, B, \dots, Z, a, b, \dots, z, 0, 1, \dots, 9\}$.
- $\emptyset, V^1, V^2, \dots, V^n, V^*$ sont des langages sur V . On appelle \emptyset le langage vide. ♣

Un langage est dit fini ou infini selon qu'il comprend un nombre fini ou infini de chaînes.

Propriété 1.2.26 Un langage L est infini si et seulement si pour tout $n \in \mathbb{N}$, il existe $x \in L$ tel que $|x| > n$.

Soit encore, de façon équivalente : un langage L est fini si et seulement s'il existe une borne à la longueur des chaînes de L .

Cette propriété est une conséquence immédiate du fait que tout vocabulaire est un ensemble fini.

Exemple 1.2.27 Le langage des identificateurs de Java est infini car il n'y a pas, dans la définition de Java, de borne pour la longueur des identificateurs. ♣

Définition 1.2.28 (Opérations sur les langages) Etant donnés les langages L_1 et L_2 sur un vocabulaire V , on définit les opérations suivantes :

- L'union : $L_1 \cup L_2 = \{w \in V^* \mid w \in L_1 \vee w \in L_2\}$.
- L'intersection : $L_1 \cap L_2 = \{w \in V^* \mid w \in L_1 \wedge w \in L_2\}$.
- Le complémentaire : $\overline{L_1} = \{w \in V^* \mid w \notin L_1\}$.
- La différence : $L_1 - L_2 = \{w \in V^* \mid w \in L_1 \wedge w \in \overline{L_2}\}$.
- La concaténation : $L_1.L_2 = \{w_1.w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$. ◇

Exemple 1.2.29 Si $L_1 = \{a^n b^n \mid n \geq 0\}$ et $L_2 = \{c^n d^n \mid n \geq 0\}$ alors $L_1.L_2 = \{a^p b^p c^q d^q \mid p, q \geq 0\}$. ♣

Remarque La concaténation des langages est associative et admet le langage $\{\epsilon\}$ comme élément neutre. Autrement dit $\langle \mathcal{P}(V^*), \cdot, \{\epsilon\} \rangle$ est un monoïde.

Exercice 1.2.30

1. Prouver que, pour tout $L \in V^*$, $\emptyset.L = L.\emptyset = \emptyset$.
2. Etudier le monoïde $\langle \mathcal{P}(V^*), \cdot, \{\epsilon\} \rangle$ quand $V = \emptyset$.

Remarque Il ne faut pas confondre le langage vide \emptyset , qui ne contient aucun mot, avec le langage $\{\epsilon\}$, qui contient le mot vide ϵ . Le premier est l'élément absorbant et le second est l'élément neutre de la concaténation des langages.

Définition 1.2.31 L'élevation à une puissance entière d'un langage L est définie comme pour tout monoïde multiplicatif :

- $L^0 = \{\epsilon\}$
- pour tout $n > 0$, $L^n = L.L^{n-1} = L.L \dots L$ avec n facteurs.

On définit enfin les opérations de concaténation itérée $*$ et $+$ par : $L^* = \bigcup_{i \geq 0} L^i$ et $L^+ = \bigcup_{i > 0} L^i$. ◇

Quand le langage L est fini, il peut être considéré comme un vocabulaire. Les notations L^n , L^* et L^+ sont alors ambiguës : elles désignent aussi, respectivement, les chaînes sur L de longueur n , l'ensemble des chaînes sur L et l'ensemble des chaînes non vides sur L (voir les Définitions 1.2.4 et 1.2.9). Par exemple, si $L = \{a, aa\}$, L^2 est d'une part le langage qui est la concaténation de L à lui-même, d'autre part l'ensemble des chaînes de longueur 2 sur L . Dans le premier cas, on a $L^2 = \{aa, aaa, aaaa\}$, dans le second cas $L^2 = \{a.a, a.aa, aa.a, aa.aa\}$.

Chapitre 2

Automates finis

1 Introduction

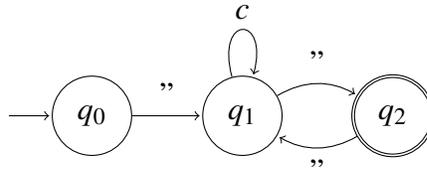
Un automate fini est une machine abstraite qui effectue des calculs en utilisant une mémoire de taille bornée. Il n'est pas possible d'augmenter la taille de la mémoire au delà de cette borne. Les tailles des données et des résultats peuvent être supérieures à cette borne puisque les données sont lues, les résultats produits progressivement au cours du calcul. La longueur d'un calcul peut aussi être supérieure à la taille de la mémoire de l'automate ; celui-ci passe alors nécessairement plusieurs fois par un même état de sa mémoire. Les automates finis ont des applications importantes : définition de certains aspects des langages naturels ou artificiels (en particulier de la lexicographie), description de machines physiques (circuits électroniques, machines à calculer, distributeur d'objets, etc.), définition de protocoles de communication dans des réseaux, description de systèmes de commandes (comme le système de commandes d'un ascenseur), etc. Les automates finis peuvent être utilisés pour calculer des fonctions, ou pour reconnaître des langages. On montre que les langages reconnus par les automates finis coïncident avec les langages réguliers, c'est-à-dire les langages qui sont les valeurs des expressions régulières. Les langages réguliers comprennent, outre les langages finis, les langages infinis que l'on peut définir en utilisant l'union, la concaténation et l'opération de concaténation itérée (l'étoile). Un ordinateur - en faisant abstraction des mémoires périphériques - peut être décrit par un automate fini, puisque sa mémoire centrale est de taille bornée. Mais ce type de descriptions n'a que peu d'intérêt, au vu de la taille très grande des mémoires des ordinateurs. Un meilleur modèle est fourni par les machines de Turing, qui utilisent une mémoire finie non bornée.

Exemple 2.1.1 (Chaînes de caractères d'un langage de programmation) Cet ensemble est défini informellement de la façon suivante :

Toute chaîne commence et se termine par le caractère " (guillemets); toute occurrence du caractère " (guillemets) à l'intérieur d'une chaîne doit être doublée.

Par exemple, la chaîne de caractères je dis "oui" s'écrit : "je dis ""oui"""; la chaîne de caractères de longueur 1 se composant d'un seul guillemet s'écrit : """". Cet ensemble peut être défini

plus formellement par l'expression régulière¹ $(c^* + "")^*$, où c désigne un caractère quelconque différent de ". Le dessin ci-dessous représente un automate fini reconnaissant l'ensemble des chaînes de caractères.



Ces chaînes de caractères sont les chemins reliant le sommet le plus à gauche (l'état initial de l'automate) au sommet le plus à droite (l'état final de l'automate). Comme précédemment, le symbole c désigne un caractère quelconque différent du caractère ". ♣

Exemple 2.1.2 (Déclarations de procédure dans un langage de programmation.) Informellement, une déclaration de procédure commence par le mot clé `procedure`, suivi d'un identificateur qui est le nom de la procédure. On a ensuite la déclaration de la liste des paramètres (cette partie peut être vide), puis, après un ";", le corps de la procédure. Chaque paramètre est un identificateur, auquel est associé un type. On sépare un paramètre de son type par le caractère ":". On peut mettre en facteur des paramètres ayant la même type. Le corps de la procédure commence par le mot clé `début`, suivi d'une liste non vide d'instructions, et se termine par le mot clé `fin`. Les instructions sont séparées par le caractère ";".

Dans l'automate de la Figure 2.1, qui contient 12 états, les déclarations de procédures sont définies par l'ensemble des chemins de l'état initial (en haut à gauche) à l'état final (en haut à droite). On n'a pas défini les identificateurs, les types et les instructions. Il n'y a pas de difficulté à définir les identificateurs avec un automate fini. Par contre, si les instructions peuvent contenir des déclarations de procédures, on ne peut pas décrire les déclarations de procédure par un automate fini : un formalisme équivalent aux grammaires hors-contexte est alors nécessaire. ♣

Nous verrons au paragraphe 6 de ce chapitre que les fonctionnalités d'un automate fini peuvent être étendues en lui ajoutant une fonction de sortie. L'exemple suivant montre comment se servir d'un tel automate pour calculer la somme de deux entiers binaires.

Exemple 2.1.3 (Somme de deux entiers notés en binaire) Les deux entiers à additionner sont écrits sur le même nombre de chiffres binaires. L'automate lit simultanément les deux opérandes, en commençant pas les chiffres des unités, et produit le résultat chiffre par chiffre en commençant par le chiffre des unités. Ses entrées sont donc les couples de chiffres binaires. Pour chaque entrée, l'automate produit un résultat qui est un chiffre binaire. La notation $00 / 1$ signifie : pour l'entrée 00 , on produit 1 .

L'automate a deux états : l'état s correspond à l'absence de retenue, l'état r à la présence d'une retenue. Deux entiers à additionner définissent un chemin de l'automate, commençant par

1. Les expressions régulières seront définies avec précision au Chapitre 4.

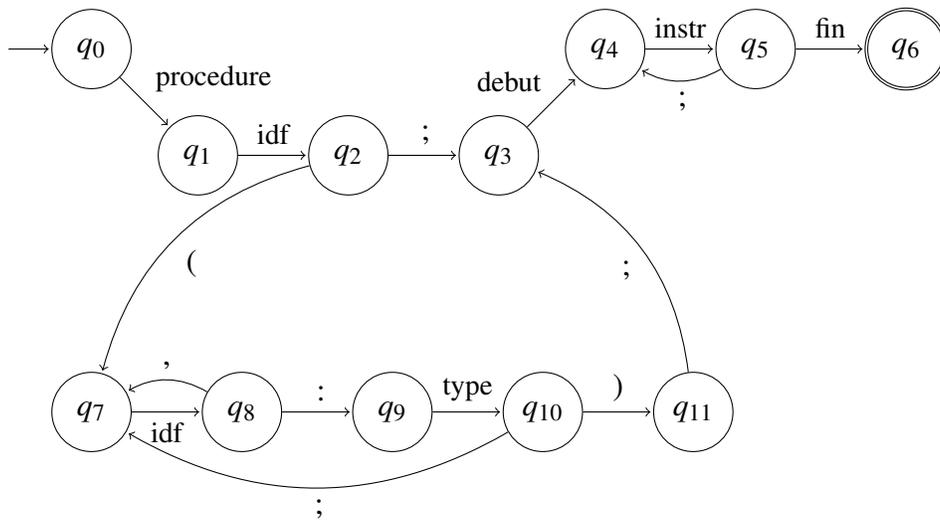
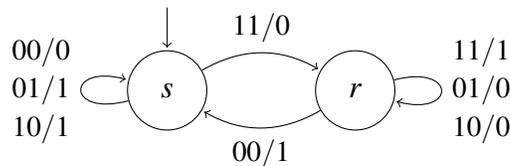


FIGURE 2.1 – Déclarations de procédures



l'état sans retenue s . Si ce chemin se termine en s , il définit aussi le résultat de l'addition ; s'il se termine en r , il faut encore écrire un 1 pour la retenue. ♣

La suite de ce chapitre comprend :

- La définition des automates finis reconnaissant des langages, comme ceux des exemples 2.1.1 et 2.1.2, ainsi que l'énumération de certaines de leurs propriétés.
- Les preuves de l'équivalence des automates finis avec des automates finis particuliers : automates sans ε -transitions, automates déterministes.
- Les automates définissant des fonctions, comme celui de l'exemple 2.1.3, - encore appelés machines finies - sont étudiés brièvement au paragraphe 6.

Le problème de la minimisation des automates finis est traité au chapitre 3, et la preuve de l'équivalence des automates finis et des expressions régulières fait l'objet du chapitre 4.

2 Automates finis et langages

Définition 2.2.1 (automate fini) Un *automate fini* est un quintuplet $\langle Q, V, \delta, I, F \rangle$ où :

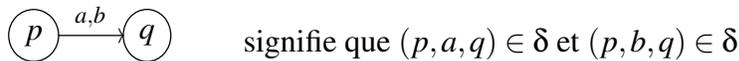
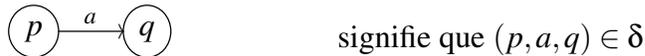
- Q est un ensemble fini d'états,

- V est un ensemble fini de symboles (le vocabulaire de l'automate),
- δ est un sous-ensemble de $Q \times (V \cup \{\epsilon\}) \times Q$ appelé la relation de transition,
- I est un sous-ensemble de Q , l'ensemble des états initiaux,
- F est un sous-ensemble de Q , l'ensemble des états finals (*sic*).

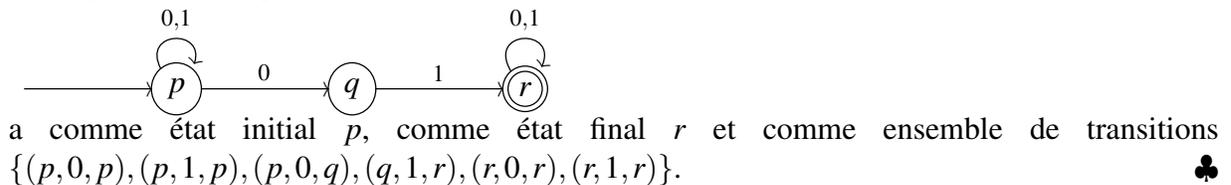
Une transition $(p, a, q) \in \delta$ est appelée une a -transition. L'état p est l'origine et l'état q est l'extrémité de cette transition.

On peut aussi voir δ comme une application de $Q \times V \cup \{\epsilon\}$ dans $\mathcal{P}(Q)$ en posant pour tout $p \in Q$ et tout $a \in V \cup \{\epsilon\}$: $\delta(p, a) = \{q \mid (p, a, q) \in \delta\}$ \diamond

Un automate ayant δ comme ensemble de transitions est dessiné avec les conventions suivantes :



Exemple 2.2.2 D'après ces conventions, l'automate ci-dessous :



Définition 2.2.3 (chemins et traces) Soit $A = \langle Q, V, \delta, I, F \rangle$ un automate fini. Un *chemin* de A de longueur n est une suite de transitions (r_i, a_{i+1}, r_{i+1}) où $0 \leq i < n$. Ce chemin mène de l'état r_0 (son *origine*) à l'état r_n (son *extrémité*) avec la *trace* $a_1 \dots a_n$. Par convention, la suite vide est un chemin de longueur 0 et de trace ϵ conduisant d'un état à lui-même. \diamond

Dans l'exemple ci-dessus, $(p, 1, p)(p, 0, q)(q, 1, r)(r, 1, r)$ est un chemin de p à r dont la trace est 1011.

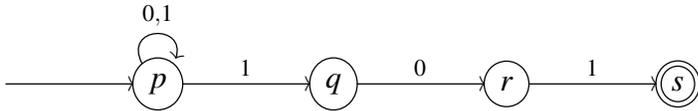
Définition 2.2.4 (mots et langage reconnu par un automate) Le mot x est *reconnu* (ou *accepté*) par un automate s'il y a un chemin de cet automate de trace x menant d'un état initial à un état final. On note $L(A)$ l'ensemble des mots reconnus par l'automate A ; $L(A)$ est le *langage reconnu* par l'automate A .

Un langage reconnu par un automate fini est appelé *langage régulier*, ou bien *langage d'états finis*. \diamond

Définition 2.2.5 Deux automates sont *équivalents* s'ils reconnaissent le même langage. \diamond

Le mot 01 est reconnu par l'automate de l'exemple 2.2.2; comme on le verra plus tard, le langage reconnu par l'automate est l'ensemble $\{0, 1\}^* \{01\} \{0, 1\}^*$; c'est l'ensemble de mots sur $\{0, 1\}$ comportant le sous-mot 01.

Exemple 2.2.6 Considérons l'automate suivant sur le vocabulaire $\{0, 1\}$:



$(p, 0, p)(p, 1, q)(q, 0, r)(r, 1, s)$ est un chemin menant de p à s avec la trace 0101. Ce mot est reconnu par l'automate car p est un état initial et s est un état final.

Le langage reconnu par l'automate est défini par l'ensemble $\{0, 1\}^* \{101\}$, qui caractérise l'ensemble de mots sur $\{0, 1\}$ qui se terminent par 101. \clubsuit

Exercice 2.2.7 Définir un algorithme, qui, étant donné un automate fini A et une chaîne x sur le vocabulaire de A , décide si $x \in L(A)$.

Décomposition de traces

Propriété 2.2.8 Soient A un automate fini de vocabulaire V , x et y deux chaînes sur V , p et q deux états de A . Un chemin mène de p à q avec la trace xy si et seulement s'il existe un état r tel qu'un chemin mène de p à r avec la trace x et de r à q avec la trace y .

PREUVE. La condition suffisante est évidente et nous prouvons seulement la condition nécessaire. Soit un chemin (r_i, a_{i+1}, r_{i+1}) , pour i de 0 à $n-1$, menant de l'état p à l'état q avec la trace xy . Puisque les symboles de chaque transition sont éléments de V ou sont égaux à ϵ , il existe k tel que $0 \leq k \leq n$ et $x = a_1 \dots a_k$. Par suite, il existe un chemin conduisant de p à r_k avec la trace x et de r_k à q avec la trace y . \blacksquare

Propriété 2.2.9 Soient A un automate fini de vocabulaire V , a un élément de V et p, q deux états de A . Il existe un chemin de trace a entre p et q , si et seulement s'il existe deux états r et s tels que, un chemin de trace ϵ mène de p à r , (r, a, s) est une transition et un chemin de trace ϵ mène de s à q .

PREUVE. Par définition, il existe un chemin (r_i, a_{i+1}, r_{i+1}) de A , où $0 \leq i < n$, tel que $r_0 = p$, $r_n = q$ et $a = a_1 \dots a_n$. Puisque $|a| = 1$, il existe un unique $i \in [1, n]$ tel que $a_i \neq \epsilon$. On a donc : $\forall j \in [1, n] \setminus \{i\} a_j = \epsilon$, et $a_i = a$. Prenons alors $r = r_{i-1}$ et $s = r_i$. La séquence (r_j, a_{j+1}, r_{j+1}) (pour $0 \leq j \leq i-2$) est un chemin dans A de $r_0 = p$ vers $r_{i-1} = r$ et de trace $a_1 \dots, a_{i-1} = \epsilon$, (r_j, a_{j+1}, r_{j+1}) (pour $i \leq j \leq n$) est un chemin dans A de $r_i = s$ vers $r_n = q$ et de trace $a_{i+1} \dots, a_n = \epsilon$, et (r, a, s) est une transition de A . \blacksquare

3 Automates finis sans ε -transition

Le but de cette partie est de démontrer le théorème suivant :

Théorème 2.3.1 *Tout automate fini est équivalent à un automate fini sans ε -transition.*

La démonstration de ce résultat est constructive : elle fournit un algorithme permettant, pour un automate A donné, de construire un automate sans ε -transition équivalent à A .

PREUVE. Soit $A = \langle Q, V, \delta, I, F \rangle$ un automate fini.

Soit $B = \langle Q, V, \eta, I, G \rangle$ l'automate fini sans ε transition ainsi défini :

- Pour tout $a \in V$, pour tous $p, q \in Q$, $(p, a, q) \in \eta$ si et seulement s'il existe un état r dans A tel que $(r, a, q) \in \delta$, et un chemin de A de trace ε d'origine p et d'extrémité r .
- $G = F \cup \{p \in Q \mid \text{il existe un chemin de trace } \varepsilon \text{ entre } p \text{ et un état de } F\}$.

Une façon équivalente de décrire la construction de B est la suivante :

- Pour tout $p \in Q$, soit $f(p) = \{q \in Q \mid \text{il existe un chemin de trace } \varepsilon \text{ entre } p \text{ et } q\}$.
- Pour tout $a \in V$, pour tout $p \in Q$, $\eta(p, a) = \bigcup_{r \in f(p)} \delta(r, a)$.
- $G = \{p \in Q \mid f(p) \cap F \neq \emptyset\}$.

Prouvons que les automates A et B sont équivalents.

Cette démonstration se fait en deux étapes. On montre d'abord pour tout mot x la propriété $P(x)$ suivante :

Il existe un chemin de A de trace x entre l'état p et l'état q si et seulement s'il existe un état r et un chemin de B menant de p à r avec la trace x et un chemin de A menant de r à q avec la trace ε .

Ce résultat est ensuite utilisé pour prouver que les automates A et B sont équivalents.

Preuve de la propriété.

On procède par induction sur la longueur de x . Pour $x = \varepsilon$, $P(x)$ est vérifiée en choisissant $r = p$. Supposons maintenant $P(x)$ vraie et montrons $P(ax)$ pour $a \in V$.

Il existe un chemin de A de trace ax entre p et q

\Leftrightarrow (propriété 2.2.8) il existe un état t tel qu'un chemin de A mène de p à t avec la trace a , et un chemin de A mène de t à q avec la trace x .

\Leftrightarrow (propriété 2.2.9) il existe des états r, s, t tels qu'un chemin de A de trace ε mène de p à r , $(r, a, s) \in \delta$, un chemin de A de trace ε mène de s à t , et un chemin de A de trace x mène de t à q .

\Leftrightarrow (définition de B) il existe des états s, t tels que $(p, a, s) \in \eta$, un chemin de A de trace ε mène de s à t , et un chemin de A de trace x mène de t à q .

\Leftrightarrow (définition des traces) il existe des états s, t tels que $(p, a, s) \in \eta$, un chemin de A de trace x mène de s à q .

\Leftrightarrow (hypothèse d'induction) il existe des états s, u tels que $(p, a, s) \in \eta$, un chemin de B de trace x mène de s à u et un chemin de A de trace ε mène de u à q

\Leftrightarrow (définition des traces) il existe un état u tel que un chemin de B de trace xa mène de p à u et un chemin de A de trace ε mène de u à q .

Ceci établit la propriété $P(ax)$, donc prouve $P(x)$ pour toute chaîne x .

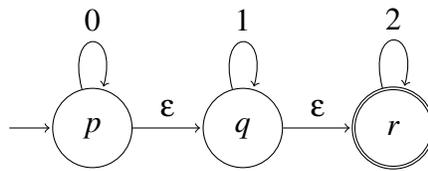


FIGURE 2.2 – Automate de l'exemple 2.3.2

Equivalence de A et B

La propriété P ayant été prouvée, nous prouvons maintenant l'équivalence des deux automates. Soit $x \in V^*$,

$$x \in L(A)$$

\Leftrightarrow il existe un chemin de A de trace x entre un état initial et un état final de A.

\Leftrightarrow (propriété $P(x)$) il existe un état r et un chemin de B de trace x entre un état initial de B et r et un chemin de A de trace ϵ entre r et un état final de A.

\Leftrightarrow (définition des états finals de B) il y a un chemin de B de trace x entre un état initial et un état final de B.

$$\Leftrightarrow x \in L(B).$$



Exemple 2.3.2 On élimine les ϵ transitions de l'automate A de la figure 2.2, ayant pour vocabulaire $\{0, 1, 2\}$.

Pour chaque état s , on calcule $f(s)$ l'ensemble des états atteignables avec la trace ϵ :

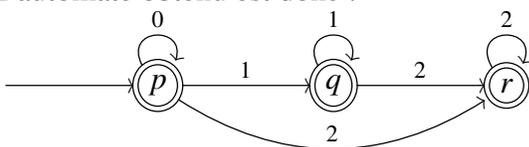
état	atteignables
p	$\{p, q, r\}$
q	$\{q, r\}$
r	$\{r\}$

Puisqu'il y a des chemins de trace ϵ entre p et l'état final r ainsi qu'entre q et r , ces deux états deviennent aussi des états finals.

La fonction de transition de l'automate sans ϵ -transitions est présentée sous forme d'une table dont les cases sont les valeurs de $\eta(s, a)$ pour chaque état s et symbole a .

	0	1	2
p	$\{p\}$	$\{q\}$	$\{r\}$
q	\emptyset	$\{q\}$	$\{r\}$
r	\emptyset	\emptyset	$\{r\}$

L'automate obtenu est donc :



4 Automates finis déterministes

Définition 2.4.1 (automate déterministe) Un automate fini *déterministe* est un automate fini $\langle Q, V, \delta, I, F \rangle$ n'ayant qu'un état initial, sans ε -transition, tel que pour tout état $p \in Q$ et tout symbole $a \in V$, il existe *un et un seul* état $q \in Q$ tel que $(p, a, q) \in \delta$. \diamond

Remarque Dans la description d'un tel automate, l'ensemble I des états initiaux peut être remplacé par son seul état. Ainsi, si $I = \{q_0\}$, on pourra noter $A = \langle Q, V, \delta, q_0, F \rangle$ au lieu de $A = \langle Q, V, \delta, \{q_0\}, F \rangle$

Puisque pour tout état p et symbole a il existe un et un seul état q tel que $(p, a, q) \in \delta$, la relation de transition d'un automate déterministe est une fonction *totale* de $Q \times V$ dans Q . On peut donc noter $\delta(p, a) = q$, au lieu de $(p, a, q) \in \delta$, et dire que δ est une *fonction de transition*.

Remarque Il est possible de trouver dans la littérature des formalismes où δ est une fonction partielle, c'est-à-dire que pour tout $p \in Q$ et $a \in V$, il existe *au plus* un état tel que $(p, a, q) \in \delta$. Dans toute la suite, sauf mention contraire, nous supposons toujours que δ est une fonction totale.

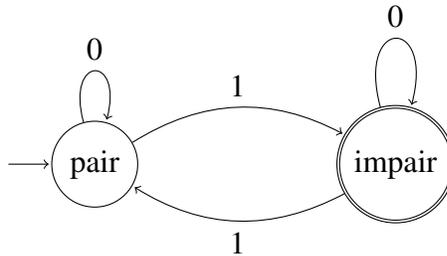
Définition 2.4.2 (extension de la fonction de transition) Soit $\delta : Q \times V \mapsto Q$ la fonction de transition d'un automate déterministe.

On définit par induction la fonction $\delta^* : Q \times V^* \mapsto Q$ de la façon suivante :

- $\delta^*(p, \varepsilon) = p$ pour tout $p \in Q$,
- $\delta^*(p, ax) = \delta^*(\delta(p, a), x)$ pour tout $p \in Q, a \in V$ et $x \in V^*$. \diamond

Remarque Remarquons que pour p un état et a un symbole, $\delta^*(p, a) = \delta^*(\delta(p, a), \varepsilon) = \delta(p, a)$. Donc la fonction δ^* est une extension de δ et dans la suite nous pourrions nous permettre de ne pas distinguer ces deux fonctions et de noter δ à la place de δ^* .

Exemple 2.4.3 L'automate A dessiné ci-dessous est un automate déterministe sur le vocabulaire $\{0, 1\}$: de chaque état part une et une seule 0-transition ainsi qu'une et une seule 1-transition.



$L(A)$ est l'ensemble de chaînes ayant un nombre impair de 1. Notez le choix du nom des états : toute trace d'un chemin de l'état pair à l'état pair comprend un nombre pair de 1, toute trace d'un chemin de l'état pair à l'état impair comprend un nombre impair de 1. \clubsuit

Propriété 2.4.4 Soit $\delta : Q \times V \mapsto Q$ la fonction de transition d'un automate déterministe. Soient p, q deux états et $x \in V^*$. Alors $\delta(p, x) = q$ si et seulement s'il existe un chemin de trace x menant de p à q .

La preuve est immédiate par induction sur la longueur de x .

Propriété 2.4.5 Soit $A = \langle Q, V, \delta, i, F \rangle$ un automate déterministe. $L(A) = \{x \in V^* \mid \delta(i, x) \in F\}$.

PREUVE. Soit $x \in V^*$. D'après la propriété 2.4.4, $\delta(i, x) \in F$ si et seulement s'il existe un chemin de trace x entre l'état initial et un état final de A . D'après la définition 2.2.4, cela signifie que x est bien élément de $L(A)$. ■

Propriété 2.4.6 Soit $\delta : Q \times V \mapsto Q$ la fonction de transition d'un automate déterministe. Pour tout état $p \in Q$ et toutes chaînes $x, y \in V^*$, on a : $\delta(p, xy) = \delta(\delta(p, x), y)$.

PREUVE. La propriété est démontrée par induction sur la longueur de x .

Pour $x = \varepsilon$, d'après la définition 2.4.2, $\delta(\delta(p, x), y) = \delta(p, y)$. Puisque $y = xy$, on a bien : $\delta(p, xy) = \delta(\delta(p, x), y)$.

Supposons l'égalité vérifiée pour x et montrons qu'elle reste vérifiée pour la chaîne ax où $a \in V$:

$$\delta(p, axy) = \delta(\delta(p, a), xy) \quad (\text{Définition 2.4.2})$$

$$\delta(\delta(p, a), xy) = \delta(\delta(\delta(p, a), x), y) \quad (\text{Hypothèse d'induction})$$

$$\delta(\delta(\delta(p, a), x), y) = \delta(\delta(p, ax), y) \quad (\text{Définition 2.4.2})$$

Donc $\delta(p, axy) = \delta(\delta(p, ax), y)$ ■

Définition 2.4.7 (État accessible, automate connecté) Soit $A = \langle Q, V, \delta, i, F \rangle$ un automate fini déterministe. L'état p est *accessible* si et seulement s'il existe une chaîne $x \in V^*$ telle que $p = \delta(i, x)$. Un automate dont tous les états sont accessibles est dit *connecté*. ◇

Exercice 2.4.8 Écrire un algorithme calculant les états accessibles d'un automate fini déterministe.

Théorème 2.4.9 (Élimination des états inaccessibles) Tout automate fini déterministe est équivalent à un automate fini déterministe connecté.

PREUVE. Soit $A = \langle Q, V, \delta, i, F \rangle$ un automate fini déterministe et R l'ensemble de ses états accessibles. Nous prouvons que l'automate B défini par $B = \langle R, V, \eta, i, R \cap F \rangle$, où $\eta = \delta \cap (R \times V \times R)$, est un automate fini déterministe connecté équivalent à A .

L'automate B est déterministe. Il s'agit de prouver que pour tout état $p \in R$ et pour tout symbole $a \in V$, il existe un et un seul état q tel que $(p, a, q) \in \eta$. Puisque l'automate A est déterministe, pour tout état $p \in R$ et tout symbole $a \in V$, il y a *au plus* un état $q \in R$ tel que $(p, a, q) \in \eta$. Montrons qu'il existe au moins un tel état. Soit $p \in R$ et considérons un symbole $a \in V$. Puisque p est accessible, il existe une chaîne x telle que $\delta(i, x) = p$, et par la propriété 2.4.6, on a $\delta(p, a) = \delta(\delta(i, x), a) = \delta(i, xa)$. Donc l'état $\delta(p, a)$ est également accessible et on a bien $(p, a, \delta(p, a)) \in \eta$.

L'automate B est connecté. D'après la preuve ci-dessus, δ et η sont identiques sur le domaine $R \times V$, donc par une induction triviale, elles sont aussi identiques sur le domaine $R \times V^*$.

Soit $p \in R$, puisque p est accessible dans A , il existe une chaîne x telle $\delta(i, x) = p$. Puisque δ et η sont identiques sur le domaine $R \times V^*$, $\eta(i, x) = p$, donc p est accessible dans B .
Donc tous les états de B sont accessibles : B est connecté.

Les automates A et B sont équivalents.

$x \in L(A)$

si et seulement si $\delta(i, x) \in F$ Propriété 2.4.5

si et seulement si $\delta(i, x) \in R \cap F$ Car l'état $\delta(i, x)$ est accessible, donc élément de R

si et seulement si $\eta(i, x) \in R \cap F$ Car δ et η sont identiques sur le domaine $R \times V^*$

si et seulement si $x \in L(B)$ Propriété 2.4.5

D'où le résultat. ■

Exercice 2.4.10 *Écrire un algorithme calculant un automate déterministe connecté équivalent à un automate déterministe donné.*

5 Construction des automates déterministes

Dans cette partie, nous allons démontrer le théorème suivant :

Théorème 2.5.1 *Tout automate fini sans ε -transition est équivalent à un automate déterministe.*

PREUVE. Nous démontrons le résultat en considérant un automate quelconque sans ε -transitions, et en construisant un automate déterministe qui lui est équivalent.

Soit $A = \langle Q, V, \delta, I, F \rangle$ un automate fini sans ε -transitions. Soit $B = \langle \mathcal{P}(Q), V, \eta, \{I\}, G \rangle$ l'automate dont les états sont des ensembles d'états de A , qui est défini de la façon suivante :

- $\eta(P, a) = \bigcup_{p \in P} \delta(p, a)$ pour $P \subset Q$ et $a \in V$,
- $G = \{P \mid P \subset Q \text{ et } P \cap F \neq \emptyset\}$.

Il s'agit de prouver que A et B sont équivalents. Pour cela, nous considérons la fonction $f : \mathcal{P}(Q) \times V^* \mapsto \mathcal{P}(Q)$ qui à tout ensemble d'états $P \subset Q$ associe l'ensemble des états $q \in Q$ tels qu'il existe un chemin de trace x entre un élément de P et q . Nous allons montrer tout d'abord que f et η sont identiques, puis que A et B sont équivalents.

Les fonctions f et η sont identiques. Considérons un ensemble d'états $P \subset Q$, un symbole $a \in V$ et un mot $x \in V^*$. Prouvons d'abord que $f(P, ax) = f(\eta(P, a), x)$.

$$q \in f(P, ax)$$

\Leftrightarrow il existe $p \in P, r \in Q$, un chemin de A menant de p à r de trace a et un chemin de A menant de r à q de trace x (Définition de f)

\Leftrightarrow il existe $p \in P, r \in Q, (p, a, r) \in \delta$ et un chemin de A menant de r à q de trace x (A n'a pas d' ε -transition)

\Leftrightarrow il existe $r \in \eta(P, a)$ et un chemin de A menant de r à q de trace x (Définition de η)

$\Leftrightarrow q \in f(\eta(P, a), x)$ (Définition de f)

On montre maintenant que pour tout $P \subset Q$ et $x \in V^*$, on a $f(P, x) = \eta(P, x)$; le résultat est démontré par induction sur x .

Pour $x = \varepsilon$, l'égalité est vérifiée, car en absence d' ε -transition, $f(P, \varepsilon) = P$ et par définition des extensions $\eta(P, \varepsilon) = P$. Supposons maintenant la propriété vérifiée pour la chaîne x . Montrons qu'elle est aussi vérifiée par la chaîne ax où $a \in V$.

$$f(P, ax) = f(\eta(P, a), x) \quad (\text{D'après le résultat ci-dessus})$$

$$f(\eta(P, a), x) = \eta(\eta(P, a), x) \quad (\text{Hypothèse d' induction})$$

$$\eta(\eta(P, a), x) = \eta(P, ax) \quad (\text{Définition 2.4.2})$$

Equivalence de A et B Soit $x \in V^*$.

$$x \in L(A)$$

si et seulement si il existe un chemin de trace x entre un état initial et un état final de A (Définition 2.2.4)

si et seulement si $f(I, x) \cap F \neq \emptyset$ (Par définition de f)

si et seulement si $f(I, x) \in G$ (Définition de G)

si et seulement si $\eta(I, x) \in G$ (Car f et η sont égales)

si et seulement si $x \in L(B)$ (Car G est l'ensemble des états finals de B)

D'où le résultat. ■

Exemple 2.5.2 Soit $A = \langle Q, V, \delta, I, F \rangle$ l'automate fini décrit dans la Figure 2.3, où $V = \{0, 1\}$. Cet automate reconnaît l'ensemble des chaînes sur $\{0, 1\}$ qui se terminent par 00 . Soit $B = \langle \mathcal{P}(Q), V, \eta, I, G \rangle$ l'automate déterministe construit comme indiqué dans le théorème 2.5.1. Nous représentons B par une table (voir la Figure 2.4). La troisième colonne représente l'ensemble des états de B (c'est-à-dire les sous-ensembles d'états de A). Les colonnes 4 et 5 représentent la fonction de transition η , la colonne 1 permet d'identifier les états accessibles, la colonne 2 le statut de ces états, et la colonne 6 permet de renommer les états de B .

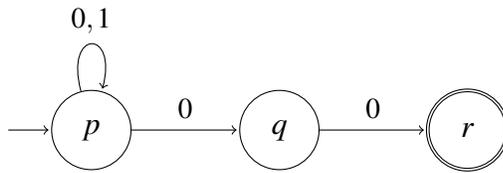


FIGURE 2.3 – Automate de l’Exemple 2.5.2

accessible	statut	état	0	1	renommage
		\emptyset	\emptyset	\emptyset	
(a)	initial	$\{p\}$	$\{p, q\}$	$\{p\}$	r_0
		$\{q\}$	$\{r\}$	\emptyset	
	final	$\{r\}$	\emptyset	\emptyset	
(b)		$\{p, q\}$	$\{p, q, r\}$	$\{p\}$	r_1
	final	$\{p, r\}$	$\{p, q\}$	$\{p\}$	
	final	$\{q, r\}$	$\{r\}$	\emptyset	
(c)	final	$\{p, q, r\}$	$\{p, q, r\}$	$\{p\}$	r_2

FIGURE 2.4 – Table représentant l’automate B de l’Exemple 2.5.2

Après suppression des états inaccessibles (voir la Propriété 2.4.9), l’automate B est équivalent à l’automate déterministe de la Figure 2.5, correspondant aux trois lignes (a), (b) et (c).

Remarque L’automate déterministe dont la construction est décrite dans la preuve du théorème 2.5.1, comporte, en général, beaucoup d’états inaccessibles. En pratique, plutôt que d’introduire ces états puis de les éliminer, on construit la fonction de transition de B en partant de son état initial et en ajoutant uniquement les états accessibles depuis cet état. Ainsi, dans l’exemple 2.5.2, seuls les états (a), (b) et (c) sont accessibles; on construit donc l’automate déterministe équivalent à A en partant de son état initial $\{p\}$ puis en ajoutant les nouveaux états au fur et à mesure qu’ils sont accessibles; on ajoute donc les lignes (b) et (c) sans construire les cinq lignes inutiles des états inaccessibles.

Corollaire 2.5.3 *Tout automate fini est équivalent à un automate fini déterministe connecté.*

PREUVE. D’après le théorème 2.3.1, un automate fini est équivalent à un automate fini sans ε -transition. D’après le théorème 2.5.1, un automate fini sans ε -transition est équivalent à un automate déterministe. D’après le théorème 2.4.9 un automate fini déterministe est équivalent à un automate fini déterministe connecté. Ceci termine la preuve. ■

Exercice 2.5.4 *Montrez que si un langage est reconnu par un automate fini, alors son complémentaire l’est également.*

Indication : utiliser le fait qu’un automate fini est équivalent à un automate déterministe.

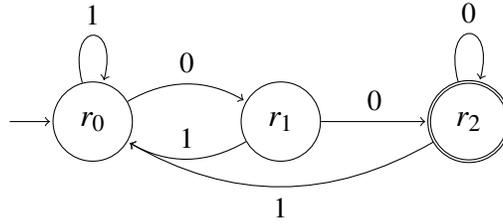


FIGURE 2.5 – Automate déterministe équivalent (Exemple 2.5.2)

6 Machines de Mealy et de Moore

Les machines étudiées dans cette section sont des automates finis déterministes munis de sorties. Ces machines sont très importantes car elles modélisent le fonctionnement des processeurs, qui changent d'états en fonction des signaux d'entrée et produisent des signaux en sortie. Avec des sorties associées aux transitions, on obtient *les machines de Mealy*. Lorsque les sorties sont associées aux états, on obtient *les machines de Moore*.

Les automates finis déterministes peuvent être vus comme des cas particuliers de ces machines.

Définition 2.6.1 (Machine de Mealy) Une machine de Mealy est un sextuplet $A = \langle Q, V, W, i, \delta, s \rangle$ où

- Q est un ensemble fini d'états
- V est le vocabulaire d'entrée
- W est le vocabulaire de sortie
- $i \in Q$ est l'état initial
- $\delta : Q \times V \mapsto Q$ est la fonction de transition
- $s : Q \times V \mapsto W$ est la fonction de sortie

La fonction δ est étendue en une fonction de $Q \times V^*$ dans Q comme dans le cas des automates déterministes, on rappelle que :

$$\begin{aligned} \delta(p, \varepsilon) &= p && \text{pour } p \in Q \\ \delta(p, ax) &= \delta(\delta(p, a), x) && \text{pour } p \in Q, a \in V, x \in V^* \end{aligned}$$

La fonction s est étendue en une fonction de $Q \times V^*$ dans W^* avec :

$$\begin{aligned} s(p, \varepsilon) &= \varepsilon && \text{pour } p \in Q \\ s(p, ax) &= s(p, a)s(\delta(p, a), x) && \text{pour } p \in Q, a \in V, x \in V^* \end{aligned}$$

Soit $s_A : V^* \mapsto W^*$ la fonction définie par $s_A(x) = s(i, x)$.

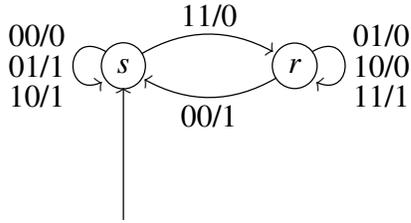
Cette fonction est la fonction (d'entrée-sortie) calculée par A .

Deux machines de Mealy (ayant même vocabulaire d'entrée et même vocabulaire de sortie) sont équivalentes si elles calculent la même fonction d'entrée-sortie. \diamond

L'état initial d'une telle machine est noté comme celui d'un automate fini. La fonction de transition et la fonction d'entrée-sortie sont notées avec les conventions suivantes :

$$\begin{array}{c} (p) \xrightarrow{a/b} (q) \end{array} \quad \text{signifie que } \delta(p, a) = q \text{ et } s(p, a) = b$$

Exemple 2.6.2 On trouve un premier exemple de machine de Mealy en 2.1.3 avec un additionneur binaire. Nous le redonnons ici :



La machine a deux états s (pas de retenue) et r (retenue). Son vocabulaire d'entrée est $V = \{00, 01, 10, 11\}$, son vocabulaire de sortie est $W = \{0, 1\}$.

Pour ajouter 110 (6 en décimal) et 110, on donne le mot d'entrée 00.11.11 dans lequel les unités sont mises à gauche et le point est utilisé pour séparer les symboles d'entrée.

Appelons A cette machine, avec les mêmes notations que dans la définition ci-dessus. On a : $s_A(00.11.11) = 001$ et $\delta(s, 00.11.11) = r$. Autrement dit la somme vaut 1100 (12 en décimal), le 1 en tête venant de la retenue et 100 venant de la sortie retournée. ♣

Propriété 2.6.3 Soit $\langle Q, V, W, i, \delta, s \rangle$ une machine de Mealy. Pour tous $p \in Q, x, y \in V^*$, on a : $s(p, xy) = s(p, x)s(\delta(p, x), y)$

Exercice 2.6.4 Prouver la propriété ci-dessus.

Définition 2.6.5 (Machine de Moore) Une machine de Moore est un sextuplet $A = \langle Q, V, W, i, \delta, s \rangle$ où

- Q est un ensemble fini d'états
- V est le vocabulaire d'entrée
- W est le vocabulaire de sortie
- $i \in Q$ est l'état initial
- $\delta : Q \times V \mapsto Q$ est la fonction de transition
- $s : Q \mapsto W$ est la fonction de sortie

La fonction s est étendue en une fonction de $Q \times V^*$ dans W^* avec :

$$\begin{aligned} s(p, \varepsilon) &= s(p) && \text{pour } p \in Q \\ s(p, ax) &= s(p)s(\delta(p, a), x) && \text{pour } p \in Q, a \in V, x \in V^* \end{aligned}$$

La fonction d'entrée-sortie calculée par A est définie comme pour la machine de Mealy, en démarrant la machine à partir de son état initial. L'équivalence entre deux machines de Moore est définie comme pour les machines de Mealy. ◇

Exercice 2.6.6 Soit $A = \langle Q, V, i, \delta, F \rangle$ un automate fini déterministe. Définir une machine de Moore $B = \langle Q, V, \{0, 1\}, i, \delta, s \rangle$ telle que pour toute chaîne $x \in V^*$, x est reconnu par A si et seulement si $s_B(x)$ est une chaîne qui se termine par 1.

Cet exercice montre que les automates finis déterministes peuvent être vues comme des machines de Moore particulières.

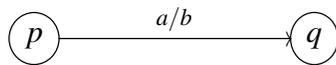
Nous montrons aussi comment passer d'une machine de Moore à une machine de Mealy et inversement.

Théorème 2.6.7 (De Moore à Mealy) Soit $A = \langle Q, V, W, i, \delta, s \rangle$ une machine de Moore.
 Soit $B = \langle Q, V, W, i, \delta, t \rangle$ la machine de Mealy telle que : pour tout $p \in Q, a \in V, t(p, a) = s(\delta(p, a))$.
 Pour tout $p \in Q, x \in V^*$, nous avons : $s(p, x) = s(p)t(p, x)$.

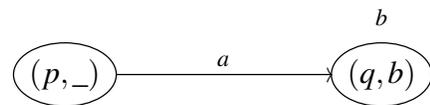
La preuve est laissée en exercice.

Théorème 2.6.8 (De Mealy à Moore) Soit $A = \langle Q, V, W, i, \delta, s \rangle$ une machine de Mealy. La transformation en une machine de Moore est résumé par le schéma suivant :

Machine de Mealy



Machine de Moore



Autrement dit, le symbole b , que la machine de Mealy doit écrire en lisant a dans l'état p , est mémorisé dans l'état (q, b) de la machine de Moore qui écrit b dans cet état. Formalisons cette construction.

Soit $B = \langle Q \times W, V, W, (i, a), \eta, t \rangle$ la machine de Moore suivante :

- l'état initial est (i, a) où $a \in W$ est un symbole quelconque
- $\eta((p, a), b) = (\delta(p, b), s(p, b))$ pour $p \in Q, a, b \in V$: la transition de la machine de Mealy est effectuée et le symbole qu'elle devait écrire lors de cette transition est mémorisée.
- $t((p, a)) = a$ pour $p \in Q, a \in W$

Pour tout $p \in Q, a \in V, x \in V^*$, nous avons : $t((p, a), x) = as(p, x)$

La preuve est laissée en exercice. On note que cette transformation multiplie le nombre d'états de la machine de Mealy par le nombre de symboles du vocabulaire de sortie.

Chapitre 3

Automates finis déterministes minimaux

Ce chapitre est consacré à la minimisation des automates finis déterministes. Formellement, un automate déterministe est *minimal* si tout automate déterministe qui lui est équivalent comporte au moins autant d'états.

Ce chapitre est divisé en deux parties :

- Construction d'un automate minimal équivalent à un automate déterministe donné,
- Démonstration de l'unicité (au nom près des états) de l'automate minimal.

Ces résultats s'étendent aux machines de Mealy et de Moore (définies au paragraphe 6 du Chapitre 2) et nous laissons cette étude en exercice.

Comme dans ce chapitre tous les automates sont finis et déterministes, nous nous abstenons, sauf exception, de le rappeler.

1 Construction de l'automate minimal

Définition 3.1.1 (Équivalence entre les états) Soit $A = \langle Q, V, \delta, i, F \rangle$ un automate. Soit p un état de cet automate. On note alors A_p l'automate $\langle Q, V, \delta, p, F \rangle$ (autrement dit, A et A_p diffèrent uniquement par leur état initial). Deux états p et q sont *équivalents* si les deux automates A_p et A_q sont équivalents. On note \equiv_A la relation d'équivalence associée à l'automate A , l'indice A pouvant être omis quand il n'y a pas d'ambiguïté. \diamond

Propriété 3.1.2 Soit $A = \langle Q, V, \delta, i, F \rangle$ un automate.

1. Les états p et q sont équivalents si et seulement si quelle que soit la chaîne $x \in V^*$, on a $(\delta(p, x) \in F \Leftrightarrow \delta(q, x) \in F)$.
2. Si les états p et q sont équivalents, alors pour toute chaîne $x \in V^*$, les états $\delta(p, x)$ et $\delta(q, x)$ sont équivalents.

PREUVE. 1. Par définition, les états p et q sont équivalents si et seulement si $L(A_p) = L(A_q)$. D'après la propriété 2.4.5, $L(A_p) = \{x \in V^* \mid \delta(p, x) \in F\}$ et $L(A_q) = \{x \in V^* \mid \delta(q, x) \in F\}$, donc p et q sont équivalents si et seulement si pour toute chaîne $x \in V^*$, $(\delta(p, x) \in F \Leftrightarrow \delta(q, x) \in F)$.

2. Supposons que les états p et q sont équivalents. Pour montrer que pour toute chaîne $x \in V^*$, les états $\delta(p, x)$ et $\delta(q, x)$ sont équivalents, il suffit, d'après la propriété ci-dessus, de prouver que pour toute chaîne $y \in V^*$, $\delta(\delta(p, x), y) \in F$ si et seulement si $\delta(\delta(q, x), y) \in F$.

$$\begin{aligned} \delta(\delta(p, x), y) \in F &\Leftrightarrow \delta(p, xy) \in F && \text{(Propriété 2.4.6 du chapitre 2)} \\ &\Leftrightarrow \delta(q, xy) \in F && \text{(Propriété ci-dessus)} \\ &\Leftrightarrow \delta(\delta(q, x), y) \in F && \text{(Propriété 2.4.6 du chapitre 2)} \quad \blacksquare \end{aligned}$$

Minimisation

Pour minimiser un automate A , on procède en deux étapes :

- on enlève les états inaccessibles depuis l'état initial (voir le théorème 2.4.9);
- on *identifie* les états équivalents.

Notation 3.1.3 Soit ρ une relation d'équivalence sur un ensemble A et $a \in A$. La classe d'équivalence de a pour la relation ρ est notée $[a]_\rho$, l'indice pouvant être omis quand le contexte le permet. \diamond

Définition 3.1.4 (Identification des états équivalents) Soit $A = \langle Q, V, \delta, i, F \rangle$ un automate et \equiv_A l'équivalence entre états associée à l'automate. On associe à A l'automate $m(A) = \langle R, V, \eta, [i]_{\equiv_A}, R \rangle$ où :

- R est l'ensemble des classes d'équivalence de la relation \equiv_A sur Q ,
- pour tout $p \in Q$, $a \in V$, $\eta([p]_{\equiv_A}, a) = [\delta(p, a)]_{\equiv_A}$,
- G est l'ensemble des classes d'équivalence des états de F . \diamond

Notons que l'application η est bien définie, car, d'après la Propriété 3.1.2 (2), si les états p et q sont équivalents, il en est de même des états $\delta(p, a)$ et $\delta(q, a)$, pour tout $a \in V$.

Propriété 3.1.5 (Équivalence de A et $m(A)$) On reprend les notations ci-dessus sauf que l'on note la relation d'équivalence $[.]$ au lieu de $[.]_A$. L'automate $m(A)$ vérifie les propriétés suivantes :

1. pour tout $p \in Q$, $x \in V^*$, $\eta([p], x) = [\delta(p, x)]$;
2. pour tout $p \in Q$, $p \in F$ si et seulement si $[p] \in G$;
3. les automates A et $m(A)$ sont équivalents;
4. deux états équivalents de $m(A)$, pour l'équivalence associée à $m(A)$, sont identiques;
5. si A est connecté alors $m(A)$ est connecté.

PREUVE. 1. Prouvons que $\eta([p], x) = [\delta(p, x)]$ par induction sur x . Pour $x = \varepsilon$, $\eta([p], x) = [p] = [\delta(p, x)]$ par définition des extensions de η et δ . Supposons maintenant que pour tout $p \in Q$, $\eta([p], x) = [\delta(p, x)]$ et soit $a \in V$. Montrons qu'on a bien $\eta([p], ax) =$

$$\begin{aligned}
& [\delta(p, ax)]. \\
\eta([p], ax) &= \eta(\eta([p], a), x) \quad \text{par définition de l'extension de } \eta \\
&= \eta([\delta(p, a)], x) \quad \text{par définition de } \eta \\
&= [\delta(\delta(p, a), x)] \quad \text{par hypothèse d'induction} \\
&= [\delta(p, ax)] \quad \text{par définition de l'extension de } \delta.
\end{aligned}$$

2. Montrons que $p \in F$ si et seulement si $[p] \in G$. Par définition de G , si $p \in F$ alors $[p] \in G$. Pour démontrer la réciproque, supposons que $[p] \in G$. Alors par définition de G , il existe un état q équivalent à p , tel que $q \in F$. En particulier, $\delta(q, \varepsilon) = q \in F$, et comme p est équivalent à q , $\delta(p, \varepsilon) = p$ est également dans F d'après la propriété 3.1.2, d'où le résultat.
3. les automates A et $m(A)$ sont équivalents
 - $x \in L(A)$
 - si et seulement si $\delta(i, x) \in F$ Par définition de A
 - si et seulement si $[\delta(i, x)] \in G$ Par la propriété ci-dessus
 - si et seulement si $\eta([i], x) \in G$ Par le premier point de cette propriété
 - si et seulement si $x \in L(m(A))$ Par définition de $m(A)$
4. Montrons que deux états équivalents de $m(A)$, pour l'équivalence associée à $m(A)$, sont identiques. Soient p et q deux états de A .
 - $[p]$ et $[q]$ sont équivalents pour $m(A)$
 - $\Leftrightarrow \forall x \in V^*, \eta([p], x) \in G$ ssi $\eta([q], x) \in G$ (propriété 3.1.2 (1))
 - $\Leftrightarrow \forall x \in V^*, [\delta(p, x)] \in G$ ssi $[\delta(q, x)] \in G$ (premier point de cette propriété)
 - $\Leftrightarrow \forall x \in V^*, \delta(p, x) \in F$ ssi $\delta(q, x) \in F$ (second point de cette propriété)
 - $\Leftrightarrow \forall x \in V^*, [p] = [q]$ (propriété 3.1.2 (1))
5. Supposons A connecté, montrons qu'il en est de même de $m(A)$.
Soit $[p]$ où $p \in Q$, un état *quelconque* de $m(A)$
Puisque A est connecté, il existe $x \in V^*$ tel que $\delta(i, x) = p$
D'après la propriété 1, $\eta([i], x) = [p]$, donc $[p]$ est accessible. ■

Remarque Supposons que A est connecté. La propriété ci-dessus montre non seulement que les automates A et $m(A)$ sont équivalents mais aussi, qu'il est impossible, par la même construction appliquée à $m(A)$, de réduire le nombre d'états de cet automate.

2 Relation d'équivalence entre les états d'un automate

Soit A un automate. La construction de l'automate minimal équivalent à A décrite au paragraphe précédent s'appuie sur l'identification des états de A qui sont équivalents. Pour achever cette construction, nous montrons comment calculer la relation d'équivalence entre les états de A . Cette relation est déterminée à l'aide d'une suite d'approximations.

Définition 3.2.1 (Approximations de l'équivalence entre états) Soit $A = \langle Q, V, \delta, i, F \rangle$ un automate. Pour tout $k \in \mathbb{N}$, on définit la relation \equiv_k de la façon suivante : pour tous $p, q \in Q$, $p \equiv_k q$ si et seulement si pour toute chaîne $x \in V^*$ de longueur *au plus* k , $(\delta(p, x) \in F \Leftrightarrow \delta(q, x) \in F)$. \diamond

Exercice 3.2.2 Montrer que la relation \equiv_k est bien une relation d'équivalence.

Propriété 3.2.3 (Relation entre les approximations de l'équivalence \equiv) Considérons un automate $A = \langle Q, V, \delta, i, F \rangle$.

1. $\equiv = \bigcap_{k \in \mathbb{N}} \equiv_k$.
2. Pour tout $k \in \mathbb{N}$, $\equiv_{k+1} \subset \equiv_k$
3. Pour tous $p, q \in Q$, $p \equiv_0 q$ si et seulement si ($p, q \in F$ ou bien $p, q \notin F$)
4. Pour tout $k \in \mathbb{N}$, $p, q \in Q$, $p \equiv_{k+1} q$ si et seulement si ($p \equiv_k q$ et pour tout $a \in V$, $\delta(p, a) \equiv_k \delta(q, a)$)
5. Soit n le nombre d'états de l'automate A , il existe un entier k tel que $k \leq n$ et $\equiv_k = \equiv_{k+1} = \equiv$

PREUVE. Les trois premières propriétés sont évidentes et nous montrons seulement les deux dernières.

— Prouvons que $p \equiv_{k+1} q$ si et seulement si ($p \equiv_k q$ et pour tout $a \in V$, $\delta(p, a) \equiv_k \delta(q, a)$).

1. Prouvons l'implication de gauche à droite. Soient p et q deux états tels que $p \equiv_{k+1} q$. Puisque $\equiv_{k+1} \subset \equiv_k$, on a $p \equiv_k q$. Par définition de la relation \equiv_{k+1} , on a : pour tout $a \in V$, pour toute chaîne $x \in V^*$ de longueur au plus k , $\delta(p, ax) \in F$ si et seulement si $\delta(q, ax) \in F$. Ce qui s'écrit encore : pour tout $a \in V$, pour toute chaîne $x \in V^*$ de longueur au plus k , $\delta(\delta(p, a), x) \in F$ si et seulement si $\delta(\delta(q, a), x) \in F$. Donc par définition de \equiv_k , pour tout $a \in V$, $\delta(p, a) \equiv_k \delta(q, a)$.
2. Prouvons l'implication de droite à gauche. Soient p et q deux états tels que $p \equiv_k q$ et pour tout $a \in V$, $\delta(p, a) \equiv_k \delta(q, a)$. De la deuxième hypothèse on déduit : pour tout $a \in V$, pour toute chaîne $x \in V^*$ de longueur k , $\delta(\delta(p, a), x) \in F$ équivaut à $\delta(\delta(q, a), x) \in F$. Cette dernière propriété se réécrit : Pour toute chaîne $x \in V^*$ de longueur $k+1$, $\delta(p, x) \in F$ équivaut à $\delta(q, x) \in F$. Puisque $p \equiv_k q$, il en résulte que $p \equiv_{k+1} q$.

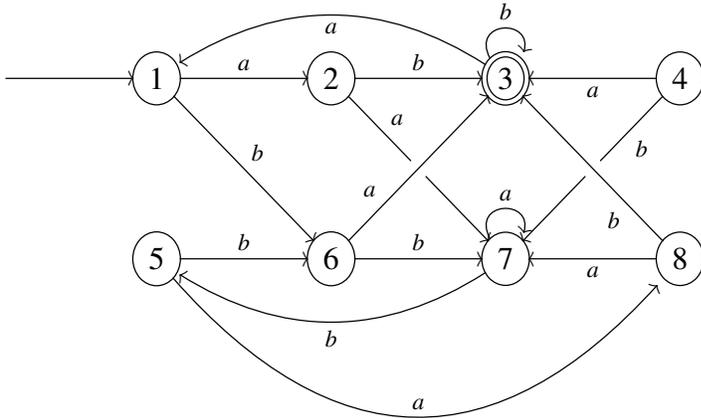
— Soit n le nombre d'états de l'automate A , montrons qu'il existe un entier k tel que $k \leq n$ et $\equiv_k = \equiv_{k+1} = \equiv$.

1. Il existe un entier $k \leq n$ tel que $\equiv_k = \equiv_{k+1}$.
En effet les partitions associées aux relations d'équivalences \equiv_k sont de plus en plus fines lorsque k augmente. Puisque toute partition des états de A comprend au plus n classes, il existe un entier $k \leq n$ tel que les partitions associées aux relations \equiv_k et \equiv_{k+1} sont égales, donc tel que ces relations sont égales.
2. Si $\equiv_m = \equiv_{m+1}$ alors $\equiv_m = \equiv$.
Supposons que $\equiv_m = \equiv_{m+1}$. De cette hypothèse et du point 4 de cette propriété, il résulte immédiatement que pour tout $k \geq m$, on a $\equiv_k = \equiv_m$. Puisque la suite des relations \equiv_k est une suite décroissante, stable à partir de l'indice m , il en résulte que :

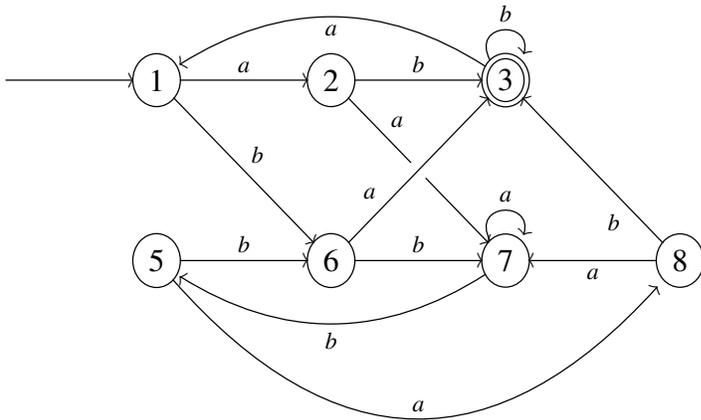
$$\equiv_m = \bigcap_{k \in \mathbb{N}} \equiv_k.$$

Donc d'après le point 1 de cette propriété, $\equiv_m = \equiv$ ■

Exemple 3.2.4 Minimisons l'automate déterministe A de vocabulaire $\{a,b\}$, dessiné ci-dessous :



On remarque que l'état 4 est inaccessible. On le supprime et on obtient l'automate connecté B ci-dessous :



On calcule incrémentalement les classes d'équivalence correspondant aux relations \equiv_k associées à l'automate B . Les classes d'équivalence correspondant à la relation \equiv_0 sont simples : une des classes contient tous les états finals, et l'autre classe contient les autres états. Ensuite, d'après la propriété 4, les deux états p et q sont dans la même classe de la relation \equiv_{k+1} si et seulement si

- ils sont dans la même classe de la relation \equiv_k
- $\delta(p,a)$ et $\delta(q,a)$ sont dans la même classe de la relation \equiv_k
- $\delta(p,b)$ et $\delta(q,b)$ sont dans la même classe de la relation \equiv_k

On obtient donc les classes d'équivalence suivantes :

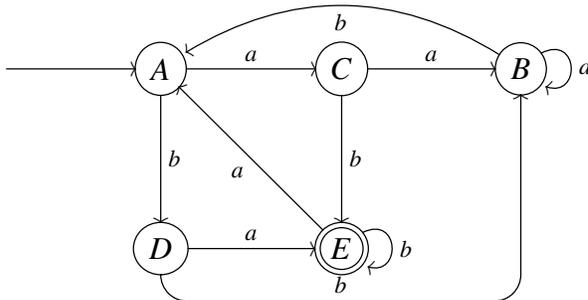
$$\begin{aligned} \equiv_0 & : \{1,2,5,6,7,8\}, \{3\} \\ \equiv_1 & : \{1,5,7\}, \{2,8\}, \{6\}, \{3\} \\ \equiv_2 & : \{1,5\}, \{7\}, \{2,8\}, \{6\}, \{3\} \\ \equiv_3 & : \{1,5\}, \{7\}, \{2,8\}, \{6\}, \{3\} \end{aligned}$$



La détermination de ces classes d'équivalence peut également être résumée par le tableau ci-dessous. Les numéros de la première colonne du tableau sont les indices des relations d'équivalence \equiv_k , et les classes sont désignées par des lettres majuscules.

k	classes						
0	1	2	5	6	7	8	3
	A						B
a	A	A	A	B	A	A	
b	A	B	A	A	A	B	
1	1	5	7	2	8	6	3
	A		B		C	D	
a	B	B	A	A	A		
b	C	C	A	D	D		
2	1	5	7	2	8	6	3
	A		B	C		D	E
a	C	C	B	B	B	E	A
b	D	D	A	E	E	B	E
3	1	5	7	2	8	6	3

Puisque les partitions associées aux relations \equiv_2 et \equiv_3 sont identiques, la relation \equiv est identique à \equiv_2 . Nous pouvons maintenant construire l'automate $m(B)$, qui est dessiné ci-dessous. Pour calculer la partition associée à \equiv_3 , il est inutile de calculer les classes des états $\delta(7,a), \delta(7,b), \delta(6,a), \delta(6,b), \delta(3,a), \delta(3,b)$ puisque les états 7,6,3 sont seuls dans leurs classes. Mais on a fait figurer ce calcul dans la table ci-dessus, car cela fait apparaître la fonction de transition de l'automate minimal dans les trois avant-dernières lignes de ce tableau.



3 Unicité de l'automate minimal

Définition 3.3.1 (Automate réduit) Un automate *réduit* est un automate dont les états sont deux à deux non équivalents. \diamond

Nous avons vu dans la section 1 comment construire un automate connecté et réduit équivalent à un automate A . Nous montrons maintenant les résultats suivants :

- Un automate est minimal si et seulement s'il est connecté et réduit (voir théorème 3.3.5).
Il en résulte que $m(B)$ est un automate minimal équivalent à A .

- L'automate minimal est unique, au nom près de ses états (voir théorème 3.3.6). Il en résulte que $m(B)$ est l'automate minimal équivalent à A .

3.1 Propriétés de la catégorie des automates

Propriété 3.3.2 *Si un automate est minimal, il est connecté et réduit.*

PREUVE. Supposons que l'automate A ne soit pas connecté ou pas réduit. Examinons ces deux cas.

1. Supposons que A ne soit pas connecté. En enlevant ses états inaccessibles, on obtient un automate équivalent avec moins d'états que A . Donc A n'est pas minimal.
2. Supposons que A ne soit pas réduit. L'automate $m(A)$ est un automate équivalent avec moins d'états que A . Donc A n'est pas minimal. ■

Définition 3.3.3 (homomorphisme et isomorphisme d'automates) Soient $A = \langle Q, V, \delta, i, F \rangle$ et $B = \langle R, V, \eta, j, G \rangle$ deux automates. L'application $h : Q \mapsto R$ est un homomorphisme de A dans B lorsqu'elle vérifie :

- $h(i) = j$,
- $h(F) = G$, où $h(F) = \{h(p) \mid p \in F\}$,
- pour tout $p \in Q$ et $a \in V$, $h(\delta(p, a)) = \eta(h(p), a)$.

Un homomorphisme bijectif est un isomorphisme.

Deux automates sont isomorphes, s'il existe un isomorphisme de l'un dans l'autre. Informellement, ces deux automates sont identiques au nom de leurs états près. ◇

Lemme 3.3.4 *Soient $A = \langle Q, V, \delta, i, F \rangle$ et $B = \langle R, V, \eta, j, G \rangle$ deux automates équivalents et connectés. Si B est réduit alors il existe un homomorphisme h de A dans B tel que $h(Q) = R$, où $h(Q) = \{h(p) \mid p \in Q\}$.*

PREUVE. Soit $h \subset Q \times R$ la relation définie par $h = \{(\delta(i, x), \eta(j, x)) \mid x \in V^*\}$.

On montre que h est un homomorphisme de A dans B .

1. Q est le domaine de la relation h .
En effet soit $p \in Q$. Puisque A est connecté, il existe une chaîne $x \in V^*$ telle que $p = \delta(i, x)$. Donc p est dans le domaine de h .
2. h est une fonction de Q dans R telle que pour tout $x \in V^*$, $h(\delta(i, x)) = \eta(j, x)$.
D'après la définition de la relation h et le fait, prouvé ci-dessus, que Q est le domaine de la relation h , il s'agit de vérifier que h est une fonction bien définie, c'est-à-dire que pour toutes chaînes $x, y \in V^*$, si $\delta(i, x) = \delta(i, y)$ alors $\eta(j, x) = \eta(j, y)$. Supposons au contraire qu'il existe deux chaînes $x, y \in V^*$ telles que $\delta(i, x) = \delta(i, y)$ et $\eta(j, x) \neq \eta(j, y)$. Puisque l'automate B est réduit par hypothèse, les deux états $\eta(j, x)$ et $\eta(j, y)$ ne sont pas équivalents, donc il existe $z \in V^*$ tel que l'un des deux états $\eta(\eta(j, x), z)$ et $\eta(\eta(j, y), z)$ est dans G tandis que l'autre n'est pas dans G .

Soit L le langage reconnu par les deux automates équivalents. Puisque G est l'ensemble des états finals de B , la propriété ci-dessus implique que

Il existe $z \in V^$ tel que l'une des deux chaînes xz et yz est dans L tandis que l'autre n'est pas dans L . (\star)*

Mais puisque $\delta(i,x) = \delta(i,y)$, alors pour toute chaîne $z \in V^*$, $\delta(i,xz) = \delta(i,yz)$. Et puisque l'automate A reconnaît L , on en déduit que pour toute chaîne $z \in V^*$, soit les deux chaînes xz et yz sont toutes les deux dans L , soit aucune des deux n'est dans L . Mais ceci contredit (\star), et l'hypothèse d'après laquelle il existe deux chaînes $x, y \in V^*$ telles que $\delta(i,x) = \delta(i,y)$ et $\eta(j,x) \neq \eta(j,y)$ est absurde.

3. $h(Q) = R$.

Puisque h est une application de Q dans R , il suffit de vérifier que tout élément de R est l'image par h d'un élément de Q . Soit $p \in R$. Puisque B est connecté, il existe $x \in V^*$ tel que $p = \eta(j,x)$. D'après le point 2 ci-dessus, $p = h(\delta(i,x))$; donc $p \in h(Q)$.

Nous avons donc prouvé que h est une fonction de Q dans R telle que :

1. pour tout $x \in V^*$, $h(\delta(i,x)) = \eta(j,x)$;
2. $h(Q) = R$.

Nous prouvons maintenant que h est un homomorphisme de A dans B .

1. $h(i) = j$.

Comme pour tout $x \in V^*$, $h(\delta(i,x)) = \eta(j,x)$, en particulier, $h(\delta(i,\epsilon)) = \eta(j,\epsilon)$. Par définition des extensions des fonctions δ et η , $\delta(i,\epsilon) = i$ et $\eta(j,\epsilon) = j$. Donc $h(i) = j$.

2. $h(F) = G$.

On prouve que $h(F) \subset G$, puis que $G \subset h(F)$.

(a) $h(F) \subset G$.

Soit $p \in F$. Puisque A est connecté, il existe $x \in V^*$ tel que $p = \delta(i,x)$, et on a $h(p) = \eta(j,x)$. Puisque $p \in F$, la chaîne x est donc dans le langage reconnu par A . Puisque les automates A et B sont équivalents, x est également reconnue par B , donc $\eta(j,x) \in G$ et par suite $h(p) \in G$.

(b) $G \subset h(F)$.

Soit $p \in G$. Puisque B est connecté, il existe $x \in V^*$ tel que $p = \eta(j,x)$, et on a $p = h(\delta(i,x))$. Puisque $p \in G$, la chaîne x est dans le langage reconnu par B . Puisque les automates A et B sont équivalents, x est également reconnue par A , donc $\delta(i,x) \in F$ et par suite $p \in h(F)$.

3. Pour tout $p \in Q, a \in V$, $h(\delta(p,a)) = \eta(h(p),a)$.

Soit $p \in Q$ et $a \in V$. Puisque A est connecté, il existe $x \in V^*$ tel que $p = \delta(i,x)$. D'après les propriétés des automates déterministes, $h(\delta(p,a)) = h(\delta(i,xa)) = \eta(j,xa)$. Or, $\eta(j,xa) = \eta(\eta(j,x),a) = \eta(h(\delta(i,x)),a)$. Puisque $p = \delta(i,x)$, on a bien $h(\delta(p,a)) = \eta(h(p),a)$. ■

Remarque De ce lemme, il est facile de déduire la propriété “un automate est minimal si et seulement s'il est connecté et réduit”, ainsi que l'unicité de l'automate minimal.

Théorème 3.3.5 (Propriété caractéristique de l'automate minimal) *Un automate est minimal si et seulement s'il est connecté et réduit.*

PREUVE. L'implication de gauche à droite est déjà prouvée par la propriété 3.3.2. Réciproquement, soit $A = \langle Q, V, \delta, i, F \rangle$ un automate connecté et réduit. Montrons qu'il est minimal, c'est-à-dire que tout autre automate équivalent a au moins autant d'états.

Soit $B = \langle R, V, \eta, j, G \rangle$ un automate quelconque équivalent à A et soit $C = \langle S, V, \theta, j, H \rangle$ l'automate obtenu en retirant les états inaccessibles de A . L'automate C est donc équivalent à A et connecté. D'après le lemme 3.3.4, il existe un homomorphisme h de C dans A tel que $h(S) = Q$ ce qui signifie que le nombre d'états de C est supérieur ou égal au nombre d'états de A . Puisque C est obtenu en enlevant les états inaccessibles de B , le nombre d'états de B est supérieur ou égal au nombre d'états de C . Donc tout automate équivalent à A possède au moins autant d'états que A : A est bien minimal. ■

Théorème 3.3.6 (Unicité de l'automate minimal) *Si deux automates sont équivalents et minimaux, ils sont isomorphes.*

PREUVE. Soient $A = \langle Q, V, \delta, i, F \rangle$ et $B = \langle R, V, \eta, j, G \rangle$ deux automates équivalents et minimaux. Puisqu'ils sont minimaux, d'après le Théorème 3.3.5, ils sont connectés et réduits. Donc, d'après le Lemme 3.3.4, il existe un homomorphisme h de A dans B tel que $h(Q) = R$. Puisque les deux automates sont minimaux, les ensembles Q et R ont le même nombre d'éléments et puisque $h(Q) = R$, la fonction h est bijective. Donc A et B sont isomorphes. ■

4 Pour aller plus loin : le théorème de Myhill-Nérode

Ce paragraphe est dédié à une preuve alternative du résultat suivant : soit A un automate connecté. L'automate $m(A)$ de la Définition 3.1.4 est l'unique automate minimal équivalent à A .

Définition 3.4.1 (Dérivé et équivalence entre chaînes) Soit L un langage sur le vocabulaire V .

1. Soit $u \in V^*$. On pose $u^{-1}L = \{v \in V^* \mid uv \in L\}$. Ce langage est la *dérivée de u relativement à L* .
2. Deux chaînes u et v , où $u, v \in V^*$, sont *équivalentes relativement à L* si et seulement elles ont la même dérivée relativement à L . On note \equiv_L cette relation d'équivalence entre chaînes. L'indice L peut être omis quand il n'y a pas d'ambiguïté. ◇

Propriété 3.4.2 (Propriété de l'équivalence entre chaînes) Soit L un langage sur le vocabulaire V . Soient $u, v \in V^*$

1. $u \equiv_L v$ si et seulement si pour toute chaîne $x \in V^*$, $ux \in L$ si et seulement si $vx \in L$.
2. si $u \equiv_L v$ alors pour toute chaîne $x \in V^*$, $ux \equiv_L vx$

La preuve est immédiate, et est laissée en exercice.

Définition 3.4.3 (Automate des classes d'un langage) Soit L un langage sur le vocabulaire V . Soit $A(L) = \langle Q, V, \delta, i, F \rangle$ l'automate déterministe suivant¹ :

1. Q est l'ensemble des classes de la relation \equiv_L
2. Pour tout $x \in V^*, a \in V$, $\delta([x]_{\equiv_L}, a) = [xa]_{\equiv_L}$
3. $i = [\varepsilon]_{\equiv_L}$
4. F est l'ensemble des classes des chaînes du langage L

$A(L)$ est appelé *l'automate des classes du langage L* . ◇

Notons que l'application δ est bien définie, car d'après la propriété 3.4.2, pour tout $x, y \in V^*, a \in V$, si $x \equiv_L y$ alors $xa \equiv_L ya$.

Propriété 3.4.4 Soit $A(L)$ l'automate des classes du langage L . On reprend les notations ci-dessus.

1. Pour tous $x, y \in V^*$, $\delta([x]_{\equiv_L}, y) = [xy]_{\equiv_L}$
2. L'automate $A(L)$ reconnaît L .

PREUVE. 1. La première partie de la propriété se prouve par une induction immédiate sur y et est laissée en exercice.

2. Montrons la deuxième propriété :
 x est reconnu par $A(L)$
 si et seulement si $\delta([\varepsilon]_{\equiv_L}, x) \in F$
 si et seulement si $[x]_{\equiv_L} \in F$
 si et seulement si $x \in L$

d'après la première partie de la propriété
 car F est l'ensemble des classes des
 chaînes du langage L

D'où le résultat. ■

Théorème 3.4.5 (Théorème de Myhill-Nérode) Le langage L est régulier si et seulement si le nombre de classes de \equiv_L est fini.

PREUVE. 1. Supposons L régulier et montrons que le nombre de classes de \equiv_L est fini. Puisque L est régulier, il existe un automate déterministe fini $A = \langle Q, V, \delta, i, F \rangle$ qui reconnaît L . Soit h l'application définie de la façon suivante :

- son domaine est l'ensemble des classes d'équivalence de la relation \equiv_L (voir la Définition 3.4.1),
- sa portée est l'ensemble des classes d'équivalence de la relation \equiv_A , (voir la Définition 3.1.1),
- pour tout $x \in V^*$, $h([x]_{\equiv_L}) = [\delta(i, x)]_{\equiv_A}$

1. Notons qu'en toute rigueur, le nombre des états de l'automate n'est pas nécessairement fini.

(a) Montrons que $x \equiv_L y$ si et seulement si $\delta(i, x) \equiv_A \delta(i, y)$.

$$\begin{aligned} x \equiv_L y &\Leftrightarrow (\forall z \in V^*, xz \in L \text{ ssi } yz \in L) && \text{(Propriété 3.4.2)} \\ &\Leftrightarrow (\forall z \in V^*, \delta(i, xz) \in F \text{ ssi } \delta(i, yz) \in F) && \text{(Car } A \text{ reconnaît } L) \\ &\Leftrightarrow (\forall z \in V^*, \delta(\delta(i, x), z) \in F \text{ ssi } \delta(\delta(i, y), z) \in F) && \text{(Prop. 2.4.6, chapitre 2)} \\ &\Leftrightarrow \delta(i, x) \equiv_A \delta(i, y) && \text{(Par définition de } \equiv_A) \end{aligned}$$

(b) h est une application entre classes.

D'après ce qui précède, si $x \equiv_L y$ alors $\delta(i, x) \equiv_A \delta(i, y)$.

(c) h est injective.

Supposons $[\delta(i, x)]_{\equiv_A} = [\delta(i, y)]_{\equiv_A}$. D'après ce qui précède, $x \equiv_L y$, donc $[x]_{\equiv_L} = [y]_{\equiv_L}$.

Puisque h est injective, le nombre de classes de \equiv_L est au plus égal au nombre de classes de \equiv_A . Il est évident que le nombre de classes de \equiv_A est au plus égal au nombre d'états de A . Puisque A est un automate *fini*, le nombre de classes de \equiv_L est donc fini.

- Supposons que le nombre de classes de \equiv_L est fini. L'automate $A(L)$ de la Définition 3.4.3, est alors un automate déterministe *fini*, qui, d'après la propriété 3.4.4, reconnaît L , par suite L est un langage régulier. ■

Théorème 3.4.6 (L'automate minimal est unique) *Soit L un langage régulier sur le vocabulaire V . L'automate $A(L)$ des classes de L est minimal et tout automate minimal équivalent lui est isomorphe.*

PREUVE. 1. $A(L)$ est minimal.

Soit A un automate quelconque reconnaissant L . Comme nous l'avons vu dans la démonstration du théorème de Myhill-Nérode, le nombre d'états de $A(L)$ est au plus égal à celui de A , donc $A(L)$ est minimal.

- Tout automate minimal reconnaissant L est isomorphe à $A(L)$.

Soit $A = \langle Q, V, \delta, i, F \rangle$ un automate minimal reconnaissant L , et soit $A(L) = \langle R, V, \eta, j, G \rangle$. D'après la démonstration du théorème de Myhill-Nérode, le nombre d'états de $A(L)$ est au plus égal à celui des classes de \equiv_A , et donc le nombre de classes de \equiv_A est au plus égal au nombre d'états de A . Puisque A est minimal, le nombre d'états de $A(L)$, le nombre de classes de \equiv_A et le nombre d'états de A sont égaux. Par suite il n'y a qu'un état par classes de \equiv_A , autrement dit A est réduit.

L'automate A est aussi connecté, car s'il ne l'était pas, il serait équivalent à l'automate obtenu en lui enlevant ses états inaccessibles, et A ne serait pas minimal. Soit h l'application, définie dans la preuve du théorème de Myhill-Nérode, par : pour toute chaîne $x \in V^*$, $h([x]_{\equiv_L}) = [\delta(i, x)]_{\equiv_A}$. D'après cette preuve, h est une injection entre les états de $A(L)$ et les classes de \equiv_A . Puisque le nombre des états de $A(L)$ et le nombre des classes de \equiv_A sont égaux, h est une bijection. Soit h' l'application entre les états de $A(L)$ et ceux de A définie par : Pour toute chaîne $x \in V^*$, $h'([x]_{\equiv_L}) = \delta(i, x)$. Puisque h est une bijection et qu'il n'y a qu'un état par classe de \equiv_A , h' est aussi une bijection. Montrons que c'est un isomorphisme entre $A(L)$ et A . Il suffit de vérifier que c'est un homomorphisme (voir la Définition 3.3.3).

- (a) $h'(j) = i$.
 En effet, par définition de $A(L)$, $j = [\varepsilon]_{\equiv_L}$. Par définition de h' , $h'([\varepsilon]_{\equiv_L}) = \delta(i, \varepsilon) = i$.
 Par suite, $h'(j) = i$.
- (b) Pour tout $x \in V^*$, $a \in V$, $h'(\eta([x]_{\equiv_L}, a)) = \delta(h'([x]_{\equiv_L}, a))$.
 $h'(\eta([x]_{\equiv_L}, a)) = h'([xa]_{\equiv_L})$ D'après la propriété 3.4.4
 $h'([xa]_{\equiv_L}) = \delta(i, xa)$ Par définition de h'
 $\delta(i, xa) = \delta(\delta(i, x), a)$ Par la propriété 2.4.6 du chapitre 2
 $\delta(\delta(i, x), a) = \delta(h'([x]_{\equiv_L}), a)$ Par définition de h'
- (c) $h'(G) = F$.
 i. $h'(G) \subset F$
 Soit p un état de G . Il existe $x \in L$ tel que $p = [x]_{\equiv_L}$. Par définition de h' , $h'(p) = h'([x]_{\equiv_L}) = \delta(i, x)$. Puisque $x \in L$ et que A reconnaît L , $\delta(i, x) \in F$, donc $h'(p) \in F$.
 ii. $F \subset h'(G)$
 Soit $p \in F$. Puisque A est connecté, il existe x tel que $\delta(i, x) = p$. Puisque p est un état final de l'automate A reconnaissant L , $x \in L$. Par définition de h' , $p = h'([x]_{\equiv_L})$. Puisque $x \in L$, $[x]_{\equiv_L}$ est un état final de $A(L)$, donc $p \in h'(G)$. ■

Nous justifions, par une autre méthode, la construction de l'automate minimal fournie dans la Section 1.

Théorème 3.4.7 *Si A est un automate connecté, alors $m(A)$ est l'automate minimal équivalent à A .*

PREUVE. Soit A un automate connecté et L le langage reconnu par A . Soit $m(A) = \langle Q, V, \delta, i, F \rangle$. On sait que $m(A)$ est réduit et connecté. On en déduit que $m(A)$ est isomorphe à $A(L)$. Soit h l'application entre les états de $A(L)$ et les classes de la relation $\equiv_{m(A)}$ définie par : Pour tout $x \in V^*$, $h([x]_{\equiv_L}) = [\delta(i, x)]_{\equiv_{m(A)}}$. D'après la preuve du théorème de Mihill-Nérode, h est une injection. Puisque $m(A)$ est connecté, on vérifie facilement que c'est une bijection. Soit h' l'application entre les états de $A(L)$ et les états de $m(A)$ définie par : Pour tout $x \in V^*$, $h'([x]_{\equiv_L}) = \delta(i, x)$. Puisque h est une bijection et que $m(A)$ est réduit (autrement il n'y a qu'un état par classe de $\equiv_{m(A)}$), l'application h' est aussi une bijection entre les états de $A(L)$ et ceux de $m(A)$. Puisque $A(L)$ est minimal, $m(A)$ est aussi minimal. ■

Exercice 3.4.8 *Soit L un langage sur le vocabulaire V et soit $A = \langle Q, V, \delta, i, F \rangle$ l'automate suivant :*

- Les états de A sont les dérivées relativement à L .
- Pour tous $u \in V^*$, $a \in V$, $\delta(u^{-1}L, a) = (ua)^{-1}L$.
- L'état initial est L .
- Un état est final si et seulement si il est la dérivée d'une chaîne de L .

L'automate A est l'automate des dérivées de L . Montrez que cet automate reconnaît L et qu'il a le même nombre d'états que l'automate $A(L)$ des classes de L .

Chapitre 4

Expressions régulières

Ce chapitre est consacré à la définition d'un nouveau formalisme, les expressions régulières, et à la démonstration du théorème : un langage est reconnu par un automate fini si et seulement s'il peut être représenté par une expression régulière. La démonstration de ce résultat est constructive : elle définit des algorithmes de traduction des expressions régulières en automates finis équivalents et, pour la réciproque, de traduction des automates finis en expressions régulières. Ce formalisme équivalent est principalement utilisé pour décrire des langages réguliers de façon concise.

1 Définition des expressions régulières

On appelle *expression régulière* sur un vocabulaire V une expression dont les opérandes sont des lettres de V ou le symbole \emptyset et les opérateurs sont pris dans l'ensemble $\{+, \cdot, *\}$, où $+$ et \cdot sont binaires et $*$ est unaire. On écrit usuellement une expression régulière avec les conventions suivantes :

- on utilise la notation infixée pour les opérateurs binaires $+$ et \cdot , la notation en exposant pour l'opérateur unaire $*$
- les priorités des opérateurs sont dans l'ordre : $*$, \cdot , $+$.

Une expression régulière définit un langage sur V quand on identifie chaque lettre a de V au langage $\{a\}$, \emptyset au langage vide et que l'on interprète $+$ comme l'union, \cdot comme la concaténation des langages et $*$ comme la concaténation itérée de la Définition 1.2.31 du Chapitre 1. On dit encore que le langage ainsi défini est la valeur de l'expression régulière.

Définition 4.1.1 On appelle *langage régulier* tout langage qui peut être défini par une expression régulière. En particulier tout langage fini est régulier. \diamond

Exemple 4.1.2

1. $(\$ + _ + A + B + \dots + Z + a + b + \dots + z) \cdot (A + B + \dots + Z + a + b + \dots + z + 0 + 1 + \dots + 9)^*$ est une expression régulière définissant le langage des identificateurs de Java.
2. \emptyset^* est une expression régulière définissant le langage $\{\epsilon\}$. \clubsuit

Exercice 4.1.3

1. Montrer que les égalités suivantes sont vraies pour tout langage L :
 - $(L^*)^* = L^*$
 - $(\varepsilon + L)^* = L^*$
 - $L^*.L^* = L^*$
2. Montrer que, pour tout langage L, M, N et P , si $L \subset M$ et $N \subset P$, alors $L^* \subset M^*$ et $L.N \subset M.P$ (en d'autres termes, l'étoile et la concaténation sont des opérations croissantes.)
3. Montrer que les égalités suivantes sont vraies pour tous langages L et M :
 - $(L^*.M^*)^* = (L+M)^*$
 - $(L.M)^*L = L.(M.L)^*$
 - $(L.M+L)^*.L = L.(M.L+L)^*$
 Montrer que l'égalité suivante n'est pas valide : $(L+M)^* = L^* + M^*$
4. Soit $V = \{0, 1\}$. Donner les définitions des langages suivants en se servant des opérations de concaténation, d'union et de l'étoile :
 - (a) L'ensemble des chaînes sur V contenant au moins une occurrence de 0.
 - (b) L'ensemble des chaînes sur V de longueur impaire.
 - (c) L'ensemble des chaînes sur V ne contenant pas deux occurrences consécutives de 0, ni deux occurrences consécutives de 1.

2 Tout langage régulier est reconnu par un automate fini

Dans ce paragraphe sauf mention explicite, on identifie une expression régulière et le langage qui lui est associé.

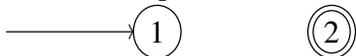
Définition 4.2.1 (Automate simple) Un automate fini est *simple* s'il a un seul état initial et un seul état final, si aucune transition n'a pour extrémité cet état initial et aucune transition n'a pour origine cet état final. \diamond

Théorème 4.2.2 *Tout langage régulier est reconnu par un automate fini.*

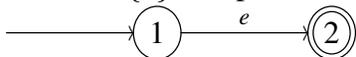
PREUVE. On prouve par induction structurelle la propriété suivante :

(1) Soit e une expression régulière sur V . Il existe un entier α (où $\alpha \geq 2$) et un automate simple $A = \langle \{1, 2, \dots, \alpha\}, V, \delta, \{1\}, \alpha \rangle$ qui reconnaît e .

Si $e = \emptyset$, l'expression est reconnue par l'automate simple :



Si $e \in V \cup \{\varepsilon\}$, l'expression est reconnue par l'automate simple :



Supposons que la propriété soit vérifiée par les expressions f et g . D'après cette hypothèse d'induction, il existe deux entiers α et β où $\alpha \geq 2$ et $\beta \geq 2$, un automate simple $A = \langle \{1, \dots, \alpha\}, V, \delta, \{1\}, \{\alpha\} \rangle$ reconnaissant f et un automate simple $B =$

$\langle \{1, \dots, \beta\}, V, \eta, \{1\}, \{\beta\} \rangle$ reconnaissant g . et montrons qu'elle est vérifiée par e ayant l'une des formes $f^*, f + g, fg$.

1. Cas $e = f^*$

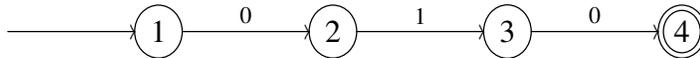
Soit l'automate $C = \langle \{1, \dots, \alpha + 1\}, V, \zeta, \{1\}, \{\alpha + 1\} \rangle$ où C est obtenu, en «collant» l'état initial et final de A , c'est-à-dire en changeant les états $1, \dots, \alpha - 1, \alpha$ de l'automate A en les états $2, \dots, \alpha, 2$, et en ajoutant les deux transitions $(1, \varepsilon, 2)$ et $(2, \varepsilon, \alpha + 1)$.

Plus précisément

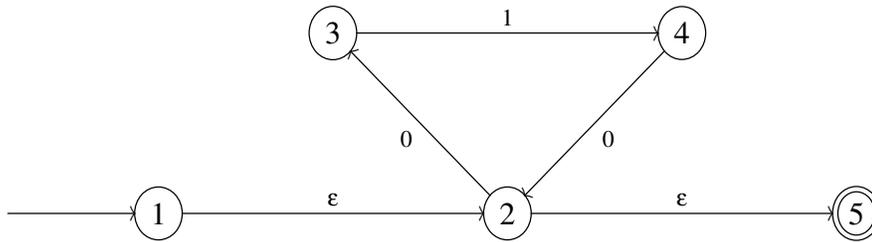
$$\begin{aligned} \zeta = & \{(p + 1, a, q + 1) \mid (p, a, q) \in \delta \text{ et } q < \alpha\} \\ & \cup \{(p + 1, a, 2) \mid (p, a, \alpha) \in \delta\} \\ & \cup \{(1, \varepsilon, 2), (2, \varepsilon, \alpha + 1)\} \end{aligned}$$

Il est évident, en examinant ζ , que l'automate C est simple. Nous montrons que C reconnaît e .

Exemple 4.2.3 Le langage 010 est reconnu par l'automate simple



Avec la construction indiquée, le langage $(010)^*$ est reconnu par l'automate simple



On montre que $L(C) = L(A)^*$.

(a) $L(A)^* \subset L(C)$.

On montre que C vérifie la propriété suivante : Si $x \in L(A)$ alors il existe un chemin de C de trace x de l'état 2 à l'état 2.

Soit $x \in L(A)$. Il existe $n \in \mathbb{N}$ et une suite de transitions, $(r_i, a_{i+1}, r_{i+1}) \in \delta$ pour $i = 0, \dots, n - 1$ telle que $r_0 = 1, r_n = \alpha$ et $x = a_1 \dots a_n$. Puisque l'automate A est simple, l'état α n'est l'origine d'aucune transition, donc pour $i = 0, \dots, n - 1, 1 \leq r_i < \alpha$. Par définition de ζ , il en résulte que pour $i = 0, \dots, n - 2, (r_i, a_{i+1}, r_{i+1}) \in \zeta$, et que $(r_{n-1}, a_n, 2) \in \zeta$. Donc il existe un chemin de C de trace x de l'état 2 à l'état 2 : la propriété est établie.

Soit $x \in L(A)^*$. Il existe $n \in \mathbb{N}$ tel que $x = y_1 \dots y_n$ et pour i de 1 à $n, y_i \in L(A)$.

D'après la propriété (a), pour i de 1 à n , il y a un chemin de C de trace y_i de l'état 2 à l'état 2, donc il existe un chemin s de C de trace x de l'état 2 à l'état 2. Par suite le mot x est reconnu par C grâce au chemin $(1, \varepsilon, 2)s(2, \varepsilon, \alpha + 1)$, et $x \in L(C)$.

(b) $L(C) \subset L(A)^*$.

On montre par induction sur n que l'automate C vérifie la propriété suivante : S'il

existe un chemin de C de trace x et de longueur n entre l'état 2 et l'état 2, alors $x \in L(A)^*$.

Supposons cette propriété vérifiée pour tout chemin de longueur inférieure à n . Soit un chemin de C de trace x et de longueur n entre l'état 2 et l'état 2. Il existe donc une suite de transitions $(r_i, a_{i+1}, r_{i+1}) \in \zeta$ pour i de 0 à $n-1$ telle que $r_0 = 2, r_n = 2$ et $x = a_1 \dots a_n$. Si $n = 0$ alors $x = \varepsilon$ donc $x \in L(A)^*$, ce qui prouve la propriété. Si $n > 0$ alors il existe un plus petit entier j tel que $j > 0$ et $r_j = 2$. Puisque l'automate C est simple, ni l'état 1, ni l'état $\alpha + 1$ ne figurent sur un chemin de l'état 2 à l'état 2. Donc par définition de ζ , pour i de 0 à $j-2$, $(r_i - 1, a_{i+1}, r_{i+1} - 1) \in \delta$, et $(r_{j-1}, a_j, \alpha) \in \delta$. Ainsi, il existe un chemin de A de trace $a_1 \dots a_j$ de l'état 1 à l'état α , autrement dit, $a_1 \dots a_j \in L(A)$. Par définition de j , il existe un chemin de C de longueur $n - j$, donc inférieure à n , entre l'état 2 et l'état 2 : par hypothèse d'induction, $a_{j+1}, \dots, a_n \in L(A)^*$, donc $x = a_1 \dots a_n \in L(A)^*$.

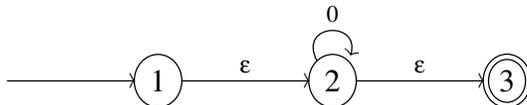
Ainsi $L(C) = L(A)^* = f^* = e$. L'automate C reconnaît bien l'expression e .

2. Cas $e = f + g$.

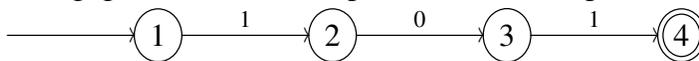
Soit l'automate $C = \langle \{1, \dots, \alpha + \beta - 2\}, V, \zeta, \{1\}, \{\alpha + \beta - 2\} \rangle$ où C est obtenu en «collant» d'une part l'état initial de A avec celui de B , et d'autre part l'état final de A avec celui de B . Ce collage est effectué en changeant l'état α de l'automate A en $\alpha + \beta - 2$ et en changeant pour i de 2 à β , l'état i de B en $i + \alpha - 2$. La relation de transition est définie de la façon suivante :

$$\begin{aligned} \zeta = & \{(p, a, q) \mid (p, a, q) \in \delta \text{ et } q < \alpha\} \cup \{(p, a, \alpha + \beta - 2) \mid (p, a, \alpha) \in \delta\} \\ & \cup \{(1, a, p + \alpha - 2) \mid (1, a, p) \in \eta\} \\ & \cup \{(p + \alpha - 2, a, q + \alpha - 2) \mid (p, a, q) \in \eta \text{ et } 1 < p\} \end{aligned}$$

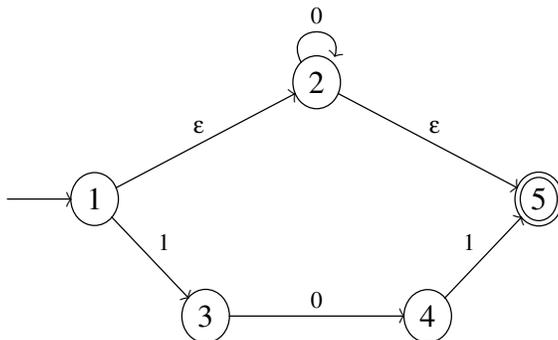
Exemple 4.2.4 Le langage 0^* est reconnu par l'automate simple



Le langage 101 est reconnu par l'automate simple



Avec la construction indiquée, le langage $0^* + 101$ est reconnu par l'automate simple



Il est clair que C est un automate simple. Nous montrons que C reconnaît e . Un chemin de A de l'état 1 à l'état α devient après renommage des états un chemin de C de l'état 1 à l'état $\alpha + \beta - 2$, un chemin de B de l'état 1 à l'état β devient aussi après renommage des états un chemin de C de l'état 1 à l'état $\alpha + \beta - 2$; donc, $L(A) \cup L(B) \subset L(C)$.

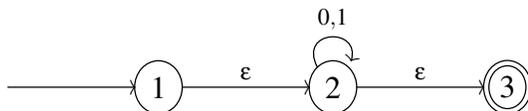
Réciproquement, puisque les automates A et B sont simples et que l'automate C est obtenu en «collant» d'une part l'état initial de A avec celui de B , et d'autre part l'état final de A avec celui de B , tout chemin de C , entre l'état 1 et l'état $\alpha + \beta - 2$, est, au nom près des états, soit un chemin de A de l'état 1 à l'état α , soit un chemin de B de l'état 1 à l'état β . Donc $L(C) = L(A) \cup L(B)$, et C reconnaît bien l'expression e .

3. Cas $e = fg$.

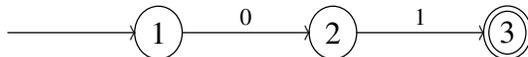
Soit l'automate $C = \langle \{1, \dots, \alpha + \beta - 1\}, V, \zeta, \{1\}, \{\alpha + \beta - 1\} \rangle$ où C est obtenu en «collant» l'état final de A et l'état initial de B . Tout état i de l'automate B est renommé $i + \alpha - 1$, et la relation de transition est définie par :

$$\zeta = \delta \cup \{(p + \alpha - 1, a, q + \alpha - 1) \mid (p, a, q) \in \eta\}$$

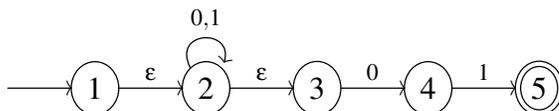
Exemple 4.2.5 Le langage $(0 + 1)^*$ est reconnu par l'automate simple



Le langage 01 est reconnu par l'automate simple



Avec la construction indiquée, le langage $(0 + 1)^*00$ est reconnu par l'automate simple



Il est facile de prouver que $L(C) = L(A)L(B)$ et que l'automate C reconnaît bien e . ■

Exercice 4.2.6 Construire avec la méthode décrite dans la preuve ci-dessus, un automate fini simple reconnaissant $((10)^*1^*)^*$.

3 Tout langage reconnu par un automate fini est un langage régulier

Dans la suite, pour dénoter les opérations sur les langages (union, concaténation), on utilise les notations des expressions régulières.

Théorème 4.3.1 (Résolution d'une équation) Soient α et β deux langages sur le vocabulaire V et x une variable. L'équation $x = \alpha x + \beta$ a une solution $x = \alpha^* \beta$ et cette solution est la plus petite des solutions de l'équation.

PREUVE. $x = \alpha^*\beta$ est évidemment une solution car $\alpha^*\beta = \alpha(\alpha^*\beta) + \beta$. Montrons que c'est la plus petite des solutions. Soit $x = L$ une autre solution. Par induction sur $i \in \mathbb{N}$, on montre que pour tout i , $\alpha^i\beta \subset L$

1. $i = 0$. On a $\alpha^0\beta = \{\varepsilon\}\beta = \beta$. Puisque $L = \alpha L + \beta$, on a bien $\alpha^0\beta \subset L$.
2. $i > 0$. Supposons que $\alpha^i\beta \subset L$. Puisque la concaténation est croissante, $\alpha^{i+1}\beta \subset \alpha L$. Mais comme $L = \alpha L + \beta$, on a $\alpha^{i+1}\beta \subset L$.

Soit $u \in \alpha^*\beta$. Par définition de l'étoile, il existe $i \in \mathbb{N}$ tel que $u \in \alpha^i\beta$. Donc d'après ce qui précède, $u \in L$, et $\alpha^*\beta \subset L$. ■

Remarque Les équations sur les langages peuvent avoir plusieurs solutions. Avec le vocabulaire $V = \{a\}$, l'équation $x = x + a$ admet comme solution tout langage L sur V tel que $a \in L$. En effet si $a \in L$, alors $L = L + a$.

Avec le vocabulaire $V = \{a, b\}$, l'équation $x = (a + \varepsilon)x + b$ a comme solution tout langage $a^*(L + b)$ où L est un langage quelconque sur V . En effet $(a + \varepsilon)a^*(L + b) + b = a^*(L + b) + b = a^*(L + b)$.

Avec le vocabulaire $V = \{a\}$, l'équation $x = ax$ a comme unique solution $x = \emptyset$.

Définition 4.3.2 (Système régulier d'équations) Soit V un vocabulaire. Un système régulier de $n \in \mathbb{N}$ équations sur V est un ensemble de n variables x_1, \dots, x_n et de n équations où de la forme

$$x_i = \beta_i + \sum_{j=1}^n \alpha_{i,j} x_j,$$

où $i = 1, \dots, n$, et les coefficients $\alpha_{i,j}$ et β_i sont des langages réguliers sur V . ◇

Théorème 4.3.3 (Résolution d'un système régulier d'équations) *Tout système régulier sur le vocabulaire V avec n variables x_1, \dots, x_n admet une plus petite solution régulière, c'est-à-dire une solution $x_1 = L_1, \dots, x_n = L_n$ où L_1, \dots, L_n sont des langages réguliers sur V .*

PREUVE. On prouve ce résultat par induction sur n , et la preuve est la recherche de la solution par la méthode de Gauss : la valeur de x_n est calculée en fonction des valeurs des autres variables, puis la variable x_n est éliminée, ce qui réduit le calcul à la recherche de la solution d'un système régulier à $n - 1$ variables. Soit S_1 le système régulier sur x_1, \dots, x_n , où pour tout $i = 1 \dots, n$,

$$x_i = \beta_i + \sum_{j=1}^n \alpha_{i,j} x_j.$$

1. $n = 1$.

La seule équation du système est $x_1 = \beta_1 + \alpha_{1,1}x_1$. D'après le Théorème 4.3.1, cette équation a comme plus petite solution $x_1 = \alpha_{1,1}^*\beta_1$. Puisque $\alpha_{1,1}$ et β_1 sont des langages réguliers, il en est de même de la valeur de x_1 .

2. $n > 1$.

Supposons que tout système régulier à $n - 1$ de variables x_1, \dots, x_{n-1} a une plus petite solution régulière. D'après le Théorème 4.3.1, si le système S_1 a une solution, elle vérifie

$$x_n = \alpha_{n,n}^*(\beta_n + \sum_{j=1}^{n-1} \alpha_{n,j}x_j).$$

Ceci permet de remplacer x_n par sa valeur en fonction des variables x_1, \dots, x_{n-1} . On obtient le nouveau système S_2 de la forme

$$x_i = \beta'_i + \sum_{j=1}^{n-1} \alpha'_{i,j}x_j,$$

où, pour tout $i = 1, \dots, n - 1$,

$$\begin{aligned} \beta'_i &= \beta_i + \alpha_{i,n}(\alpha_{n,n})^*\beta_n, \\ \alpha'_{i,j} &= \alpha_{i,j} + \alpha_{i,n}(\alpha_{n,n})^*\alpha_{n,j}. \end{aligned}$$

Il est clair que ce système est régulier, et par définition du système S_2 , l'ensemble $\{x_1 = L_1, \dots, x_{n-1} = L_{n-1}, x_n = L_n\}$ est la plus petite solution de S_1 si et seulement si l'ensemble $\{x_1 = L_1, \dots, x_{n-1} = L_{n-1}\}$ est la plus petite solution de S_2 , et $L_n = \alpha_{n,n}^*(\beta_n + \sum_{j=1}^{n-1} \alpha_{n,j}L_j)$.

Par hypothèse d'induction, le système S_2 a une plus petite solution régulière $\{x_1 = L_1, \dots, x_{n-1} = L_{n-1}\}$. Puisque L_n est construit par les opérations union, concaténation, étoile appliquées à des langages réguliers, L_n est un langage régulier. Par suite S_1 a une plus petite solution régulière. ■

Exemple 4.3.4 Cherchons la plus petite solution du système suivant sur le vocabulaire $\{0, 1\}$.

$$\begin{aligned} x_1 &= 0x_1 + 1x_2 + \varepsilon \\ x_2 &= 1x_1 + 0x_3 \\ x_3 &= 0x_2 + 1x_3 \end{aligned}$$

D'après le Théorème 4.3.1, la plus petite solution vérifie $x_3 = 1^*0x_2$. En reportant cette expression de x_3 dans les deux premières équations, nous obtenons

$$\begin{aligned} x_1 &= 0x_1 + 1x_2 + \varepsilon \\ x_2 &= 1x_1 + 01^*0x_2 \end{aligned}$$

Encore d'après le théorème 4.3.1, la plus petite solution vérifie $x_2 = (01^*0)^*1x_1$. En reportant cette expression de x_2 dans la première équation, nous obtenons

$$x_1 = 0x_1 + 1(01^*0)^*1x_1 + \varepsilon$$

Toujours d'après le théorème 4.3.1, la plus petite solution en x_1 est donc $x_1 = (0 + 1(01^*0)^*1)^* + \varepsilon = (0 + 1(01^*0)^*1)^*$. En reportant la valeur de x_1 dans les équations $x_2 = (01^*0)^*1x_1$ et $x_3 = 1^*0x_2$, on obtient la plus petite solution :

$$\begin{aligned} x_1 &= (0 + 1(01^*0)^*1)^* \\ x_2 &= (01^*0)^*1(0 + 1(01^*0)^*1)^* \\ x_3 &= 1^*0(01^*0)^*1(0 + 1(01^*0)^*1)^* \end{aligned}$$

4 Système d'équations associé à un automate fini

Théorème 4.4.1 Soit $A = \langle Q, V, \delta, I, F \rangle$ un automate fini et q_1, \dots, q_n une liste sans répétition des états de l'automate ; à chacun de ces états q_i on associe une variable distincte x_i . Pour $i = 1, \dots, n$, soit $A_i = \langle Q, V, \delta, \{q_i\}, F \rangle$ l'automate identique à A , sauf que son unique état initial est q_i . Pour tout $i, j = 1, \dots, n$, on pose $\alpha_{i,j} = \{a \in V \cup \{\varepsilon\} \mid (q_i, a, q_j) \in \delta\}$.

Le système d'équations sur V défini pour $i = 1, \dots, n$ par :

$$\begin{aligned} \text{si } q_i \in F \quad x_i &= \varepsilon + \sum_{j=1}^n \alpha_{i,j} x_j \\ \text{si } q_i \notin F \quad x_i &= \sum_{j=1}^n \alpha_{i,j} x_j \end{aligned}$$

à pour plus petite solution $x_1 = L(A_1), \dots, x_n = L(A_n)$.

PREUVE. Nous prouvons que $x_1 = L(A_1), \dots, x_n = L(A_n)$ est une solution du système, puis que c'est la plus petite solution.

1. $x_1 = L(A_1), \dots, x_n = L(A_n)$ est une solution du système.

Pour toute chaîne u , on a $u \in L(A_i)$ si et seulement si, par définition de A_i , il existe un chemin de trace u entre q_i et un état de F . Par définition des traces, ceci est équivalent à :

- soit $u = \varepsilon$ et $q_i \in F$,
- soit il existe un chemin avec au moins une transition de trace u entre q_i et un état de F ;

ce qui équivaut à :

- soit $u = \varepsilon$ et $q_i \in F$,
- soit il existe un entier j compris entre 1 et n , un symbole a et une chaîne v tels que $u = av$, $a \in \alpha_{i,j}$ et $v \in L(A_j)$.

On en déduit que

$$\begin{aligned} \text{si } q_i \in F \quad L(A_i) &= \varepsilon + \sum_{j=1}^n \alpha_{i,j} L(A_j) \\ \text{si } q_i \notin F \quad L(A_i) &= \sum_{j=1}^n \alpha_{i,j} L(A_j), \end{aligned}$$

Ce qui prouve la propriété 1.

2. $x_1 = L(A_1), \dots, x_n = L(A_n)$ est la plus petite solution.

Soit $x_1 = M_1, \dots, x_n = M_n$ une solution du système. Soit $L_{i,j}$ l'ensemble des chaînes $u \in V^*$ telles qu'il existe un chemin de longueur j et de trace u entre l'état q_i et un état de F . On note $P(j)$ la propriété : pour tout $i = 1, \dots, n, L_{i,j} \subset M_i$. On montre que $P(j)$ est vraie pour tout $j \in \mathbb{N}$. Par définition des $L_{i,j}$, nous avons

$$\begin{aligned} \text{si } q_i \in F \quad L_{i,0} &= \{\varepsilon\}, \\ \text{si } q_i \notin F \quad L_{i,0} &= \emptyset. \end{aligned}$$

Puisque $x_1 = M_1, \dots, x_n = M_n$ est une solution du système, pour $i = 1, \dots, n$, si $q_i \in F$ alors $\varepsilon \in M_i$. On en déduit que pour tout $i = 1, \dots, n, L_{i,0} \subset M_i$, ce qui prouve $P(0)$.

Supposons $P(j)$ vérifiée et prouvons $P(j+1)$. Par définition des $L_{i,j}$ et des $\alpha_{i,j}$, nous avons

$$L_{i,j+1} = \sum_{k=1}^n \alpha_{i,k} L_{k,j}. \quad (4.1)$$

Par hypothèse d'induction $L_{k,j} \subset M_k$, donc $\alpha_{i,k} L_{k,j} \subset \alpha_{i,k} M_k$. De l'égalité 4.1 et de cette dernière inégalité, il résulte que

$$L_{i,j+1} \subset \sum_{k=1}^n \alpha_{i,k} M_k. \quad (4.2)$$

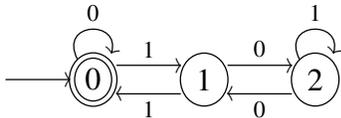
Puisque $x_1 = M_1, \dots, x_n = M_n$ est une solution du système, on en déduit que pour tout $i = 1, \dots, n$,

$$\sum_{k=1}^n \alpha_{i,k} M_k \subset M_i. \quad (4.3)$$

Des inégalités 4.2 et 4.3, il résulte que pour $i = 1, \dots, n, L_{i,j+1} \subset M_i$, ce qui montre que la propriété $P(j)$ est vraie pour tout $j \in \mathbb{N}$.

Il nous reste à prouver que pour i de 1 à $n, L(A_i) \subset M_i$. Soit $u \in L(A_i)$. Par définition des $L_{i,j}$, il existe j tel que $j \geq 0$ et $u \in L_{i,j}$. D'après la propriété $P(j)$, pour tout $i \in \mathbb{N}, L_{i,j} \subset M_i$. Donc $u \in M_i$. Par suite $x_1 = L(A_1), \dots, x_n = L(A_n)$ est bien la plus petite solution du système. ■

Exemple 4.4.2 Soit A l'automate ci-dessous



Cet automate déterministe reconnaît les chaînes binaires représentant un nombre écrit en base 2 et divisible par 3. Pour le prouver, il faut associer les états 0, 1, 2 aux nombres binaires égaux respectivement à 0, 1, 2 modulo 3 et remarquer que le corps $\mathbb{Z}/3\mathbb{Z}$ on a

— $2 * 0 + 0 = 0, 2 * 0 + 1 = 1$, ce qui «explique» les flèches sortant de l'état 0

- $2 * 1 + 0 = 2, 2 * 1 + 1 = 0$, ce qui «explique» les flèches sortant de l'état 1
- $2 * 2 + 0 = 1, 2 * 2 + 1 = 2$, ce qui «explique» les flèches sortant de l'état 2

D'après le théorème 4.4.1 on associe à cet automate le système

$$\begin{aligned}x_1 &= 0x_1 + 1x_2 + \varepsilon \\x_2 &= 1x_1 + 0x_3 \\x_3 &= 0x_2 + 1x_3\end{aligned}$$

dont les variables x_1, x_2, x_3 correspondent respectivement aux états 0, 1, 2, et dont la plus petite solution vérifie $x_1 = L(A)$. Nous avons déjà trouvé cette solution dans l'Exemple 4.3.4 : on a $L(A) = (0 + 1(01^*0)^*1)^*$. ♣

Théorème 4.4.3 *Tout langage reconnu par un automate fini est un langage régulier.*

PREUVE. Soit $A = \langle Q, V, \delta, I, F \rangle$ un automate fini. Soit q_1, \dots, q_n une liste sans répétition des états de l'automate. Pour i de 1 à n , on pose $A_i = \langle Q, V, \delta, \{q_i\}, F \rangle$. Par définition des automates A_i , on a : $L(A) = \sum_{i \text{ tel que } q_i \in I} L(A_i)$.

D'après le Théorème 4.4.1, on associe à l'automate A un système régulier (ses coefficients sont des langages finis) dont les variables correspondent respectivement aux états q_1, \dots, q_n et dont la plus petite solution est $x_1 = L(A_1), \dots, x_n = L(A_n)$. D'après le Théorème 4.3.3, le système étant régulier, sa plus petite solution est régulière, donc les langages $L(A_i)$ sont réguliers. Puisque $L(A)$ est une somme finie de ces langages, $L(A)$ est un langage régulier. ■

Théorème 4.4.4 *Un langage est reconnu par un automate fini si et seulement s'il est défini par une expression régulière.*

PREUVE. D'après le Théorème 4.4.3, un langage reconnu par un automate est régulier. Réciproquement, d'après le Théorème 4.2.2, un langage régulier est reconnu par un automate fini. ■

Chapitre 5

Propriétés de fermeture des langages réguliers

Le but de ce chapitre est d'étudier certaines opérations dont l'application à un (des) langage(s) régulier(s) engendre nécessairement un langage régulier. Si certaines de ces propriétés peuvent être utilisées pour prouver qu'un langage est régulier, en général, elles servent plutôt à démontrer qu'un langage *n'est pas* régulier. Le principe est simple : s'il est possible d'appliquer des opérations préservant la régularité à un langage L et d'aboutir à un langage L' qui lui n'est pas régulier, cela signifie que le langage L ne peut pas être régulier.

Cette méthode de preuve nécessite cependant de savoir au préalable que le langage L' n'est pas régulier. Nous présentons donc dans un premier temps une méthode permettant de prouver qu'un langage n'est pas régulier, grâce au *lemme de l'étoile*, avant d'exposer différentes propriétés de fermeture et de montrer comment s'en servir une fois qu'au moins un langage non-régulier est connu.

1 Le lemme de l'étoile

Le lemme de l'étoile est une condition nécessaire vérifiée par les langages réguliers. Ainsi, si un langage donné ne le vérifie pas, alors il ne peut pas être régulier. Ce lemme est donc utilisé pour effectuer des preuves par l'absurde, en supposant qu'un langage est régulier et en prouvant que le lemme de l'étoile n'est pas satisfait.

Remarque Le lemme de l'étoile n'est pas une condition suffisante, qui garantit qu'un langage est régulier : il existe des langages qui ne sont pas réguliers, mais pour lesquels le lemme de l'étoile est valide. En résumé, si le lemme de l'étoile n'est pas vérifié par un langage, ce dernier ne peut pas être régulier. Mais dans le cas contraire, on ne peut aboutir à aucune conclusion concernant le langage.

Nous définissons l'itération d'un chemin avant de prouver le résultat.

Définition 5.1.1 Soit p un chemin dans un automate fini, tel que l'origine et l'extrémité de p sont identiques. Notons q l'origine (et l'extrémité) de p . On définit alors $p^0 = (q, \varepsilon, q)$, et, pour tout $i \geq 1$, $p^i = p.p^{i-1}$. \diamond

Autrement dit, p^i représente le chemin obtenu par concaténation de p au chemin p^{i-1} . Comme l'origine et l'extrémité de p sont identiques, cette opération est bien définie. Intuitivement, quand p est une boucle, p^i représente le parcours de cette boucle i fois.

Proposition 5.1.2 Soit p un chemin dans un automate fini A , dont l'origine et l'extrémité sont identiques. Si p est de trace z , alors pour tout $i \in \mathbb{N}$, p^i est un chemin dans A , de trace z^i .

PREUVE. Par induction triviale sur i . \blacksquare

Soit L un langage régulier, et soit $A = \langle Q, V, \delta, I, F \rangle$ un automate fini sans ε -transition¹ qui reconnaît L . Dénotons par n le nombre d'états de A (i.e. $n = |Q|$), et considérons un mot $z = z_1 \cdots z_m$ de longueur m , où $m \geq n$. Si $z \in L$, alors par définition, il existe un chemin p de $q_0 \in I$ à $q_m \in F$ de trace z . Ce chemin, de longueur m , est donc constitué de $m + 1 > n$ états de Q , ce qui signifie qu'un état s'y retrouve nécessairement deux fois. Plus formellement, p est de la forme

$$p = (q_0, z_1, q_1)(q_1, z_2, q_2) \cdots (q_{m-1}, z_m, q_m),$$

et il existe $0 \leq j < k \leq m$ tels que $q_j = q_k$. Le chemin p peut donc être réexprimé sous la forme

$$p = \underbrace{(q_0, z_1, q_1) \cdots (q_{j-1}, z_j, q_j)}_{p_1} \underbrace{(q_j, z_{j+1}, q_{j+1}) \cdots (q_{k-1}, z_k, q_k)}_{p_2} \underbrace{(q_k, z_{k+1}, q_{k+1}) \cdots (q_{m-1}, z_m, q_m)}_{p_3},$$

c'est-à-dire décomposé en les trois sous-chemins p_1, p_2 et p_3 , de traces respectives $u = z_1 \cdots z_j$, $v = z_{j+1} \cdots z_k$ et $w = z_{k+1} \cdots z_m$. Comme on a supposé que A ne contenait pas d' ε -transition, aucune de ces traces ne peut être réduite à la chaîne vide.

En particulier, comme p_2 est un chemin dont l'origine et l'extrémité sont identiques, d'après la Proposition 5.1.2, pour tout $i \in \mathbb{N}$, p_2^i est un chemin dans A , et $p_1.p_2^i.p_3$ est également un chemin dans A . Ce dernier chemin est de trace $uv^i w$; comme il a pour origine $q_0 \in I$ et pour extrémité $q_m \in F$, ceci signifie que le mot $uv^i w$ est reconnu par A , et est donc élément de L ; cette propriété est vraie quel que soit $i \in \mathbb{N}$.

Lemme 5.1.3 (Lemme de l'étoile) Soit L un langage régulier. Alors il existe $n \geq 1$ tel que pour tout $z \in L$, si $|z| \geq n$ alors z est de la forme uvw , où :

- $|v| \geq 1$;
- $|uv| \leq n$;
- $\forall i \geq 0, uv^i w \in L$.

1. Cette condition n'est pas nécessaire, mais elle permet une démonstration plus aisée du résultat. En particulier, si A possédait des ε -transitions, on pourrait le remplacer par A' , automate sans ε -transitions, avec le même nombre d'états que A (voir le paragraphe 3 du Chapitre 2).

Remarque Si A est un automate reconnaissant un langage régulier L , alors le nombre n du lemme de l'étoile est au plus égal au nombre d'états de A . On pourrait donc préciser dans l'énoncé du lemme que la valeur de n est au plus égale au *plus petit* automate reconnaissant L . Ceci n'a cependant que très peu d'utilité en pratique, car aucune information n'est connue quant au nombre d'états de ce plus petit automate.

Exemple 5.1.4 ($\{0^m 1^m \mid m \geq 0\}$ n'est pas régulier.) Soit $L = \{0^m 1^m \mid m \geq 0\}$, on cherche à prouver que L n'est pas régulier. Supposons que L est régulier, alors ce langage devrait vérifier le lemme de l'étoile. Prenons n comme dans le lemme de l'étoile. Sa valeur n'est pas connue, mais il est garanti que tout mot de longueur au moins n vérifie les conditions du lemme.

Soit $z = 0^n 1^n$; par définition, z est élément de L , et est de longueur $|z| = 2n \geq n$. D'après le lemme de l'étoile, z est de la forme uvw , où $|uv| \leq n$, $|v| \geq 1$ et $\forall i \geq 0, uv^i w \in L$. Posons $k = |uv|$; par hypothèse, $k \leq n$, ce qui implique que uv , qui est un préfixe de z , est nécessairement de la forme 0^k . Toujours par hypothèse, en prenant $i = 0$, on doit avoir $uv^0 w = uw \in L$. Or, $uw = 0^{n-|v|} 1^n$, et comme $|v| \geq 1$, ce mot n'est pas élément de L , ce qui contredit l'hypothèse de départ que L est régulier. ♣

Il est important de noter que le lemme de l'étoile exprime une propriété nécessaire **mais pas suffisante** sur les langages réguliers : il existe des langages qui vérifient cette propriété mais ne sont pas réguliers, comme le montre l'exemple ci-dessous.

Exemple 5.1.5 L'exemple 5.1.4 prouve que le langage $L = \{a^m b^m \mid m \geq 0\}$ n'est pas régulier. Considérons maintenant le langage $M = \{c\}^+ . L \cup \{a, b\}^*$, sur le vocabulaire $V = \{a, b, c\}$, et considérons tous les mots de M de longueur au moins 1. Soit $z \in M$ de longueur supérieure ou égale à 1. Si $z \in \{c\}^+ . L$, alors on écrit z sous la forme uvw , où $u = \varepsilon$ et $v = c$. Sinon, on écrit z sous la forme uvw , où $u = \varepsilon$ et v est la première lettre de z . Il est simple de vérifier que dans les deux cas, pour tout $i \geq 0$, $uv^i w$ est élément de M . Ce langage vérifie donc la conclusion du lemme de l'étoile, mais il n'est pas régulier, comme le montrera l'Exemple 5.2.13. ♣

Nous terminons cette partie en présentant une condition nécessaire et suffisante de régularité d'un langage (voir le livre de Arto Salomaa, *Computation and Automata*, pour plus de détails).

Théorème 5.1.6 *Un langage $L \subseteq V^*$ est régulier si et seulement s'il existe un entier $n \geq 1$ tel que, pour tout $z \in V^*$, si $|z| \geq n$, alors z est de la forme uvw où :*

- $|v| \geq 1$;
- $\forall i \geq 0, \forall x \in V^*, uwx \in L \Leftrightarrow uv^i wx \in L$.

PREUVE. Supposons que L est régulier, soit $A = \langle Q, V, \delta, q_0, F \rangle$ un automate fini déterministe qui reconnaît L , et posons $n = |Q|$. Comme pour le lemme de l'étoile, il est simple de prouver que z peut être décomposé en uvw , où $v \neq \varepsilon$, tel que pour tout $i \geq 0$, $\delta^*(q_0, uv^i w) = \delta^*(q_0, uw)$. Donc, pour tout $x \in V^*$, $\delta^*(q_0, uv^i wx) = \delta^*(q_0, uwx)$, d'où le résultat.

Supposons maintenant qu'il existe un entier $n \geq 1$ tel que les conditions du théorème soient valables, et construisons un automate fini déterministe A tel que $L(A) = L$. L'ensemble des états de A est défini en associant à chaque mot x de longueur inférieure à n un état q_x ; on a donc

l'ensemble d'états $Q = \{q_x \mid x \in V^* \text{ et } |x| < n\}$. L'état initial de A est q_ε , et l'ensemble des états finals est défini par : $F = \{q_x \in Q \mid x \in L\}$. Enfin, la fonction de transition δ est définie pour tout $q_y \in Q$ et pour tout $a \in V$ par :

- si $|y| < n - 1$, alors $\delta(q_y, a) = q_{ya}$;
- sinon, y est de longueur $n - 1$, et par hypothèse, ya peut être décomposé en uvw , où $v \neq \varepsilon$ et, pour tout $i \geq 0$ et pour tout $x \in V^*$, $uwx \in L$ si et seulement si $uv^iwx \in L$. S'il y a plusieurs décompositions possibles de ya , alors on en fixe une ², et on pose $\delta(q_y, a) = q_{uw}$.

Il reste à prouver que $L(A) = L$; nous démontrons ce résultat par induction sur la longueur des mots reconnus par A . D'après la définition des états finals de A et de la fonction de transition δ , il est clair que pour tout mot z tel que $|z| < n$, z est reconnu par A si et seulement si $z \in L$. Soit maintenant un $k \geq n$ fixé, et supposons que pour tout mot z de longueur strictement inférieure à k , $z \in L(A)$ si et seulement si $z \in L$. Soit z un mot de longueur k , on pose $z = z_0x$, où $|z_0| = n$. Alors on a la décomposition $z_0 = uvw$, où $v \neq \varepsilon$ et $uwx \in L$ si et seulement si $uv^iwx \in L$, pour tout $i \geq 0$. Donc, en particulier, $uwx \in L$ si et seulement si $uvw \in L$. Par définition de δ , on a $\delta^*(q_\varepsilon, z_0) = q_{uw}$, d'où :

$$\begin{aligned} z \in L(A) &\Leftrightarrow \delta^*(q_\varepsilon, z) \in F &&\Leftrightarrow \delta^*(q_\varepsilon, z_0x) \in F \\ &\Leftrightarrow \delta^*(\delta^*(q_\varepsilon, z_0), x) \in F &&\Leftrightarrow \delta^*(q_{uw}, x) \in F \\ &\Leftrightarrow \delta^*(q_\varepsilon, uw) \in F &&\Leftrightarrow uw \in L(A). \end{aligned}$$

Comme uw est de longueur strictement inférieure à k , nous pouvons appliquer l'hypothèse d'induction pour conclure : $uw \in L(A) \Leftrightarrow uw \in L \Leftrightarrow uvw \in L \Leftrightarrow z \in L$, d'où le résultat. ■

2 Opérations préservant la régularité d'un langage

Nous allons démontrer que si certaines opérations sont appliquées à des langages réguliers, alors elles engendrent des langages qui sont eux aussi réguliers; on dit alors que la classe des langages réguliers est *fermée* pour cette opération.

2.1 Opérations ensemblistes

La propriété de fermeture des langages réguliers est évidente pour un premier ensemble d'opérations :

Lemme 5.2.1 *La classe des langages réguliers est fermée par :*

- concaténation,
- union,
- fermeture de Kleene (l'étoile “*”).

PREUVE. C'est une conséquence immédiate du Théorème 4.2.2, en passant par les expressions régulières. ■

2. par exemple, on choisit la décomposition telle que $(|u|, |v|)$ est minimal pour l'ordre produit sur $\mathbb{N} \times \mathbb{N}$.

Un autre résultat de fermeture simple à démontrer est le suivant : si L est un langage régulier, alors le complément de L , noté \bar{L} est également régulier.

Lemme 5.2.2 *La classe des langages réguliers est fermée par l'opération de complémentation.*

PREUVE. Soit L un langage régulier, montrons qu'il existe un automate A' qui reconnaît \bar{L} , c'est-à-dire qu'un mot w est reconnu par A' si et seulement s'il n'est pas dans L .

Comme L est régulier, il existe un automate fini *déterministe* $A = \langle Q, V, \delta, q_0, F \rangle$ qui reconnaît L . Posons $A' = \langle Q, V, \delta, q_0, F' \rangle$, où $F' = Q \setminus F$; nous allons prouver que A' reconnaît \bar{L} . Soit $w \in V^*$:

$$\begin{aligned} w \text{ est reconnu par } A' &\Leftrightarrow \delta^*(q_0, w) \in F' \\ &\Leftrightarrow \delta^*(q_0, w) \in Q \setminus F \\ &\Leftrightarrow \delta^*(q_0, w) \notin F \\ &\Leftrightarrow w \text{ n'est pas reconnu par } A. \end{aligned}$$

Ceci prouve le résultat. ■

Corollaire 5.2.3 *La classe des langages réguliers est fermée par :*

- *intersection,*
- *différence.*

PREUVE. On a les égalités :

$$\begin{aligned} L_1 \cap L_2 &= \overline{\bar{L}_1 \cup \bar{L}_2} \\ L_1 \setminus L_2 &= L_1 \cap \bar{L}_2. \end{aligned}$$

En se servant des résultats précédents, la preuve est immédiate. ■

2.2 Substitutions régulières

Définition 5.2.4 Soient deux vocabulaires V et W pas nécessairement disjoints. Une *substitution régulière* est une fonction $s : V \rightarrow \mathcal{P}(W^*)$, qui à chaque lettre $a \in V$ associe un langage régulier $s(a)$ sur le vocabulaire W .

La définition d'une substitution régulière est étendue à V^* par induction, en posant :

- $s(\epsilon) = \epsilon$,
- $s(a.w) = s(a).s(w)$, pour toute lettre $a \in V$ et tout mot $w \in V^*$.

Enfin, la définition d'une substitution régulière est étendue aux langages de V^* , en posant :

$$\forall L \subseteq V^*, s(L) = \bigcup_{w \in L} s(w). \quad \diamond$$

Proposition 5.2.5 *Pour tout $x, y \in V^*$ et pour toute substitution régulière s , $s(xy) = s(x)s(y)$.*

PREUVE. Par induction triviale sur la longueur de x . ■

Lemme 5.2.6 Soient L, L' des langages réguliers sur V et s une substitution régulière. Alors :

1. $s(L.L') = s(L).s(L')$;
2. $s(L \cup L') = s(L) \cup s(L')$;
3. $s(L^*) = [s(L)]^*$.

PREUVE. Nous montrons chaque résultat par double inclusion.

$s(L.L') = s(L).s(L')$. Soit $w \in s(L.L')$, nous prouvons que $w \in s(L).s(L')$. Par définition, il existe $u \in L.L'$ tel que $w \in s(u)$, et u est de la forme $u_1.u_2$, où $u_1 \in L$ et $u_2 \in L'$. D'après la proposition 5.2.5, $s(u) = s(u_1)s(u_2)$, et w est donc de la forme w_1w_2 , où $w_1 \in s(u_1) \subseteq s(L)$ et $w_2 \in s(u_2) \subseteq s(L')$. D'où $w \in s(L)s(L')$, et $s(LL') \subseteq s(L)s(L')$.

Soit $w \in s(L)s(L')$, nous montrons que $w \in s(L.L')$. Par définition, w est de la forme w_1w_2 , où $w_1 \in s(L)$ et $w_2 \in s(L')$. Il existe donc $u_1 \in L$ tel que $w_1 \in s(u_1)$, et il existe $u_2 \in L'$ tel que $w_2 \in s(u_2)$. Donc, $w \in s(u_1)s(u_2)$, et d'après la proposition 5.2.5, $w \in s(u_1u_2) \subseteq s(L.L')$, d'où le résultat.

$s(L \cup L') = s(L) \cup s(L')$. Soit $w \in s(L \cup L')$, il existe $u \in L \cup L'$ tel que $w \in s(u)$. Si $u \in L$, alors $w \in s(u) \subseteq s(L)$, et si $u \in L'$ alors $w \in s(u) \subseteq s(L')$. Donc, $w \in s(L) \cup s(L')$.

Il est clair que $s(L) \subseteq s(L \cup L')$ et $s(L') \subseteq s(L \cup L')$, donc $s(L) \cup s(L') \subseteq s(L \cup L')$, d'où l'égalité.

$s(L^*) = [s(L)]^*$. Nous démontrons par induction que pour tout $i \geq 0$, $s(\bigcup_{j=0}^i L^j) = \bigcup_{j=0}^i [s(L)]^j$. Pour $i = 0$, on a $s(L^0) = s(\{\epsilon\}) = \{\epsilon\} = [s(L)]^0$. Supposons le résultat vrai pour $n > 0$, alors d'après les deux premiers points de ce lemme,

$$s\left(\bigcup_{j=0}^{n+1} L^j\right) = s\left(\{\epsilon\} \cup \bigcup_{j=1}^{n+1} L^j\right) = \{\epsilon\} \cup s\left(L \cdot \bigcup_{j=0}^n L^j\right) = \{\epsilon\} \cup s(L) \cdot s\left(\bigcup_{j=0}^n L^j\right).$$

D'après l'hypothèse d'induction, $s(\bigcup_{j=0}^n L^j) = \bigcup_{j=0}^n [s(L)]^j$, d'où

$$s\left(\bigcup_{j=0}^{n+1} L^j\right) = \{\epsilon\} \cup s(L) \cdot \bigcup_{j=0}^n [s(L)]^j = \bigcup_{j=0}^{n+1} [s(L)]^j. \quad \blacksquare$$

Nous allons démontrer que la classe des langages réguliers est fermée par les substitutions régulières en passant par les expressions régulières. Pour cela, par abus de langage, nous confonderons les expressions régulières et les langages qu'elles représentent. Ainsi, nous considérerons une substitution régulière comme une fonction associant à chaque lettre dans V une expression régulière sur W , et nous appliquerons cette substitution à des expressions régulières, et non pas à des langages réguliers. Sous ces conventions, nous allons donc démontrer que si E est une expression régulière sur V et s est une substitution régulière, alors $s(E)$ est une expression régulière sur W . Nous traduisons le lemme 5.2.6 à l'aide d'expressions régulières :

Lemme 5.2.7 Soient E, E' des expressions régulières sur V et s une substitution régulière. Alors :

1. $s(E.E') = s(E).s(E')$;
2. $s(E + E') = s(E) + s(E')$;
3. $s(E^*) = [s(E)]^*$.

Nous démontrons maintenant le résultat de fermeture énoncé :

Théorème 5.2.8 La classe des langages réguliers est fermée par substitution régulière.

PREUVE. Nous démontrons par induction structurelle que si E est une expression régulière, alors $s(E)$ est également une expression régulière.

- Si $E = \emptyset$, alors $s(E) = \emptyset$ est une expression régulière.
- Si $E = \varepsilon$ alors $s(E) = \varepsilon$ est une expression régulière.
- Si $E = a \in V$, alors $s(E)$ est une expression régulière.
- Si $E = E_1.E_2$, alors par hypothèse d'induction, $s(E_1)$ et $s(E_2)$ sont des expressions régulières; et d'après le lemme 5.2.7, $s(E) = s(E_1)s(E_2)$ est bien une expression régulière.
- Si $E = E_1 + E_2$, alors par hypothèse d'induction, $s(E_1)$ et $s(E_2)$ sont des expressions régulières; et d'après le lemme 5.2.7, $s(E) = s(E_1) + s(E_2)$ est bien une expression régulière.
- Si $E = E_1^*$, alors par hypothèse d'induction, $s(E_1)$ est une expression régulière; et d'après le lemme 5.2.7, $s(E) = [s(E_1)]^*$ est bien une expression régulière. ■

2.3 Homomorphismes et homomorphismes inverses

Nous démontrons maintenant des résultats de fermeture de la classe des langages réguliers par les homomorphismes et les homomorphismes inverses.

Définition 5.2.9 Un homomorphisme est une substitution qui à toute lettre $a \in V$ associe un singleton dans W^* . Si h est un homomorphisme, et l'image de $a \in V$ par h est le singleton $\{w\} \subseteq W^*$, alors on note $h(a) = w$. ◇

Corollaire 5.2.10 La classe des langages réguliers est fermée par homomorphisme.

PREUVE. Ceci est une conséquence immédiate du théorème 5.2.8. ■

Définition 5.2.11 Etant donnés deux vocabulaires V, W , un langage $L \subseteq W^*$ et un homomorphisme h , l'image par homomorphisme inverse de L est l'ensemble

$$h^{-1}(L) = \{v \in V^* \mid h(v) \in L\}. \quad \diamond$$

Notons qu'aucune hypothèse n'a été faite sur l'injectivité de h ; la définition est valable que h soit injectif ou non.

Théorème 5.2.12 *Si $L \subseteq W^*$ est un langage régulier et h est un homomorphisme, alors $h^{-1}(L)$ est un langage régulier.*

PREUVE. Posons $L' = h^{-1}(L)$ et soit $A = \langle Q, W, \delta_1, q_0, F \rangle$ un automate fini déterministe reconnaissant L ; nous allons construire, à partir de A , un automate fini déterministe qui reconnaît L' .

Soit $A' = \langle Q, V, \delta_2, q_0, F \rangle$, où δ_2 est défini pour tout $q \in Q$ et pour tout $a \in V$ par : $\delta_2(q, a) = \delta_1^*(q, h(a))$. Il est clair que A' est un automate fini déterministe, nous allons montrer qu'il reconnaît exactement L' .

Nous prouvons d'abord par induction structurelle que pour tout $w \in V^*$ et pour tout $q \in Q$, $\delta_2^*(q, w) = \delta_1^*(q, h(w))$. Si $w = \varepsilon$, alors $h(w) = \varepsilon$ et $\delta_2^*(q, w) = \varepsilon = \delta_1^*(q, h(w))$. Supposons maintenant le résultat vrai pour $w \in V^*$, soit $a \in V$ et prouvons que $\delta_2^*(q, aw) = \delta_1^*(q, h(aw))$. Posons $q' = \delta_2(q, a)$; par définition, $q' = \delta_1^*(q, h(a))$, d'où $\delta_2^*(q, aw) = \delta_2^*(q', w)$. Par hypothèse d'induction, $\delta_2^*(q', w) = \delta_1^*(q', h(w)) = \delta_1^*(\delta_1^*(q, h(a)), h(w))$. D'après la propriété 2.4.6, $\delta_2^*(q', w) = \delta_1^*(q, h(a)h(w)) = \delta_1^*(q, h(aw))$.

Nous pouvons maintenant caractériser le langage reconnu par A' grâce aux équivalences :

$$w \in L(A') \Leftrightarrow \delta_2^*(q_0, w) \in F \Leftrightarrow \delta_1^*(q_0, h(w)) \in F \Leftrightarrow h(w) \in L(A),$$

donc $w \in L(A')$ si et seulement si $w \in h^{-1}(L)$. ■

Exemple 5.2.13 Reprenons les langages de l'exemple 5.1.5, c'est-à-dire $L = \{a^m b^m \mid m \geq 0\}$ et $M = \{c\}^+ . L \cup \{a, b\}^*$. Nous savons que le langage L n'est pas régulier, prouvons maintenant par contradiction que M n'est pas régulier non plus. Supposons que M est régulier, et considérons langage $M' = M \cap \{c\}^+ \{a, b\}^* = \{c\}^+ . L$. Il est clair que le langage $\{c\}^+ \{a, b\}^*$ est régulier, donc, comme la classe des langages réguliers est fermée par intersection (Corollaire 5.2.3), M' doit être régulier. Considérons maintenant l'homomorphisme $h : V \rightarrow V \cup \{\varepsilon\}$ tel que $h(a) = a$, $h(b) = b$ et $h(c) = \varepsilon$. Comme les langages réguliers sont stables par homomorphisme, $h(M) = L$ doit être régulier, ce qui est impossible. Donc, l'hypothèse de départ est fautive, et M n'est pas régulier. ♣

Chapitre 6

Systemes de réécriture, grammaires

Les chapitres précédents étaient consacrés à l'étude de langages formels via des machines capables de *reconnaître* des éléments du langage. De façon duale, nous nous intéressons dans la suite à des systèmes capables d'*engendrer* les éléments d'un langage. Ces éléments sont engendrés grâce à la réécriture de symboles, ce que nous définissons formellement ci-dessous, puis nous présentons le formalisme des grammaires, qui permettent d'engendrer les langages formels.

1 Systemes de réécriture

1.1 Définitions

Définition 6.1.1 On appelle *système de réécriture (de mots)* un couple $S = \langle V, R \rangle$ où :

- V est un vocabulaire,
- R est un ensemble fini de couples de chaînes sur V .

Tout élément (α, β) de R est appelé *règle de réécriture* et est noté $\alpha \rightarrow \beta$; α est la *partie gauche*, β la *partie droite* de la règle $\alpha \rightarrow \beta$. Des règles $\alpha \rightarrow \beta_1, \alpha \rightarrow \beta_2, \dots, \alpha \rightarrow \beta_k$ ayant même partie gauche sont souvent notées $\alpha \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_k$.

Soient un système de réécriture $S = \langle V, R \rangle$ et deux chaînes x et y de V^* . S'il existe une règle $\alpha \rightarrow \beta$ dans R telle que $x = u\alpha v, y = u\beta v$, alors on dit que x se réécrit *directement* en y , ce qu'on note $x \Rightarrow_S y$, ou $x \Rightarrow_R y$, ou encore, s'il n'y a pas d'ambiguïté sur le système de réécriture utilisé, $x \Rightarrow y$. \diamond

On dit encore que x se réécrit en y par application de la règle $\alpha \rightarrow \beta$. La partie gauche α de la règle appliquée est une sous-chaîne de x , et y est obtenue à partir de x en remplaçant une occurrence de α dans x par la partie droite β .

Si $x \Rightarrow_R y$, il est possible que plusieurs règles de R permettent de réécrire x en y . Par exemple, soit $R = \{ab \rightarrow ba, aab \rightarrow aba\}$: on a $aab \Rightarrow_R aba$, par application de la première ou de la deuxième règle de R . Il est aussi possible qu'une même règle puisse s'appliquer à différentes

sous-chaînes de x pour donner y ; soit par exemple la règle $a \rightarrow aa$: la chaîne aaa peut être obtenue en réécrivant la première ou la deuxième lettre de la chaîne aa .

La relation \Rightarrow associée à un système de réécriture $\langle V, R \rangle$ est une relation sur V^* , c'est-à-dire une partie de $V^* \times V^*$. On définit comme pour toute relation (voir paragraphe 1 du chapitre 1) les relations \Rightarrow^n , pour n entier positif ou nul, \Rightarrow^* et \Rightarrow^+ . Pour tout x, y de V^* , on a :

- $x \Rightarrow^n y$ signifie : x se réécrit en y par application de n règles (en particulier, $x \Rightarrow^0 y$ si et seulement si $x = y$)
- $x \Rightarrow^* y$ signifie : x se réécrit en y (par application d'un nombre quelconque, éventuellement nul, de règles)
- $x \Rightarrow^+ y$ signifie : x se réécrit y par application d'au moins une règle.

Définition 6.1.2 On appelle *dérivation dans un système de réécriture* toute suite finie x_1, x_2, \dots, x_k de chaînes telles que $k \geq 1$ et $x_1 \Rightarrow x_2 \Rightarrow \dots \Rightarrow x_k$. On dit que x_k est dérivée à partir de x_1 : x_1 est l'origine, x_k est l'extrémité de la dérivation. La *longueur* de la dérivation est égale à $k - 1$, c'est-à-dire au nombre d'applications de règles de réécriture dans la dérivation. Les chaînes x_i sont les éléments de la dérivation. \diamond

On a évidemment :

- $x \Rightarrow^* y$ si et seulement s'il existe une dérivation d'origine x et d'extrémité y .
- $x \Rightarrow^n y$ si et seulement s'il existe une dérivation de longueur n de x à y .

La donnée d'une dérivation ne permet pas toujours de retrouver les règles de réécriture appliquées, ni les sous-chaînes réécrites (car la relation \Rightarrow ne le permet pas toujours). On introduit maintenant la notion de dérivation indicée, qui décrit complètement le processus de réécriture.

Définition 6.1.3 Etant donné un entier $i \geq 1$ et une règle r , on note $\Rightarrow_{i,r}$ la relation sur V^* définie par : $x \Rightarrow_{i,r} y$ si et seulement si

- $x = a_1 \dots a_n$, avec $a_1, \dots, a_n \in V$,
- $r = a_i \dots a_k \rightarrow \omega$, $1 \leq i \leq n + 1$, $i - 1 \leq k \leq n$
- $y = a_1 \dots a_{i-1} \omega a_{k+1} \dots a_n$.

L'entier i désigne le premier caractère de la sous-chaîne réécrite de x , et r la règle appliquée. On a $k = i - 1$ quand la partie gauche de la règle r est la chaîne vide et $i = n + 1$ quand la chaîne vide est réécrite en ω à droite de la chaîne x .

Une *dérivation indicée* (pour un ensemble de règles de réécriture R) est une suite finie $x_1, \langle i_1, r_1 \rangle, x_2, \langle i_2, r_2 \rangle, \dots, x_{k-1}, \langle i_{k-1}, r_{k-1} \rangle, x_k$, telle que :

- $k \geq 1$, $x_1, \dots, x_k \in V^*$,
- r_1, \dots, r_{k-1} appartiennent à R ,
- $x_1 \Rightarrow_{i_1, r_1} x_2 \dots x_{k-1} \Rightarrow_{i_{k-1}, r_{k-1}} x_k$

\diamond

Comme ci-dessus, la dérivation est d'origine x_1 , d'extrémité x_k et de longueur $k - 1$.

On associe ainsi à un système de réécriture quatre relations de dérivation :

- $\Delta_1 : x \Rightarrow^* y$ (il existe une dérivation de x à y)
- $\Delta_2 : x \Rightarrow^n y$ (il existe une dérivation de x à y de longueur n)
- $\Delta_3 : x, x_2, \dots, x_n, y$ est une dérivation de x à y

— $\Delta_4 : x, \langle i_1, r_1 \rangle, x_2, \langle i_2, r_2 \rangle, \dots, x_n, \langle i_n, r_n \rangle, y$ est une dérivation indicée de x à y .

On a les propriétés suivantes :

- Δ_1 s'écrit $\exists n$ tel que Δ_2
- Δ_2 s'écrit $\exists x_2, \dots, x_n$ tels que Δ_3
- Δ_3 s'écrit $\exists i_1, r_1, \dots, i_n, r_n$ tels que Δ_4

Exemples de systèmes de réécriture

1. $V_1 = \{a, b, 0, 1, r, s\}, R_1 = \{ra \rightarrow s, rb \rightarrow s, sa \rightarrow s, sb \rightarrow s, s0 \rightarrow s, s1 \rightarrow s\}$.

On vérifie facilement que pour tout w dans $\{a, b, 0, 1\}^*$, on a $rw \Rightarrow^* s$ si et seulement si le premier caractère de w est a ou b . Ce système de réécriture reconnaît ainsi les identificateurs sur le vocabulaire $\{a, b, 0, 1\}$.

2. $V_2 = \{0, 1, +, E\}, R_2 = \{E \rightarrow 0 \mid 1 \mid E + E\}$.

Pour tout w appartenant à $\{0, 1, +\}^*$, on a $E \Rightarrow^* w$ si et seulement si w est une expression construite avec les opérands 0 et 1 et l'opérateur $+$.

3. $V_3 = \{0, 1, +\}, R_3 = \{0 + 0 \rightarrow 0, 0 + 1 \rightarrow 1, 1 + 0 \rightarrow 1, 1 + 1 \rightarrow 0\}$.

Ce système de réécriture calcule la somme modulo 2 d'une suite d'entiers binaires en ce sens que si w appartenant à $\{0, 1, +\}^*$ est expression dérivée à partir de E dans le système de l'exemple précédent, on a dans R_3 la dérivation $w \Rightarrow^* 0$ ou $w \Rightarrow^* 1$ suivant la valeur de l'expression quand l'opérateur $+$ est interprété comme l'addition modulo 2.

4. $V_4 = \{0, 1\}, R_4 = \{10 \rightarrow 01\}$.

Soit $x \in \{0, 1\}^*$ et soit y telle que a) $x \Rightarrow^* y$ et b) la règle de R_4 ne s'applique pas à y (en d'autres termes : y ne contient pas d'occurrence de 10). Alors y est la permutation des lettres de x où tous les 0 sont avant les 1 . Ce système de réécriture trie donc les chaînes de $\{0, 1\}^*$ par chiffres croissants.

Proposition 6.1.4 (Composition des dérivations) Soit $S = \langle V, R \rangle$ un système de réécriture, et considérons $u_1, \dots, u_n, v_1, \dots, v_n$ des chaînes de V^* . Si $u_1 \Rightarrow^* v_1, \dots, u_n \Rightarrow^* v_n$, alors $u_1 \dots u_n \Rightarrow^* v_1 \dots v_n$.

PREUVE. Cette propriété se démontre comme suit :

1. si $u_1 \Rightarrow^* v_1$, on a alors $u_1 u_2 \Rightarrow^* v_1 u_2$ (démonstration facile par récurrence sur la longueur de la dérivation).
2. On a de même : si $u_2 \Rightarrow^* v_2$, alors $v_1 u_2 \Rightarrow^* v_1 v_2$.
3. On en conclut que si $u_1 \Rightarrow^* v_1$ et $u_2 \Rightarrow^* v_2$, alors $u_1 u_2 \Rightarrow^* v_1 v_2$.

La généralisation au cas de n dérivations est immédiate. ■

Nous énonçons la propriété avec la relation de dérivation Δ_2 (c'est-à-dire \Rightarrow^n) :

Proposition 6.1.5 Pour tout $u_1, \dots, u_n, v_1, \dots, v_n$ appartenant à V^* , pour tout q_1, \dots, q_n appartenant à \mathbb{N} , si $u_1 \Rightarrow^{q_1} v_1, \dots, u_n \Rightarrow^{q_n} v_n$ alors $u_1 \dots u_n \Rightarrow^p v_1 \dots v_n$, avec $p = \sum_{i=1}^n q_i$.

Exercice 6.1.6 *Enoncer la propriété de composition pour les relations de dérivation Δ_3 et Δ_4 (c'est-à-dire pour les dérivations et les dérivations indicées). La dérivation à partir de $u_1 \dots u_n$ qui engendre $v_1 \dots v_n$ est-elle unique ?*

2 Grammaires

2.1 Définitions

Définition 6.2.1 On appelle *grammaire* tout quadruplet $G = \langle V_T, V_N, S, R \rangle$ où

- V_T et V_N sont deux vocabulaires disjoints. V_T est le *vocabulaire terminal* (ses éléments sont appelés *symboles terminaux*), V_N est le *vocabulaire non-terminal* (ses éléments sont les *symboles non-terminaux*). Les chaînes sur V_T sont dites *chaînes terminales*.

On pose $V = V_T \cup V_N$; V est appelé le *vocabulaire* de la grammaire.

- S est un élément de V_N , l'*axiome* de la grammaire.

- $\langle V, R \rangle$ est un système de réécriture. R est l'ensemble des règles de la grammaire. \diamond

Définition 6.2.2 On appelle *langage engendré* par une grammaire $G = \langle V_T, V_N, S, R \rangle$ l'ensemble des chaînes sur le vocabulaire terminal V_T que l'on peut dériver à partir de l'axiome S . On note $L(G)$ le langage engendré par la grammaire G . On a donc : $L(G) = \{x \in V_T^* \mid S \Rightarrow_R^* x\}$. \diamond

Des exemples de grammaires sont donnés au paragraphe 2.4.

Définition 6.2.3 Deux grammaires sont dites *équivalentes* si elles engendrent le même langage. \diamond

En imposant des restrictions sur la forme des règles, on définit différents types de grammaires :

Grammaires générales (de type 0) Une grammaire est dite *générale* si toutes ses règles sont de la forme $\alpha \rightarrow \beta$, avec $\alpha, \beta \in V^*$, et $\alpha \neq \varepsilon$.

Grammaires sous-contexte (de type 1) Une grammaire est dite *sous-contexte* si toutes ses règles sont de la forme $\alpha A \beta \rightarrow \alpha \omega \beta$, avec $\alpha, \beta \in V^*$, $A \in V_N$, $\omega \in V^+$. On autorise parfois d'ajouter la règle $S \rightarrow \varepsilon$, à condition que l'axiome S n'apparaisse dans aucune partie droite de règle. Ceci permet à des grammaires sous-contexte d'engendrer le mot vide.

Grammaires hors-contexte (de type 2) Une grammaire est dite *hors-contexte* si toutes ses règles sont de la forme $A \rightarrow \omega$, avec $A \in V_N$, $\omega \in V^*$. On appelle ε -*règle* une règle de la forme $A \rightarrow \varepsilon$, et *1-règle* une règle de la forme $A \rightarrow B$ avec $B \in V_N$.

Les grammaires hors-contexte sont souvent appelées *formes de Backus*. Elles sont équivalentes aux cartes syntaxiques couramment utilisées pour définir la syntaxe des langages de programmation.

Grammaires linéaires à droite, grammaires linéaires à gauche (de type 3) Une grammaire est dite *linéaire à droite* si toutes ses règles sont de l'une des deux formes suivantes :

— $A \rightarrow \omega B$, avec $A, B \in V_N$, $\omega \in V_T^*$,

— $A \rightarrow \omega$, avec $A \in V_N$, $\omega \in V_T^*$.

Une grammaire est dite *linéaire à gauche* si toutes ses règles sont de l'une des deux formes suivantes :

— $A \rightarrow B\omega$, avec $A, B \in V_N$, $\omega \in V_T^*$,

— $A \rightarrow \omega$, avec $A \in V_N$, $\omega \in V_T^*$.

Ces grammaires sont également appelées *grammaires régulières*.

2.2 Une définition équivalente des grammaires sous-contexte

Il existe une autre formulation, qui définit les grammaires sous-contexte comme étant constituée de règles de la forme $\alpha \rightarrow \beta$, où $|\alpha| \leq |\beta|$. Appelons les grammaires dont toutes les règles sont de cette forme des grammaires *non-contractantes*. Il est clair que toute grammaire sous-contexte (dont les règles sont de la forme $\alpha A \beta \rightarrow \alpha \omega \beta$, avec $\alpha, \beta \in V^*$, $A \in V_N$, $\omega \in V^+$) est une grammaire non-contractante. Cependant, les grammaires non-contractantes ne sont pas plus générales que les grammaires sous-contexte : nous montrons comment transformer une telle grammaire en une grammaire sous-contexte.

Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire non-contractante. On définit une grammaire sous-contexte $G' = \langle V_T, V'_N, S, R' \rangle$ équivalente à G de la façon suivante. Pour chaque règle $r \in R$, r étant de la forme $\alpha_1 \cdots \alpha_m \rightarrow \beta_1 \cdots \beta_n$ où $\alpha_i, \beta_i \in V$ et $m \leq n$, on effectue les opérations suivantes :

1. On remplace chaque symbole terminal $a \in V_T$ apparaissant dans r par un nouveau non-terminal \bar{a} , et on ajoute la règle $\bar{a} \rightarrow a$ à R' . Par exemple, la règle $S \rightarrow Abc$ est transformée en $S \rightarrow A\bar{b}\bar{c}$. La règle ainsi obtenue est ajoutée à R' , notons-la $A_1 \cdots A_m \rightarrow B_1 \cdots B_n$.
2. Soient C_1^r, \dots, C_m^r de nouveaux symboles non-terminaux n'apparaissant nulle part ailleurs. On ajoute à R' les $2m$ règles suivantes :

$$\begin{aligned}
 A_1 A_2 \cdots A_m &\rightarrow C_1^r A_2 \cdots A_m, \\
 C_1^r A_2 \cdots A_m &\rightarrow C_1^r C_2^r \cdots A_m, \\
 &\vdots \\
 C_1^r \cdots A_{m-1} A_m &\rightarrow C_1^r \cdots C_{m-1}^r A_m, \\
 C_1^r \cdots C_{m-1}^r A_m &\rightarrow C_1^r \cdots C_m^r B_{m+1} \cdots B_n, \\
 C_1^r \cdots C_m^r B_{m+1} \cdots B_n &\rightarrow B_1 C_2^r \cdots C_m^r B_{m+1} \cdots B_n \\
 &\vdots \\
 B_1 \cdots B_{m-1} C_m^r B_{m+1} \cdots B_n &\rightarrow B_1 \cdots B_n.
 \end{aligned}$$

On obtient ainsi une grammaire sous-contexte équivalente à G .

2.3 Quelques propriétés

- Toute grammaire linéaire à droite, ainsi que toute grammaire linéaire à gauche est une grammaire hors-contexte.

Toute grammaire hors-contexte ne contenant pas d' ϵ -règle est une grammaire sous-contexte.

- Un langage est dit *linéaire à droite* (*hors-contexte, sous-contexte*) s'il existe une grammaire linéaire à droite (*hors-contexte, sous-contexte*) qui l'engendre.
- Pour tout langage régulier (c'est-à-dire défini par une expression régulière - voir le paragraphe 1 du chapitre 1), il existe une grammaire linéaire à droite et une grammaire linéaire à gauche qui l'engendent. Les langages réguliers sont aussi définis les automates finis (voir le chapitre 2). Tout langage régulier est hors-contexte (puisque toute grammaire linéaire à droite est hors-contexte); la réciproque est fautive : par exemple le langage $\{a^n b^n \mid n \geq 1\}$ est hors-contexte mais n'est pas régulier.
- Tout langage hors-contexte ne contenant pas la chaîne vide est sous-contexte (en effet, tout langage hors-contexte ne contenant pas ϵ peut être engendré par une grammaire hors-contexte sans ϵ -règle. La réciproque est fautive : par exemple le langage $\{a^n b^n c^n \mid n \geq 1\}$ est sous-contexte mais n'est pas hors-contexte.
- Les langages engendrés par les grammaires générales (c'est-à-dire sans restriction sur les règles) sont dits récursivement énumérables : à partir d'une grammaire, on peut construire un algorithme qui énumère toutes les chaînes du langage engendré par la grammaire; réciproquement, étant donné un algorithme énumérant les chaînes d'un langage, on peut construire une grammaire engendrant ce langage.
- On appelle langage récursif tout langage pour lequel il existe un algorithme de reconnaissance (c'est-à-dire un algorithme qui, pour toute chaîne w sur le vocabulaire du langage, décide si w appartient au langage ou non). On peut montrer que tout langage récursif est récursivement énumérable, et que la réciproque est fautive. Il existe d'autre part des langages qui ne sont pas récursivement énumérables. Les langages sous-contexte (et a fortiori hors-contexte et réguliers) sont récursifs.

Exercice 6.2.4

1. Définir un algorithme qui étant donné une grammaire quelconque, énumère les chaînes du langage engendré par cette grammaire.
2. Dédire de l'algorithme ci-dessus que pour tout langage récursivement énumérable L il existe un algorithme de reconnaissance partielle (c'est-à-dire un algorithme qui reconnaît toute chaîne w de L , mais ne s'arrête pas si w n'appartient pas à L).
3. Prouver que tout langage engendré par une grammaire dont les règles satisfont la contrainte « $\alpha \rightarrow \beta$ implique $|\alpha| \leq |\beta|$ » est récursif.
On peut montrer que ces grammaires sont équivalentes aux grammaires sous-contexte.

Problèmes décidables - problèmes indécidables

Étant donné une grammaire G (linéaire à droite, hors-contexte, etc.), on s'intéressera par la suite à des problèmes du genre :

- $L(G)$ est-il vide? (problème du langage vide)
- $L(G)$ est-il infini? (problème du langage infini)

- $L(G) = V_T^*$?
- $L(G)$ est-il un langage régulier ? , etc.

On dit qu'un de ces problèmes est décidable pour les grammaires linéaires à droite (hors-contexte, etc.) s'il existe un algorithme qui pour toute grammaire linéaire à droite (hors-contexte, etc.) calcule la réponse (oui ou non) au problème posé ; il est dit indécidable dans le cas contraire. Par exemple tous les problèmes énoncés ci-dessus sont décidables pour les grammaires régulières (la réponse au quatrième problème est toujours oui) ; les problèmes du langage vide et du langage infini sont décidables pour les grammaires hors-contexte ; les deux autres problèmes sont indécidables pour cette classe de grammaires.

D'autres problèmes importants sont :

- le problème du mot : étant donné une grammaire G et une chaîne terminale x , est-ce que x appartient à $L(G)$?
- le problème de l'équivalence : étant données deux grammaires G_1 et G_2 , est-ce que $L(G_1) = L(G_2)$?

Le problème du mot est décidable pour les grammaires sous-contexte (voir l'exercice 6.2.4 ci-dessus), pour les grammaires hors-contexte (voir le chapitre 7) et donc pour les grammaires linéaires à droite qui sont des grammaires hors-contexte particulières ; il est indécidable pour les grammaires générales . Si le problème du mot est décidable pour une classe de grammaires, les langages engendrés par les grammaires de cette classe sont rékursifs. Le problème de l'équivalence est indécidable pour les grammaires hors-contexte ; il est décidable pour les grammaires linéaires à droite (voir le chapitre 3).

Choix d'un formalisme (classe de définitions) pour définir un langage de programmation

Les types de grammaires introduits au paragraphe 2.1 constituent différents formalismes que l'on peut utiliser pour définir la syntaxe d'un langage de programmation. Pour choisir parmi ces formalismes on est guidé par les critères suivants :

1. Le formalisme doit être suffisamment puissant pour contenir une définition du langage.
2. Le formalisme doit être suffisamment simple pour que chaque définition puisse être compilée en un algorithme efficace de reconnaissance des programmes du langage défini.
3. Le formalisme doit contenir une définition du langage concise et facile à comprendre.

Le critère 1) conduit à exclure le formalisme des grammaires régulières, qui ne permet pas de décrire les parenthésages à une profondeur arbitrairement grande que l'on rencontre dans les langages de programmation (expressions avec parenthèses, structure de bloc, etc.). Les grammaires régulières (ou des formalismes équivalents) sont par contre souvent utilisées pour définir la lexicographie des langages de programmation (c'est-à-dire les identificateurs et les constantes).

Le critère 2) conduit à exclure les grammaires générales (qui peuvent engendrer des langages non rékursifs) et les grammaires sous-contexte, auxquelles on ne peut pas associer des algorithmes de reconnaissance efficaces.

C'est pourquoi la plupart des langages de programmation sont définis par des grammaires hors-contexte (ou des formalismes équivalents comme celui des diagrammes syntaxiques). Le critère

1) est alors à peu près satisfait (mais pas complètement ; par exemple le traitement des déclarations dans un langage de programmation ne peut être décrit par une grammaire hors-contexte. Le critère 2) est lui aussi à peu près satisfait (on utilise souvent des grammaires hors-contexte particulières admettant des algorithmes de reconnaissance plus efficaces, comme les grammaires LL(1) ou LR(1)). Enfin définir un langage avec une grammaire hors-contexte revient à en donner une définition par récurrence qui satisfait souvent le critère 3).

2.4 Exemples

Définition des nombres réels sans signe d'un langage de programmation avec une grammaire hors-contexte

- $V_T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, E, .\}$
- $V_N = \{R, D, N, F\}$
- Axiome : R
- Règles :
 - $R \rightarrow D|DF|NF$
 - $D \rightarrow N.N$
 - $F \rightarrow EN|ESN$
 - $N \rightarrow C|NC$
 - $S \rightarrow +|-$
 - $C \rightarrow 0|1|2|3|4|5|6|7|8|9$

Exercice 6.2.5

1. Montrer que $R \Rightarrow^* 2.3E - 6$
2. Trouver une grammaire régulière équivalente à la grammaire donnée.

Langage $\{a^n b^n | n \geq 1\}$

Soit la grammaire hors-contexte $G_1 = \langle \{a, b\}, \{S\}, S, \{S \rightarrow aSb|ab\} \rangle$.
Il est facile de montrer que $L(G_1) = \{a^n b^n | n \geq 1\}$.

Langage des parenthèses des expressions

Soit la grammaire hors-contexte $G_2 = \langle \{a, b\}, \{S\}, S, \{S \rightarrow aSb|SS|\epsilon\} \rangle$. Les symboles a et b représentant respectivement le début et la fin d'un bloc, le langage engendré par G_2 décrit toutes les structures de blocs possibles. La grammaire G_2 est équivalente à la grammaire $G_3 = \langle \{a, b\}, \{S\}, S, \{S \rightarrow SaSb|\epsilon\} \rangle$ et à la grammaire $G_4 = \langle \{a, b\}, \{S\}, S, \{S \rightarrow aSbS|\epsilon\} \rangle$.

Grammaire engendrant les expressions régulières sur $\{a, b\}$

- $V_T = \{a, b, \emptyset, +, \cdot, *, (,)\}$
- $V_N = \{E, T, F, X\}$

- Axiome : E
- Règles :
 - $E \rightarrow E + T | T$
 - $T \rightarrow T.F | F$
 - $F \rightarrow X^* | X$
 - $X \rightarrow \emptyset | a | b | (E)$

Langage $\{a^n b^n c^n | n \geq 1\}$

Soit la grammaire $G_5 = \langle \{a, b, c\}, \{S, B, C\}, S, R \rangle$, où l'ensemble des règles R est :

- $S \rightarrow aSBC | aBC$
- $CB \rightarrow BC$
- $aB \rightarrow ab$
- $bB \rightarrow bb$
- $bC \rightarrow bc$
- $cC \rightarrow cc$

L'ensemble R satisfait la contrainte suivante : pour tout $\alpha \rightarrow \beta$ appartenant à R , $0 < |\alpha| \leq |\beta|$. On peut montrer que les grammaires satisfaisant cette contrainte sont équivalentes aux grammaires sous-contexte définies au paragraphe 2.1 (Exercice 6.2.10). La grammaire G_5 engendre le langage $\{a^n b^n c^n | n \geq 1\}$. Il est possible de démontrer que ce langage n'est pas hors-contexte.

2.5 Exercices

Exercice 6.2.6 Construire des grammaires hors-contexte pour les langages suivants

1. $\{ww^w | w \in \{a, b\}^*\}$ (w^w désigne le mot miroir de w : si $w = w_1 \cdots w_n$ alors $w^w = w_n \cdots w_1$)
2. $\{a^i b^j | i \neq j \text{ et } i, j \geq 0\}$

Exercice 6.2.7

1. On définit l'ensemble des formules de la logique propositionnelle comme l'ensemble des expressions construites avec l'opérateur \neg (non), les opérateurs binaires \vee (ou), \wedge (et), \rightarrow (implique), et des opérandes qui sont des symboles de propositions atomiques. On utilise l'ensemble des suites de lettres sur l'alphabet $\{A, B, \dots, Z\}$ pour noter les propositions atomiques.

Exemples : $\neg A \vee B$, $A \rightarrow (B \rightarrow A)$, $AA \wedge B \vee \neg AA \wedge C$, $(ABA \rightarrow BB) \wedge (ABA \rightarrow \neg C)$, ... sont des formules de la logique propositionnelle.

Donner une grammaire hors-contexte engendrant l'ensemble des formules de la logique propositionnelle.

2. Soit $F = \{a, b, f, g, h\}$, où a et b sont deux symboles de constantes, f et g deux symboles de fonctions unaires, h un symbole de fonction binaire. L'ensemble des termes sur F est l'ensemble des expressions que l'on peut construire avec les éléments de F .

Exemple : $h(a, b)$, $f(f(b))$, $h(f(a), h(b, b))$, ... sont des termes.

Donner une grammaire hors-contexte engendrant l'ensemble des termes sur F .

3. Soit $G = \{=, <\}$, où $=$ et $<$ sont deux symboles de prédicats binaires. L'ensemble des propositions atomiques sur $F \cup G$ est l'ensemble des expressions de la forme $t_1 = t_2$ ou $t_1 < t_2$, où t_1 et t_2 sont des termes sur F .

Donner une grammaire hors-contexte engendrant l'ensemble des propositions atomiques sur $F \cup G$.

4. Soit X un ensemble infini de symboles de variables. On pose par exemple $X = \{x, y, z\}^+$. L'ensemble des termes sur $F \cup X$, avec F défini en 2, est alors l'ensemble des expressions construites avec des opérandes pris dans $\{a, b\} \cup X$, et des opérateurs pris dans $\{f, g, h\}$.

L'ensemble des formules atomiques sur $F \cup G \cup X$ est l'ensemble des expressions de la forme $t_1 = t_2$ ou $t_1 < t_2$, où t_1 et t_2 sont des termes sur $F \cup X$.

L'ensemble des formules de la logique des prédicats sur $F \cup G \cup X$ est l'ensemble des expressions construites avec les formules atomiques sur $F \cup G \cup X$, les opérateurs de la logique propositionnelle $\neg, \vee, \wedge, \rightarrow$ et les quantificateurs \forall et \exists .

Exemples : $(\forall x f(x) = f(x))$, $(\exists x f(x) = g(x) \rightarrow (\forall x f(x) = g(x)))$, $(\forall x f(x) = f(y))$, $f(x) < g(y) \vee (\exists z f(z) < f(x))$, $(\forall x \forall y \forall z x < y \wedge y < z \rightarrow x < z)$, ... sont des formules de la logique des prédicats.

Donner une grammaire hors-contexte engendrant l'ensemble des formules de la logique des prédicats.

Exercice 6.2.8

Soit $G = \langle V_T, V_N, S, R \rangle$ la grammaire définie par :

$V_T = \{a, b, c\}$, $V_N = \{S, A, B\}$, R est l'ensemble de règles

$$S \rightarrow aSA | bSB | c$$

$$cA \rightarrow ca$$

$$cB \rightarrow cb$$

$$aA \rightarrow Aa$$

$$aB \rightarrow Ba$$

$$bA \rightarrow Ab$$

$$bB \rightarrow Bb$$

1. Quel est le type de cette grammaire ?
2. Donner une dérivation de la chaîne $abcab$.
3. Prouver que $L(G) = \{wcw \mid w \in \{a, b\}^*\}$.

Exercice 6.2.9 Décrire les langages engendrés par les grammaires suivantes, où S est l'axiome et les non-terminaux sont les lettres majuscules apparaissant dans les règles :

1. Grammaire hors-contexte :

$$S \rightarrow bSS | a$$

2. Grammaire sous-contexte :

$$S \rightarrow \#a\#$$

$$\#a \rightarrow \#B$$

$$Ba \rightarrow aaB$$

$$B\# \rightarrow aa\#$$

Exercice 6.2.10 *Montrer qu'un langage est sous-contexte si et seulement s'il est engendré par une grammaire dont toutes les règles $\alpha \rightarrow \beta$ sont telles que $0 < |\alpha| \leq |\beta|$.*

Exercice 6.2.11 *Montrer qu'un langage ne contenant pas la chaîne vide est engendré par une grammaire linéaire à droite si et seulement s'il est engendré par une grammaire dont toutes les règles sont de l'une des deux formes suivantes :*

- $A \rightarrow aB$, avec $A, B \in V_N$, $a \in V_T$
- $A \rightarrow a$, avec $A \in V_N$, $a \in V_T$

Chapitre 7

Grammaires hors-contexte

1 Propriété fondamentale (décomposition des dérivations hors-contexte)

La propriété 6.1.4 du chapitre 6 signifie que l'on peut composer des dérivations d'un système de réécriture; la réciproque de cette propriété, qui est fautive pour les systèmes de réécriture généraux, est vraie pour les grammaires hors-contexte : on peut décomposer une dérivation hors-contexte en sous-dérivations indépendantes. Cette propriété fondamentale des dérivations hors-contexte est très souvent utilisée dans les démonstrations.

Théorème 7.1.1 Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte; on pose $V = V_T \cup V_N$. Soit une dérivation $u_1 \cdots u_n \Rightarrow^p w$, avec $u_i \in V^*$ pour $i = 1, \dots, n$ et $w \in V^*$.

Alors pour $i = 1, \dots, n$, il existe $v_i \in V^*$ et $q_i \in \mathbb{N}$ tels que :

- $w = v_1 \cdots v_n$,
- pour $i = 1, \dots, n$, $u_i \Rightarrow^{q_i} v_i$,
- $p = \sum_{i=1}^n q_i$.

PREUVE. On procède par induction sur p .

- Si $p = 0$, alors le théorème est vrai.
- Supposons que le théorème soit vrai pour toute dérivation de longueur p , et considérons une dérivation $u_1 \dots u_n \Rightarrow^{p+1} w$. Puisque les parties gauches des règles d'une grammaire hors-contexte sont de longueur 1, la première règle de la dérivation considérée est appliquée à un u_i . Il existe donc u_i et $t_i \in V^*$ tels que :

$$u_1 \cdots u_{i-1} u_i u_{i+1} \cdots u_n \Rightarrow u_1 \cdots u_{i-1} t_i u_{i+1} \cdots u_n \Rightarrow^p w.$$

On applique l'hypothèse d'induction à la sous-dérivation de longueur p : il existe des éléments v_1, \dots, v_n dans V^* et des entiers q_1, \dots, q_n tels que :

- $w = v_1 \cdots v_n$,
- pour k de 1 à $i-1$ et de $i+1$ à n , $u_k \Rightarrow^{q_k} v_k$,
- $t_i \Rightarrow^{q_i} v_i$,

$$— p = \sum_{j=1}^n q_j.$$

Puisque $u_i \Rightarrow^{1+q_i} v_i$, le théorème est vérifié pour la dérivation $u_1 \dots u_n \Rightarrow^{1+p} w$. ■

La propriété fondamentale a été énoncée pour la relation de dérivation Δ_2 (c'est-à-dire la relation \Rightarrow^p). Nous énonçons ci-dessous comme corollaire la propriété fondamentale pour la relation \Rightarrow^* (relation de dérivation Δ_1).

Corollaire 7.1.2 *Si $u_1 \dots u_n \Rightarrow^* w$ dans une grammaire hors-contexte, alors il existe v_1, \dots, v_n tels que $w = v_1 \dots v_n$ et pour tout $i = 1, \dots, n$, $u_i \Rightarrow^* v_i$.*

Nous étudierons au paragraphe 2 la propriété fondamentale pour les relations de dérivation Δ_3 et Δ_4 .

Remarque Dans la suite de ce chapitre, sauf exceptions signalées, nous ne parlerons que des grammaires hors-contextes en utilisant implicitement les notations du théorème ci-dessus.

Exercice 7.1.3

1. Donner un contre-exemple montrant que la propriété fondamentale est fausse pour une grammaire qui n'est pas hors-contexte.
2. Soit $\langle V, R \rangle$ un système de réécriture tel que toutes ses dérivations satisfont la propriété fondamentale. Prouver que toutes ses règles sont de la forme $a \rightarrow \omega$, avec a appartenant à V et ω appartenant à V^* . Ainsi la propriété fondamentale caractérise les systèmes de réécriture dont toutes les règles ont une partie gauche de longueur 1.

2 Dérivations indicées dans une grammaire hors-contexte

La notion de dérivation indicée a été introduite au chapitre 6, définition 6.1.3. Rappelons que si $x \Rightarrow_R y$ dans un système de réécriture R , il est possible que

1. plusieurs règles de R permettent de réécrire x en y ,
2. une même règle de R puisse s'appliquer à différentes sous-chaînes de x pour donner y .

On a levé ces ambiguïtés en utilisant une relation $\Rightarrow_{i,r}$ où i est le rang du premier caractère de la sous-chaîne de x qui est réécrite, et r est la règle utilisée.

Quand R est un ensemble de règles hors-contexte il suffit d'indiquer le rang du caractère de x qui est dérivé : la donnée de la chaîne de départ x , de la chaîne obtenue y et de l'indice i du caractère réécrit permet de retrouver la règle r appliquée.

Exercice 7.2.1 *Montrer que si R est un système de réécriture arbitraire, alors la donnée de la chaîne de départ x , de la chaîne obtenue y et de l'indice i du caractère réécrit peuvent ne pas suffire à retrouver la règle appliquée pour réécrire x en y .*

Donner une condition nécessaire et suffisante sur R pour que la donnée de x, i, y permette de retrouver la règle appliquée pour réécrire x en y .

Dans la suite, nous simplifions les notations introduites dans la définition 6.1.3, en ne précisant dans les dérivations indicées, que la position du symbole dérivé. Nous définissons l'opération de concaténation de dérivation indicées.

Définition 7.2.2 Soient D et D' deux dérivations indicées telles que l'extrémité de D et l'origine de D' coïncident. On note $D.D'$ la dérivation indicée qui est la concaténation de D et D' . C'est une dérivation indicée dont l'origine est l'origine de D et dont l'extrémité est l'extrémité de D' . \diamond

2.1 Dérivations indicées gauches

Définition 7.2.3 (Dérivation indicée gauche) Soient $x_1, \dots, x_{k+1} \in V^*$ et $i_1, \dots, i_k \in \mathbb{N}$. Une dérivation indicée dans une grammaire hors-contexte $x_1, i_1, \dots, x_k, i_k, x_{k+1}$, est dite *dérivation indicée gauche* si les positions des symboles dérivés sont croissantes, c'est-à-dire si $i_p \leq i_q$ pour tout $1 \leq p < q \leq k$. \diamond

Définition 7.2.4 (Inversion dans une dérivation indicée) Soit $D = x_1, i_1, \dots, x_k, i_k, x_{k+1}$ une dérivation indicée dans une grammaire hors-contexte. On appelle *inversion dans D* toute paire de positions $(p, q) \in [1..k]^2$ telle que $p < q$ et $i_p > i_q$. On note $\text{Inv}(D)$ l'ensemble des inversions dans D . \diamond

Exemple 7.2.5 Considérons la dérivation $ABC \rightarrow ABc \rightarrow aBc \rightarrow abc$, et soit D la dérivation indicée correspondante. Alors $\text{Inv}(D) = \{(1, 2), (1, 3)\}$. \clubsuit

Notons qu'une dérivation indicée D est une dérivation indicée gauche si et seulement si $\text{Inv}(D) = \emptyset$. Le but de cette partie est de montrer qu'à toute dérivation indicée correspond une dérivation indicée gauche de même longueur, origine et extrémité. La preuve se fera par induction bien fondée (voir Annexe A). Pour cela, nous définissons un ordre sur les inversions qui sera étendu aux dérivations.

Définition 7.2.6 Soient (p_1, q_1) et (p_2, q_2) deux couples d'entiers. On note $(p_1, q_1) \prec (p_2, q_2)$ si et seulement si $(p_2 < p_1$ ou bien $p_2 = p_1$ et $q_2 < q_1)$.

Etant données deux dérivations D, D' , on note $D \ll D'$ si et seulement s'il existe des ensembles X, Y, Y' tels que

- $\text{Inv}(D) = X \cup Y$,
- $\text{Inv}(D') = X \cup Y'$,
- pour tout $(p, q) \in Y$, il existe $(p', q') \in Y'$ tel $(p, q) \prec (p', q')$. \diamond

Proposition 7.2.7 Soit $D = x_1, i_1, \dots, x_k, i_k, x_{k+1}$ une dérivation indicée dans une grammaire hors-contexte. On a pour tout $(p, q) \in \text{Inv}(D)$, $(k-1, k) \preceq (p, q) \preceq (1, 2)$.

Nous définissons également une opération de translation sur les dérivations indicées de la façon suivante.

Définition 7.2.8 (Translation d'une dérivation indicée) Soit $D = x_1, i_1, \dots, x_k, i_k, x_{k+1}$ une dérivation indicée dans une grammaire hors-contexte, où pour tout $i = 1, \dots, k+1$, x_i est de la forme uAv_i . Soient $j = |uA|$, $\alpha \in (V_T \cup V_N)^*$, et posons $l = |\alpha| - 1$. Pour $q = 1, \dots, k+1$, on définit $y_q = u\alpha v_q$ et si $q \neq k+1$, on pose $j_q = i_q + l$.

On note $\text{Tr}(D, j, \alpha)$ la séquence $y_1, j_1, \dots, y_k, j_k, y_{k+1}$. \diamond

Proposition 7.2.9 $\text{Tr}(D, j, \alpha)$ est une dérivation indicée et $\text{Inv}(\text{Tr}(D, j, \alpha)) = \text{Inv}(D)$.

Théorème 7.2.10 Pour toute dérivation indicée D , il existe une dérivation indicée gauche de même longueur, origine et extrémité que D .

PREUVE. Soit $D = x_1, i_1, \dots, i_{k-1}, x_k$ une dérivation indicée où $x_1, x_2, \dots, x_k \in V^*$ et $i_1, \dots, i_{k-1} \in \mathbb{N}$, et supposons que pour toute dérivation indicée D' de même longueur que D telle que $D' \ll D$, il existe une dérivation indicée gauche de même longueur, origine et extrémité que D' .

Si $\text{Inv}(D) = \emptyset$ alors D est déjà une dérivation indicée gauche. Supposons maintenant que $\text{Inv}(D) \neq \emptyset$. Nous allons construire une dérivation indicée D' de même longueur, origine et extrémité que D , telle que $D' \ll D$.

Soit $(p, q) = \max_{\prec} \text{Inv}(D)$, et posons $x_p = uAvBw$. Nécessairement, pour tout $1 \leq n \leq p$ on a $i_n \leq i_p$, et pour tout $p < m < q$ on a $i_m \geq i_p > i_q$. Ceci signifie qu'aucune des réécritures correspondantes ne peut avoir lieu au sein de u ou de v . On peut donc écrire :

$$\begin{aligned} x_p &= uAvBw, & i_p &= |uAvB|, & x_{p+1} &= uAv\beta w, \\ x_q &= uAv\beta'w', & i_q &= |uA|, & x_{q+1} &= u\alpha v\beta'w'. \end{aligned}$$

Définissons les dérivations indicées suivantes :

$$\begin{aligned} D_1 &= x_1, i_1, \dots, x_p, \\ D_2 &= x_{p+1}, i_{p+1}, \dots, x_q, \\ D_3 &= x_{q+1}, i_{q+1}, \dots, x_k. \end{aligned}$$

On a donc $D = D_1.(x_p, i_p, x_{p+1}).D_2.(x_q, i_q, x_{q+1}).D_3$.

Soient $j_p = |uA|$, $y_{p+1} = u\alpha vBw$, $j_{p+1} = |u\alpha vB|$ et $y_{p+2} = u\alpha v\beta w$. On définit la séquence suivante :

$$D' = D_1.(x_p, j_p, y_{p+1}).(y_{p+1}, j_{p+1}, y_{p+2}).\text{Tr}(D_2, j_p, \alpha).D_3.$$

Il est aisé de vérifier que D' est une dérivation indicée de même origine, extrémité et longueur que D . En posant $D' = y_1, j_1, \dots, j_{k-1}, y_k$, on a :

- pour tout $1 \leq m \leq p-1$, $j_m = i_m$,
- $j_p = i_q$,
- $j_{p+1} \geq j_p$,
- pour tout $p+2 \leq m \leq q-1$, $j_m = i_{m-1} + |\alpha| - 1$,
- pour tout $q+1 \leq m \leq k-1$, $j_m = i_m$.

Soit $n \in [1..p-1]$, et supposons qu'il existe $m \in \mathbb{N}$ tel que $(n, m) \in \text{Inv}(D')$.

1. Si $n < m \leq p - 2$ alors $i_n = j_n > j_m = i_m$ et on aurait donc $(n, m) \in \text{Inv}(D)$, contredisant l'hypothèse que $(p, q) = \max_{\prec} \text{Inv}(D)$.
2. Si $m = p$ alors $i_n = j_n > j_p = i_q$, et on aurait $(n, q) \in \text{Inv}(D)$, une contradiction.
3. Si $m = p + 1$ alors $i_n = j_n > j_{p+1} \geq j_p = i_q$, et on aurait à nouveau $(n, q) \in \text{Inv}(D)$, ce qui est impossible.
4. Si $p + 1 < m \leq q$, alors $i_n = j_n > j_m = i_{m-1} + |\alpha| - 1 \geq i_{m-1} - 1$. Si $i_n > i_{m-1}$, alors $(n, m - 1) \in \text{Inv}(D)$, ce qui est impossible. Sinon on a $i_n = i_{m-1}$, et comme $i_n \leq i_p \leq i_{m-1}$, on en déduit que $i_n = i_p$ et que $(n, q) \in \text{Inv}(D)$, ce qui est impossible.
5. Si $q < m \leq k - 1$, alors $i_n = j_n > j_m = i_m$, d'où $(n, m) \in \text{Inv}(D)$, ce qui est impossible.

Supposons maintenant qu'il existe $p < m \leq q$ tel que $(p, m) \in \text{Inv}(D')$. Comme $j_{p+1} \geq j_p$ par construction, nécessairement, $p + 1 < m \leq q$. Mais dans ce cas, on a $i_q = j_p > j_m = i_{m-1} + |\alpha| - 1 \geq i_{m-1} - 1$, d'où $i_p > i_q \geq i_{m-1}$, et on aurait alors $(p, m - 1) \in \text{Inv}(D)$, ce qui est impossible.

On en déduit que si $\text{Inv}(D') \neq \emptyset$, alors $\max_{\prec} \text{Inv}(D') \prec \max_{\prec} \text{Inv}(D)$, d'où $D' \ll D$. Par hypothèse d'induction, il existe une dérivation indiquée gauche de même longueur, origine et extrémité que D' , et donc que D . ■

Remarque En analysant la preuve ci-dessus, on constate que la nouvelle dérivation ne fait que permuter l'ordre d'application des règles.

Exemple 7.2.11 Soit $A \rightarrow AA|a$ une grammaire hors-contexte et la dérivation indiquée :

$AA \Rightarrow_2 AAA \Rightarrow_1 AAAA$.

En permutant l'ordre d'application de la règle $A \rightarrow AA$, on obtient la dérivation indiquée gauche :

$AA \Rightarrow_1 AAA \Rightarrow_3 AAAA$. ♣

2.2 Dérivations gauches

Définition 7.2.12 Soit $x_1 \dots x_k$ une dérivation (définition de Δ_3 - cf les relations de dérivation page 68, au chapitre 6) dans une grammaire hors-contexte. On dit que cette dérivation est une *dérivation gauche* si pour chaque élément x_i de la dérivation, le non-terminal réécrit est le non-terminal le plus à gauche de x_i .

Pour une grammaire hors-contexte donnée, on définit la relation \Rightarrow_g par :

$x \Rightarrow_g y$ si et seulement si il existe une chaîne terminale u , un non-terminal A , une chaîne v et une règle $A \rightarrow \alpha$ tels que $x = uAv$ et $y = u\alpha v$.

Une dérivation x_1, \dots, x_k est une dérivation gauche si et seulement si $x_i \Rightarrow_g x_{i+1}$ pour $i = 1 \dots k - 1$. ◇

Exemple 7.2.13 Soit la grammaire suivante avec $V_T = \{a, +, *, (,)\}$, $V_N = \{E, T, F\}$ et les règles :

$E \rightarrow E + T | T$

$T \rightarrow T * F | F$

$F \rightarrow (E) | a$

On a : $E \Rightarrow_g E + T \Rightarrow_g T + T \Rightarrow_g F + T \Rightarrow_g a + T \Rightarrow_g a + F \Rightarrow_g a + a$. ♣

En fait une dérivation gauche est un cas particulier d'une dérivation indicée gauche, comme le montre la preuve du théorème suivant.

Théorème 7.2.14 *Soit A un non-terminal et x une chaîne terminale. Si $A \Rightarrow^p x$, alors $A \Rightarrow_g^p x$.*

PREUVE. Si $A \Rightarrow^p x$, alors le Théorème 7.2.10 prouve qu'il existe une dérivation indicée gauche $x_1, i_1, \dots, x_{k-1}, i_{k-1}, x_k$ avec $x_1 = A$ et $x_k = x$.

La dérivation $D = x_1, x_2, \dots, x_k$ est alors une dérivation gauche d'origine A et d'extrémité x . En effet, soit $x_j \neq x$ un élément de cette dérivation. Aucune lettre de x_j d'indice strictement inférieur à i_j (c'est-à-dire à gauche du symbole réécrit de x_j) n'est réécrite dans la dérivation D . Toutes ces lettres sont donc des lettres de x , et la chaîne x étant terminale, sont donc des lettres terminales. Il s'ensuit que D est une dérivation gauche. ■

Remarque On peut définir de façon analogue des *dérivations indicées droites* et des *dérivations droites*, et démontrer les équivalents pour ces dérivations des théorèmes 7.2.10 et 7.2.14

3 Décomposition des dérivations indicées gauches

Définition 7.3.1 (Somme de deux dérivations indicées gauches) Soit D_1 la dérivation indicée gauche $x_1, i_1, \dots, x_{k-1}, i_{k-1}, x_k$ où $x_1, \dots, x_k \in V^*$ et $i_1, \dots, i_{k-1} \in \mathbb{N}$. Soit D_2 la dérivation indicée gauche $y_1, j_1, \dots, y_{l-1}, j_{l-1}, y_l$ où $y_1, \dots, y_l \in V^*$ et $j_1, \dots, j_{l-1} \in \mathbb{N}$.

$D_1 + D_2$ est la dérivation indicée d'origine $x_1 y_1$ obtenue en effectuant D_1 puis D_2 , c'est-à-dire la dérivation :

$$x_1 y_1, i_1, \dots, x_{k-1} y_1, i_{k-1}, x_k y_1, |x_k| + j_1, \dots, x_k y_{l-1}, |x_k| + j_{l-1}, x_k y_l.$$

Il est clair que, puisque D_1 et D_2 sont des dérivations indicées gauches, il en est de même de $D_1 + D_2$ ◇

Remarque Il est facile de voir que cette somme est associative, ce qui rend sans ambiguïté la somme de n dérivations $D_1 + D_2 + \dots + D_n$.

Exemple 7.3.2 Soit la grammaire $S \rightarrow \varepsilon | aSbS | bSaS$.

Soit D la dérivation indicée $S, 1, \varepsilon$.

Notons que a (respectivement b) sont des dérivations indicées gauches de longueur 0.

$$b + D = bS, 2, b$$

$$a + D + b = bSa, 2, ba$$

$$a + D + b + D = bSaS, 2, baS, 3, ba$$

♣

Propriété 7.3.3 (Décomposition des dérivations indicées gauches) Soient $x = u_1 u_2$, où $u_1, u_2 \in V^*$, et D une dérivation indicée gauche d'origine x . Il existe une et une seule décomposition de D en deux dérivations gauches D_1 d'origine u_1 et D_2 d'origine u_2 telles que $D = D_1 + D_2$.

PREUVE. Soit $D = w_1, i_1, \dots, w_{k-1}, i_{k-1}, w_k$ une dérivation indicée gauche où $x = w_1$, pour i de 1 à k le mot w_i est élément de V^* , et $i_1, \dots, i_{k-1} \in \mathbb{N}$. Si $k = 1$ alors on a le résultat en posant $D_1 = u_1$ et $D_2 = u_2$.

Supposons que $k > 1$ et que la propriété est vraie pour toute dérivation indicée gauche de longueur inférieure à k ; montrons que la propriété est aussi vérifiée par D . Pour cela, nous considérons les deux cas suivants :

$i_1 > |u_1|$. Dans ce cas on pose $D_1 = u_1$. Pour n de 1 à k , soit v_n tel que $w_n = u_1 v_n$ (v_n est le suffixe de u_1 dans w_n), et soit $j_n = i_n - |u_1|$. On pose $D_2 = v_1, j_1, \dots, j_{k-1}, v_k$

$i_1 \leq |u_1|$. Dans ce cas, il existe des chaînes s, t, α et un non terminal A tels que $w_1 = sAtv_1$, $A \rightarrow \alpha$, $w_2 = s\alpha tv_1$. Soit D' la sous-dérivation D d'origine w_2 . Par hypothèse d'induction, il existe une décomposition *unique* de D' en D'_1 d'origine $s\alpha t$, et D_2 d'origine v_1 . Soit $D_1 = s\alpha t, i_1, D'_1$ (ce qui assure l'unicité de D_1), on a alors $D = D_1 + D_2$. ■

Propriété 7.3.4 (Décomposition des dérivations indicées gauches) Soit $x = u_1 u_2 \dots u_n$, où pour tout i , $u_i \in V^*$, et soit D une dérivation indicée gauche d'origine x . Il existe une et une seule décomposition de D en n dérivations indicées gauches D_1 d'origine u_1 et D_2 d'origine u_2 , ... D_n d'origine u_n telles que $D = D_1 + D_2 + \dots + D_n$.

PREUVE. La preuve est une généralisation par induction sur n de la propriété 7.3.3 ci-dessus. ■

4 Arbre de dérivation

Les arbres de dérivation sont une autre représentation plus visuelle du processus de dérivation. Nous verrons qu'il y a une bijection entre les arbres de dérivations et les dérivations indicées gauche ayant comme origine un symbole. Nous donnerons deux définitions des arbres de dérivation, la première définit un arbre comme une expression et la deuxième est plus proche des arbres utilisés en informatique. Pour démontrer l'existence de la bijection, nous utiliserons la représentation des arbres de dérivation par des expressions, mais en pratique, on préfère la deuxième représentation.

Dans ce paragraphe, on note les grammaires comme précédemment par $G = \langle V_T, V_N, S, R \rangle$ et on pose $V = V_T \cup V_N$.

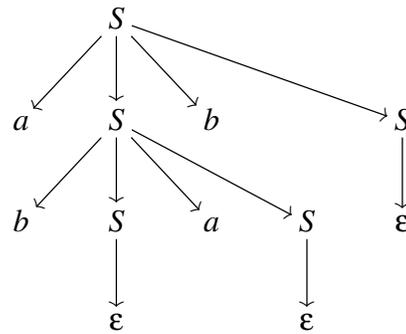
Exemple 7.4.1 Soit la grammaire $S \rightarrow aSbS | bSaS | \epsilon$ et la dérivation indicée gauche :

$$S \Rightarrow_1 aSbS \Rightarrow_2 abS \Rightarrow_3 abaSbS \Rightarrow_4 ababS \Rightarrow_5 abab. \quad \clubsuit$$

Les indices sont inutiles car cette dérivation est une dérivation gauche où le non-terminal le plus à gauche est dérivé à chaque pas de la dérivation. L'arbre de dérivation est représenté

— soit par l'expression $S(a, S(b, S(), a, S()), b, S())$,

— soit par l'arbre



Définition 7.4.2 (Arbre = Expression) Un arbre de dérivation d'origine a , où $a \in V$, pour la grammaire G est :

- soit a ,
- soit de la forme $a(e_1, \dots, e_n)$ où :
 1. $n \geq 0$,
 2. Pour i de 1 à n , e_i est un arbre de dérivation d'origine b_i ,
 3. $a \rightarrow b_1 \cdots b_n$ est une règle de la grammaire G .

◇

Exemple 7.4.3 Pour la grammaire $S \rightarrow aSbS|bSaS|\epsilon$ de l'exemple précédent, les expressions $a, b, S(), S(b, S()), a, S(), S(a, S(b, S()), a, S()), b, S()$ sont des arbres de dérivation. ♣

Définition 7.4.4 (Arbre = Domaine d'arbre et étiquettes)

1. Domaine d'arbre : un *domaine d'arbre* D est un ensemble de suites finies d'entiers naturels
 - qui est fermé par préfixe : si $u.v \in D$ alors $u \in D$
 - qui vérifie : si $i \in \mathbb{N}$ et $u.(i+1) \in D$ alors $u.i \in D$
 On reconnaîtra dans un domaine d'arbre la numérotation usuelle des livres (dont cet ouvrage) : si le paragraphe 2.3.2 existe, alors existent notamment les paragraphes 2 et 2.3.1.
2. Arbre : soit D un domaine d'arbre fini, un *arbre de dérivation* pour la grammaire G est une application ar de D dans $V \cup \{\epsilon\}$ telle que si $n \in \mathbb{N}$, $u \in D$, $u.n \in D$ et $u.(n+1) \notin D$ alors $ar(u) \rightarrow ar(u.1) \cdots ar(u.n)$ est une règle de G .

Soit $ar : D \mapsto V$ un arbre de dérivation :

- $u \in D$ est un *sommet* de l'arbre.
- ϵ (la suite vide d'entiers) est la *racine* de l'arbre.
- $ar(u)$ est l'*étiquette* du sommet u .
- Si $u \in D$, $i \in \mathbb{N}$ et $u.i \in D$ alors le sommet $u.i$ est un *successeur* du sommet u .
- Si $u \in D$ et u n'a pas de successeur, alors u est une *feuille* de l'arbre.
- Si $u \in D$ et $uv \in D$, alors uv est un *descendant* du sommet u .

◇

Les arbres de dérivation ainsi définis, sont dessinés comme dans l'exemple 7.4.1. Les suites d'entiers sont omises et remplacées par des flèches vers le bas allant d'un sommet vers chacun de ses successeurs. Le successeur $u.(i+1)$ est placé à droite du sommet $u.i$.

Théorème 7.4.5 *Soit G une grammaire. Il y a une bijection entre les arbres de dérivation de G et les dérivations indicées gauches ayant pour origine un symbole de G .*

PREUVE. Soit V le vocabulaire de G . On définit une application $a2d$ entre les arbres de dérivation de G (suivant la définition 7.4.2) et les dérivations indicées gauches ayant pour origine un symbole de G par :

- $a2d(a) = a$ où $a \in V$
- $a2d(a(e_1, \dots, e_n)) = a, 1, a2d(e_1) + \dots + a2d(e_n)$

Nous prouvons que $a2d$ est une bijection.

1. **$a2d$ est injective.** Soient deux arbres T et U tels que $a2d(T) = a2d(U)$. Par induction sur les sous-arbres de T , on montre que $T = U$. Admettons que la propriété ci-dessus soit vérifiée pour tout sous-arbre strict de T .

$T \in V$. Par définition de $a2d$, $a2d(T) = T$. Puisque $a2d(U) \in V$, seul le premier cas de la définition de $a2d$ s'applique, donc $a2d(U) = U = T$.

$T = a(e_1, \dots, e_n)$. Par définition, $a2d(T) = a, 1, a2d(e_1) + \dots + a2d(e_n)$. Soit D la dérivation indicée gauche $a2d(e_1) + \dots + a2d(e_n)$, et pour i de 1 à n , soit b_i l'origine de l'arbre e_i .

- (a) Puisque $a2d(T) = a2d(U)$ alors U et T ont même origine, donc $U = a(f_1, \dots, f_p)$. Pour i de 1 à p , on note c_i la racine de f_i .
- (b) Puisque $a2d(T) = a2d(U)$ et que donc $b_1 \dots b_n$ et $c_1 \dots c_p$ sont l'origine de la dérivation D , il en résulte que $n = p$ et pour i de 1 à n , $b_i = c_i$ et $U = a(f_1, \dots, f_n)$.
- (c) Par définition de $a2d$, $a2d(U) = a, 1, a2d(f_1) + \dots + a2d(f_n)$, on en déduit que $D = a2d(f_1) + \dots + a2d(f_n)$. En résumé, on a

$$D = a2d(e_1) + \dots + a2d(e_n) = a2d(f_1) + \dots + a2d(f_n),$$

D a comme origine $b_1 \dots b_n$ et pour i de 1 à n , b_i est la racine commune à $a2d(e_i)$ et $a2d(f_i)$. D'après la propriété 7.3.4 il en résulte que pour i de 1 à n , $a2d(e_i) = a2d(f_i)$. Par hypothèse d'induction pour i de 1 à n , $e_i = f_i$, donc $T = U$.

2. **$a2d$ est surjective.** On prouve ce résultat par induction sur la longueur des dérivations. Soit D une dérivation indicée gauche ayant comme origine le symbole $a \in V$. Si D est de longueur 0, alors $D = a$ et $a2d(D) = D$.

Supposons maintenant que D soit de longueur non nulle, et donc de la forme $D = a, 1, D'$. Soit $b_1 \dots b_n$ l'origine de D' où pour i de 1 à n , b_i est un symbole de V . Puisque D est une dérivation, par définition, $a \rightarrow b_1 \dots b_n$ est une règle de la grammaire. D'après la propriété 7.3.4, $D' = D_1 + \dots + D_n$ où pour i de 1 à n , D_i est une dérivation indicée gauche d'origine b_i .

Par hypothèse d'induction, pour i de 1 à n , il existe un arbre de dérivation e_i d'origine b_i tel que $a2d(e_i) = D_i$. D'après la définition 7.4.2 des arbres, $a(e_1, \dots, e_n)$ est un arbre de dérivation de la grammaire G . Par définition, $a2d(a(e_1, \dots, e_n)) = D$, donc $a2d$ est surjective. ■

5 Ambiguïté

Définition 7.5.1 Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors contexte. La grammaire G est *ambigüe* s'il existe une chaîne terminale admettant au moins deux arbres de dérivation de racine S , soit encore de façon équivalente, d'après le théorème 7.4.5, deux dérivations gauches distinctes d'origine S et d'extrémités identiques. Une chaîne terminale qui est l'extrémité d'au moins deux dérivations gauches d'origine S est une *chaîne ambigüe* de la grammaire G . Un langage hors-contexte est ambigu si *toutes* les grammaires hors-contextes qui l'engendrent sont ambigües. ◇

Exercice 7.5.2 Soit la grammaire suivante, ayant pour vocabulaire terminal l'unique symbole a , et comme règles :

$$S \rightarrow A|B, A \rightarrow a, B \rightarrow a$$

1. Montrez qu'elle est ambigüe.
2. Le langage engendré par cette grammaire n'est pas ambigu. Donnez une grammaire non ambigüe équivalente.

Exercice 7.5.3 Soit la grammaire G suivante, de vocabulaire terminal $0, 1$, et de règles $S \rightarrow 0S1|0S11|\epsilon$

1. Montrez qu'elle est ambigüe.
2. Prouvez que $L(G) = \{0^n 1^m \mid n \leq m \leq 2n\}$.
3. Montrez que le nombre de dérivations gauches dans la grammaire G de la chaîne $0^n 1^m$ est $\binom{n}{m-n}$.
Indication : la longueur de la dérivation de cette chaîne est $n + 1$. Une dérivation particulière est caractérisée par le choix des $m - n$ emplacements de l'usage de la règle $S \rightarrow 0S11$.
4. Donnez une grammaire non ambigüe équivalente.

Exercice 7.5.4 La grammaire suivante, de vocabulaire terminal a, b , et de règles $S \rightarrow aSb|SS|\epsilon$ peut être vue comme décrivant la structure des parenthèses d'une expression bien parenthésée, a jouant le rôle de la parenthèse ouvrante et b celui de la parenthèse fermante.

1. Montrez que cette grammaire est ambigüe.
2. Montrez qu'elle est équivalente à la grammaire $S \rightarrow aSbS|\epsilon$
3. Montrez que la grammaire ci-dessus n'est pas ambigüe

Exemple 7.5.5 Le langage $\{a^i b^j c^k \mid i = j \text{ ou } j = k\}$ est un langage hors-contexte ambigu. Il est facile de donner une grammaire hors-contexte qui l'engendre. Il est intuitif mais difficile de prouver que ce langage est ambigu, c'est pourquoi nous ne le proposons pas en exercice. ♣

Remarque Il n'y a pas d'algorithme permettant de répondre à la question "la grammaire G est-elle ambiguë ?" : le problème de l'ambiguïté d'une grammaire est indécidable.

Exercice 7.5.6 Soit la grammaire suivante, de vocabulaire terminal $si, alors, sinon, e, a$, et de règles

$I \rightarrow si e alors I \mid si e alors I sinon I \mid a$.

1. Montrez que cette grammaire est ambiguë.
2. Trouvez une grammaire équivalente non ambiguë.

Exercice 7.5.7 Soit la grammaire G suivante, de vocabulaire terminal a, b, c et de règles

$S \rightarrow aSc \mid aS \mid T$

$T \rightarrow bTc \mid bT \mid \varepsilon$

1. Montrez que cette grammaire est ambiguë.
2. Montrez que $L(G) = \{a^p b^q c^r \mid p + q \geq r \geq 0\}$
3. Trouvez une grammaire équivalente non ambiguë.

Exercice 7.5.8 Soit $L = \{w \in \{a, b\}^* \mid w \text{ contient autant de } a \text{ que de } b\}$. Soient les trois grammaires G_1, G_2, G_3 ayant pour règles

— $G_1 : S \rightarrow \varepsilon \mid aSb \mid bSa \mid SS$

— $G_2 : S \rightarrow aSbS \mid bSaS \mid \varepsilon$

— $G_3 : S \rightarrow aB \mid bA \mid \varepsilon$

$A \rightarrow aS \mid bAA$

$B \rightarrow bS \mid aBB$

1. Montrez que ces trois grammaires engendrent le langage L .
2. Montrez qu'elles sont toutes ambiguës
3. Donnez une grammaire non ambiguë engendrant L

Exercice 7.5.9 Soit $V_T = \{0, 1\}$ un vocabulaire terminal et $L = V_T^* - \{ww \mid w \in V_T^*\}$. Montrez que L est un langage hors-contexte. Le problème n'est pas facile et nous le décomposons en questions intermédiaires. Il est clair que les chaînes de longueur impaire (sur le vocabulaire terminal $\{0, 1\}$) sont éléments de L .

1. Soit Y l'ensemble des chaînes de longueur impaire dont le milieu est 1. Trouvez une grammaire engendrant Y .
Soit Z l'ensemble des chaînes de longueur impaire dont le milieu est 0. Trouvez une grammaire engendrant Z .
2. Montrez que toute chaîne de $YZ \cup ZY$ n'est pas de la forme ww

3. Montrez que toute chaîne de longueur paire, qui n'est pas de la forme ww , est élément de $YZ \cup ZY$.

4. En déduire une grammaire engendrant L

On trouvera un excellent corrigé de cet exercice dans le cours du Professeur Nievergelt :
<http://www.jn.inf.ethz.ch/education/script/chapter5.pdf>.

Exercice 7.5.10 Soit la grammaire $S \rightarrow SS|a$.

1. Montrez qu'elle engendre a^+ l'ensemble des suites non vides de a .

2. Montrez que cette grammaire est ambiguë

3. Soit X_n le nombre de dérivations gauches de a^n . Prouvez que $X_{n+1} = (1/(n+1))\binom{2n}{n}$

Indication : X_n est un nombre de Catalan. On admettra que les nombres de Catalan satisfont la relation de récurrence $C_0 = 1$ et $C_{n+1} = \sum_{i=0}^n C_i C_{n-i}$.

Deuxième partie
Notions avancées

Chapitre 8

Définition de langages par induction

Il est naturel de définir des langages par des définitions inductives, prenant la forme de systèmes d'équations. Les langages définis par des systèmes d'équations sont appelés des langages algébriques et dans ce chapitre

- on montre qu'un langage est algébrique si et seulement si il est hors-contexte
- on étudie les transformations des systèmes d'équations qui en préservent l'ensemble des solutions
- on donne une condition nécessaire et suffisante, *calculable* pour qu'un système d'équation ait une et une seule solution.

1 Exemples de définitions inductives de langages

Nous utilisons ci-dessous, sauf exception, les notations des expressions régulières.

1. $X = \{a^n b^n \mid n \geq 0\}$ est solution de l'équation $X = \varepsilon + aXb$. C'est aussi la plus petite solution de cette équation.

En effet, puisque $\{\varepsilon\} \subset X$, ε est dans toute solution, et si $a^n b^n$ est dans une solution, puisque $aXb \subset X$, $a^{n+1} b^{n+1}$ est aussi dans cette solution.

Donc, par induction sur n , $\text{iv}\{a^n b^n \mid n \geq 0\}$ est un sous-ensemble de toute solution.

Cette équation n'a d'ailleurs qu'une solution. Supposons au contraire qu'il y a un plus petit mot x dans une solution L , avec $x \notin \{a^n b^n \mid n \geq 0\}$.

Puisque x est un mot de la solution L , il s'écrit ayb avec $y \in L$ et $y \notin \{a^n b^n \mid n \geq 0\}$, ce qui contredit le fait que x est le plus petit mot avec cette propriété.

Notons que le langage $\{a^n b^n \mid n \geq 0\}$ est engendré par la grammaire $X \rightarrow \varepsilon \mid aXb$: le signe égal de l'équation devient une flèche et le plus devient une barre.

2. $X = \{a^p b^q c^{p+q} \mid p \geq 0, q \geq 0\}$ est solution de l'équation $X = \{b^q c^q \mid q \geq 0\} + aXc$. C'est aussi la plus petite.

En effet, puisque $\{b^q c^q \mid q \geq 0\} \subset X$, $\{b^q c^q \mid q \geq 0\}$ est dans toute solution, et si $a^p b^q c^{p+q}$ est dans une solution, puisque $aXc \subset X$, $a^{p+1} b^q c^{p+q+1}$ est aussi dans cette solution.

Donc, par induction sur p , $\{a^p b^q c^{p+q} \mid p \geq 0, q \geq 0\}$ est un sous-ensemble de toute

solution.

Puisque ci-dessus, on a vu que $\{b^n c^n \mid n \geq 0\}$ est la plus petite solution de $Y = \varepsilon + bYc$, il en résulte que $\{a^p b^q c^r \mid p + q = r\}$ est la valeur de X de la plus petite solution du système :

$$X = Y + aXc$$

$$Y = \varepsilon + bYc$$

Comme ci-dessus, on peut vérifier que cette solution est unique.

Notons que le langage $\{a^p b^q c^{p+q} \mid p \geq 0, q \geq 0\}$ est engendré par le non-terminal X de la grammaire

$$X \rightarrow Y | aXc$$

$$Y \rightarrow \varepsilon | bYc$$

Comme ci-dessus, le signe égal des équations devient une flèche et le plus devient une barre.

3. On définit les expressions dont les opérandes sont a et b et les opérateurs $+$ et $*$.
Toute expression est un terme, ou bien s'écrit $e + t$ où e est une expression et t est un terme.

Tout terme est un facteur, ou bien s'écrit $t * f$ où t est un terme et f est un facteur.

Un facteur est de la forme a, b ou (e) où e est une expression.

D'après la définition ci-dessus, l'ensemble des expressions, termes et facteurs est la plus petite solution respectivement en E, T et F du système d'équations :

$$E = T + \underline{E} + T$$

$$T = F + T * F$$

$$F = a + b + (E)$$

Notons que le signe $\underline{+}$ des expressions est souligné pour le distinguer du plus des expressions régulières.

Comme dans les cas ci-dessus, la solution est unique.

2 Théorèmes du point fixe

Toute définition inductive peut être vue comme la solution d'une équation $x = f(x)$ où $f : E \mapsto E$, autrement dit comme un point fixe de la fonction f . Aussi on étudie les conditions d'existence et de construction de points fixes.

2.1 Ordre faiblement complet

Définition 8.2.1 (ordre faiblement complet) Soit E un ensemble ordonné.

Une suite dénombrable sur cet ensemble, est une suite u_i d'éléments de E où $i \in \mathbb{N}$.

Une relation d'ordre sur un ensemble E est dite *faiblement complète* si toute suite croissante dénombrable a une borne supérieure. Un ensemble ordonné est faiblement complet, si sa relation d'ordre est faiblement complète. \diamond

Lorsqu'elle existe, la borne supérieure de l'ensemble X est notée $\sup(X)$.

Lorsqu'elle existe, la borne inférieure de l'ensemble X est notée $\inf(X)$.

Définition 8.2.2 (fonction continue) Soient E et F deux ensembles faiblement complets. La fonction $f : E \mapsto F$ est continue, si pour toute suite u_i croissante dénombrable d'éléments de E , $f(\sup\{u_i \mid i \in \mathbb{N}\}) = \sup\{f(u_i) \mid i \in \mathbb{N}\}$ \diamond

Propriété 8.2.3 *Toute fonction continue est croissante*

PREUVE. Soient E et F deux ensembles faiblement complets et $f : E \mapsto F$ une fonction continue. Soient $a, b \in E$ avec $a \leq b$. Soit u_i la suite dénombrable où $u_0 = a$ et pour $i \in \mathbb{N}$, $u_{i+1} = b$. Puisque $a \leq b$, $\sup\{u_i \mid i \in \mathbb{N}\} = b$, donc par continuité $f(b) = \sup\{f(a), f(b)\}$, autrement dit $f(a) \leq f(b)$. \blacksquare

Théorème 8.2.4 (Théorème de Kleene) *Soit E un ensemble faiblement complet ayant un plus petit élément noté \perp . Soit f une fonction continue de E dans lui-même. Le plus petit point fixe de f est $\sup\{f^i(\perp) \mid i \in \mathbb{N}\}$.*

PREUVE. 1. La suite $f^i(\perp)$ où $i \in \mathbb{N}$ est croissante et donc, $\sup\{f^i(\perp) \mid i \in \mathbb{N}\}$ est bien définie.

Montrons, par induction sur i , que pour tout $i \in \mathbb{N}$, $f^i(\perp) \leq f^{i+1}(\perp)$.

Puisque \perp est le plus petit élément de E , $\perp \leq f(\perp)$, ce qui rend vraie la propriété pour $i = 0$.

Supposons $f^i(\perp) \leq f^{i+1}(\perp)$. Puisque f est continue, elle est croissante donc, en appliquant f aux deux membres de l'inégalité ci-dessus, $f^{i+1}(\perp) \leq f^{i+2}(\perp)$.

Puisque E est faiblement complet et que la suite $f^i(\perp)$ est croissante, $\sup\{f^i(\perp) \mid i \in \mathbb{N}\}$ existe.

2. $\sup\{f^i(\perp) \mid i \in \mathbb{N}\}$ est un point fixe de f .

Puisque \perp est le plus petit élément de E , on a :

(a) $\sup\{f^i(\perp) \mid i \in \mathbb{N}\} = \sup\{f^{i+1}(\perp) \mid i \in \mathbb{N}\}$.

Par continuité, $f(\sup\{f^i(\perp) \mid i \in \mathbb{N}\}) = \sup\{f^{i+1}(\perp) \mid i \in \mathbb{N}\}$.

D'après l'égalité (a), $f(\sup\{f^i(\perp) \mid i \in \mathbb{N}\}) = \sup\{f^i(\perp) \mid i \in \mathbb{N}\}$.

3. $\sup\{f^i(\perp) \mid i \in \mathbb{N}\}$ est le plus petit des points fixes de f .

Soit a un point fixe de f . Il suffit de montrer que pour tout $i \in \mathbb{N}$, $f^i(\perp) \leq a$.

Pour $i = 0$, cette inégalité est vraie, car \perp est le plus petit élément de E .

Supposons $f^i(\perp) \leq a$. Puisque f est croissante, $f^{i+1}(\perp) \leq f(a)$. Puisque a est un point fixe, $f^{i+1}(\perp) \leq a$.

Donc par induction, pour tout $i \in \mathbb{N}$, $f^i(\perp) \leq a$, et par suite $\sup\{f^i(\perp) \mid i \in \mathbb{N}\} \leq a$ \blacksquare

2.2 Ordre complet

Définition 8.2.5 (ordre complet, treillis complet) Soit E un ensemble ordonné.

Une relation d'ordre sur un ensemble E est dite *complète* si tout sous-ensemble de E a une borne supérieure dans E .

Un ensemble ordonné est complet, si sa relation d'ordre est complète.

Un ensemble ordonné complet est aussi appelé un *treillis* complet. \diamond

De la définition d'un treillis complet E , il résulte que E a un plus grand élément $\sup(E)$ et un plus petit élément $\sup(\emptyset)$.

Exemple 8.2.6

1. Soit E un ensemble. L'ensemble $\mathcal{P}(E)$, aussi noté 2^E , ordonné par \subset est un treillis complet. En effet soit X un sous-ensemble de $\mathcal{P}(E)$, la borne supérieure de X est l'ensemble $\bigcup X$.
2. Soient X et E deux ensembles. Soit $\mathcal{P}(E)^X$ l'ensemble des applications X dans $\mathcal{P}(E)$. On ordonne cet ensemble par l'extension de l'ordre des sous-ensembles de E aux fonctions : Soient $f, g \in \mathcal{P}(E)^X$, $f \subset g$ si et seulement pour tout $x \in X$, $f(x) \subset g(x)$. Soit Y un sous-ensemble de $\mathcal{P}(E)^X$, la borne supérieure de Y est aussi notée $\bigcup Y$ et elle est définie par :
pour tout $x \in X$, $(\bigcup Y)(x) = \bigcup \{f(x) \mid f \in Y\}$. Par suite $\mathcal{P}(E)^X$, muni de l'ordre défini ci-dessus, est un treillis complet. \clubsuit

Propriété 8.2.7 *Tout sous-ensemble d'un treillis complet a une borne inférieure.*

PREUVE. Soit E un treillis complet et $X \subset E$. Soit M l'ensemble des minorants de X .

Puisque E est complet, la borne supérieure de M existe. Soit m cette borne.

Tout élément de X est un majorant de M , donc m , étant le plus petit des majorants de M , est un minorant de X . Donc $m \in M$. Puisque m est la borne supérieure de M , m est aussi le plus grand élément de M , donc la borne inférieure de X . \blacksquare

Définition 8.2.8 (treillis) Soit E un ensemble ordonné.

E est un treillis si et seulement si tout sous-ensemble avec deux éléments a une borne supérieure et une borne inférieure. \diamond

Propriété 8.2.9 *Soit E un ensemble ordonné.*

E est un treillis si et seulement si tout sous-ensemble fini non vide a une borne supérieure et une borne inférieure.

La preuve est évidente par induction sur le nombre d'éléments des sous-ensembles finis non vides.

Remarque Puisque tout sous-ensemble d'un treillis complet possède une borne supérieure et une borne inférieure, un treillis complet est bien un treillis.

Théorème 8.2.10 (Théorème de Tarski) Soit E un treillis complet ordonné par \leq . Soit $f : E \mapsto E$ une fonction croissante. Alors $\inf\{c \in E \mid f(c) \leq c\}$ est le plus petit point fixe de f .

PREUVE. Soit $C = \{c \in E \mid f(c) \leq c\}$ et $p = \inf(C)$.

1. L'ensemble C n'est pas vide. En effet, s'il l'était, on aurait $f(c) > c$ pour tout $c \in E$. Donc en particulier, comme $\sup(E) \in E$, on devrait avoir $f(\sup(E)) > \sup(E)$, ce qui est impossible.
2. Montrons que $f(p) \leq p$.
Soit $c \in C$. Puisque $p = \inf(C)$, $p \leq c$. Puisque f est croissante $f(p) \leq f(c)$. Puisque $c \in C$, $f(c) \leq c$. Par suite $f(p) \leq c$.
L'élément $f(p)$ est donc un minorant de C , puisque p est le plus grand de ces minorants, $f(p) \leq p$.
3. Montrons que p est le plus petit point fixe de f .
Puisque $f(p) \leq p$ et que f est croissante, $f(f(p)) \leq f(p)$. Donc $f(p) \in C$ et puisque p est la borne inférieure de C , $p \leq f(p)$.
Donc $p = f(p)$. Puisque tous les points fixes de f sont éléments de C , et que p est la borne inférieure de C , il en résulte que p est le plus petit point fixe de f . ■

2.3 Treillis finis

1. Un treillis fini, ayant un plus petit élément, est complet.
En effet soit E un treillis fini, autrement dit E est fini, ayant un plus petit élément \perp . Tout sous-ensemble non vide de E étant fini, il a une borne supérieure. De même $\sup(\emptyset) = \perp$.
Donc tout sous-ensemble de E a une borne supérieure.
2. Dans un treillis fini, une fonction croissante est continue.
En effet soit E un treillis fini et $f : E \mapsto E$ une fonction croissante. Soit u_i une suite croissante dénombrable d'éléments de E . Puisque E est fini, il existe un plus petit entier j qui est le plus grand élément de la suite, la suite devenant stationnaire à partir de cet indice. Donc $\sup\{u_i \mid i \in \mathbb{N}\} = u_j$.
Puisque f est croissante, la suite $f(u_i)$ est aussi croissante et stationnaire à partir de l'indice j .
Donc $\sup\{f(u_i) \mid i \in \mathbb{N}\} = f(u_j)$.
Par suite $f(\sup\{u_i \mid i \in \mathbb{N}\}) = \sup\{f(u_i) \mid i \in \mathbb{N}\}$, autrement dit f est continue.
3. Calcul du plus petit point fixe d'une fonction croissante dans un treillis fini, ayant un plus petit élément.
Soit E un treillis fini, ayant un plus petit élément \perp . Soit $f : E \mapsto E$ une fonction croissante.
Puisque f est continue, d'après le théorème de Kleene 8.2.4, le plus petit point fixe de f est $\sup\{f^i(\perp) \mid i \in \mathbb{N}\}$. Dans la preuve de ce théorème, il est montré que la suite $f^i(\perp)$ est croissante.

Soit n le nombre d'éléments de E . Puisque la suite $f^i(\perp)$ est croissante, il existe un plus petit indice j où $j < n$ tel que $f^j(\perp) = f^{j+1}(\perp)$.

Une induction triviale montre que pour tout k tel que $k \geq j$, on a $f^j(\perp) = f^k(\perp)$.

Donc $f^j(\perp)$ est le plus petit point fixe de f . Aussi, pour calculer ce plus petit point fixe, il suffit de calculer la suite $f^i(\perp)$ jusqu'au premier indice $j < n$ tel que $f^j(\perp) = f^{j+1}(\perp)$.

3 Langages algébriques

Définition 8.3.1 (Système d'équations) Soient V_T et V_N deux vocabulaires disjoints et V l'union de ces deux vocabulaires. V_T est l'ensemble des terminaux et V_N est l'ensemble des non terminaux. Un non terminal est aussi appelé une variable.

Un système d'équations sur V_T et V_N est un triplet $\langle V_T, V_N, d \rangle$ où d est une application de V_N dans l'ensemble des parties finies de $\mathcal{P}(V^*)$.

Soit S un tel système d'équations, il est présenté sous la forme : pour toute variable A , $A = d(A)$. L'ensemble fini de mots $d(A)$ est la *définition* de A . Il est écrit comme la *somme* de ses mots. \diamond

Voici comment se présente un système d'équations avec deux terminaux a, b et deux variables A_0, A_1 :

$$\begin{aligned} A_0 &= aA_0 + bA_1 \\ A_1 &= aA_1b + \varepsilon \end{aligned}$$

Définition 8.3.2 (Interprétation) Soient V_T et V_N deux vocabulaires disjoints et V l'union de ces deux vocabulaires. Une *interprétation* de V_N sur V_T est une application I de V_N dans l'ensemble des parties de $\mathcal{P}(V_T^*)$; on note $I(A)$ l'image de la variable $A \in V_N$.

Soit I une telle interprétation. Elle est étendue aux mots et langages sur V de la façon suivante :

- $I(\varepsilon) = \{\varepsilon\}$
- Si $a \in V_T$, alors $I(a) = \{a\}$
- Si $x, y \in V^*$, alors $I(xy) = I(x)I(y)$
- Si L est un langage sur V , alors $I(L) = \bigcup \{I(x) \mid x \in L\}$ \diamond

Exemple 8.3.3 Soient a, b deux terminaux et A_0, A_1 deux variables.

Soit I l'interprétation suivante : $I(A_0) = \{a^n b^n \mid n \geq 0\}$ et $I(A_1) = \{b^n \mid n \geq 0\}$.

Par définition des extensions de I , on a : $I(bA_0 + aA_1) = \{ba^n b^n \mid n \geq 0\} \cup \{ab^n \mid n \geq 0\}$. \clubsuit

Remarque L'interprétation I de l'ensemble V_N de variables est l'ensemble de couples $\{(A, I(A)) \mid A \in V_N\}$. Ces couples sont souvent notés $A = I(A)$.

C'est ainsi que l'interprétation I ci-dessus est aussi notée : $A_0 = \{a^n b^n \mid n \geq 0\}, A_1 = \{b^n \mid n \geq 0\}$

Définition 8.3.4 (Solution d'un système d'équations) Soit $S = \langle V_T, V_N, d \rangle$ un système d'équations.

L'interprétation $I : V_N \mapsto \mathcal{P}(V_T^*)$ est une *solution de S* si et seulement si pour tout $A \in V_N$, $I(A) = I(d(A))$. \diamond

Exemple 8.3.5 Soit le système, avec deux terminaux a, b et deux variables A_0, A_1 , déjà présenté ci-dessus :

$$A_0 = aA_0 + bA_1$$

$$A_1 = aA_1b + \varepsilon$$

L'interprétation $A_0 = \{a^n b a^p b^p \mid n, p \geq 0\}, A_1 = \{a^n b^n \mid n \geq 0\}$ est solution de S . C'est d'ailleurs l'unique solution de ce système. ♣

Définition 8.3.6 (Fonction définie par un système d'équations) Soit $S = \langle V_T, V_N, d \rangle$ un système d'équations.

La fonction définie par S , notée f_S , est une fonction de l'ensemble $\mathcal{P}(V_T^*)^{V_N}$ des interprétations dans lui-même, définie pour toute interprétation I par : pour tout $A \in V_N$, $f_S(I)(A) = I(d(A))$. ◇

D'après les deux dernières définitions, I est solution de S si et seulement si I est un point fixe de f_S .

On rappelle que, dans ce même exemple, si E est un ensemble d'interprétations, alors l'interprétation $(\bigcup E)$ est définie par :

$$\text{pour tout } A \in V_N, (\bigcup E)(A) = \bigcup \{I(A) \mid I \in E\}.$$

D'après l'exemple 8.2.6, l'ensemble $\mathcal{P}(V_T^*)^{V_N}$ des interprétations ordonné par l'extension suivante de \subset :

$$\text{Soient } I, J \text{ deux interprétations, } I \subset J \text{ si et seulement si pour tout } A \in V_N, I(A) \subset J(A),$$

est un treillis complet. Nous nous servons de cet ordre par la suite.

Nous allons montrer que f_S est une fonction continue, ce qui nous permettra d'utiliser le théorème de Kleene pour caractériser la plus petite solution du système S .

Avant de faire cette preuve, nous avons besoin de quelques propriétés de suites croissantes d'interprétations.

Propriété 8.3.7 (Suite croissante d'interprétations et concaténation) Soient V_T et V_N deux vocabulaires disjoints et V l'union de ces deux vocabulaires. Soit E un ensemble totalement ordonné d'interprétations de V_N sur V_T , autrement dit un sous-ensemble totalement ordonné de $\mathcal{P}(V_T^*)^{V_N}$, avec pour ordre l'extension de \subset définie ci-dessus.

$$\text{Pour tous } x, y \in V^*, \bigcup \{I(x)I(y) \mid I \in E\} = (\bigcup \{I(x) \mid I \in E\})(\bigcup \{I(y) \mid I \in E\})$$

PREUVE. 1. Soit $z \in \bigcup \{I(x)I(y) \mid I \in E\}$.

Montrons que $z \in (\bigcup \{I(x) \mid I \in E\})(\bigcup \{I(y) \mid I \in E\})$.

D'après l'hypothèse, il existe une interprétation $I \in E$ telle que $z \in I(x)I(y)$.

Donc $z = uv$ où $u \in I(x)$ et $v \in I(y)$.

Donc $u \in \bigcup \{I(x) \mid I \in E\}$ et $v \in \bigcup \{I(y) \mid I \in E\}$.

Par suite $z \in (\bigcup \{I(x) \mid I \in E\})(\bigcup \{I(y) \mid I \in E\})$.

2. Soit $z \in (\bigcup \{I(x) \mid I \in E\})(\bigcup \{I(y) \mid I \in E\})$.

D'après l'hypothèse, $z = uv$ avec $u \in \bigcup \{I(x) \mid I \in E\}$ et $v \in \bigcup \{I(y) \mid I \in E\}$.

Donc il existe $I \in E$ telle que $u \in I(x)$ et $J \in E$ telle que $v \in J(y)$. Puisque E est totalement ordonné, I et J sont comparables : soit K la plus grande de ces deux interprétations.

On a $u \in K(x)$ et $v \in K(y)$, Puisque $z = uv$, il en résulte que $z \in K(x)K(y)$ et par suite $z \in \bigcup \{I(x)I(y) \mid I \in E\}$. ■

Propriété 8.3.8 (Suite croissante d'interprétations et mots) Soient V_T et V_N deux vocabulaires disjoints et V l'union de ces deux vocabulaires. Soit E un ensemble totalement ordonné d'interprétations de V_N sur V_T .

Pour tout $x \in V^*$, $\bigcup\{I(x) \mid I \in E\} = (\bigcup E)(x)$

PREUVE. On prouve la propriété par induction sur la longueur de x .

1. Si $x = \varepsilon$, alors $I(x) = \{\varepsilon\}$. Par suite $\bigcup\{I(x) \mid I \in E\} = \{\varepsilon\} = (\bigcup E)(x)$
2. Si $x \in V_T$, alors $I(x) = \{x\}$. Par suite $\bigcup\{I(x) \mid I \in E\} = \{x\} = (\bigcup E)(x)$
3. Soit $x \in V_N$. D'après la définition de $\bigcup E$ rappelée ci-dessus après la définition 8.3.6, on a : $\bigcup\{I(x) \mid I \in E\} = (\bigcup E)(x)$
4. Soit $x \in V^*$ de longueur supérieure à 1.

Supposons que pour tout mot u de longueur inférieure à x , on ait : $\bigcup\{I(u) \mid I \in E\} = (\bigcup E)(u)$

Puisque la longueur de x est supérieure à 1, il existe y, z deux chaînes non vides telles que $x = yz$. On a alors :

$$\begin{aligned}
 \bigcup\{I(x) \mid I \in E\} &= \bigcup\{I(yz) \mid I \in E\} && \text{car } x = yz \\
 &= \bigcup\{I(y)I(z) \mid I \in E\} && \text{car } I(yz) = I(y)I(z) \\
 &= (\bigcup\{I(y) \mid I \in E\})(\bigcup\{I(z) \mid I \in E\}) && \text{d'après la propriété 8.3.7 (car } E \text{ est totalement ordonné)} \\
 &= (\bigcup E)(y)(\bigcup E)(z) && \text{par l'hypothèse d'induction appliquée à } y \text{ et } z \\
 &= (\bigcup E)(yz) && \text{car } \bigcup E \text{ est une interprétation} \\
 &= (\bigcup E)(x) && \text{car } x = yz
 \end{aligned}$$

■

Propriété 8.3.9 (Suite croissante d'interprétations et langages) Soient V_T et V_N deux vocabulaires disjoints et V l'union de ces deux vocabulaires. Soit E un ensemble totalement ordonné d'interprétations de V_N sur V_T .

Pour tout L langage sur V , $\bigcup\{I(L) \mid I \in E\} = (\bigcup E)(L)$

PREUVE. En effet

$x \in \bigcup\{I(L) \mid I \in E\}$

si et seulement si il existe $I \in E$ telle que $x \in I(L)$

si et seulement si il existe $I \in E$, il existe $y \in L$ tels que $x \in I(y)$

si et seulement si il existe $y \in L$, il existe $I \in E$ tels que $x \in I(y)$

si et seulement si il existe $y \in L$, $x \in \bigcup\{I(y) \mid I \in E\}$

si et seulement si il existe $y \in L$, $x \in (\bigcup E)(y)$

si et seulement si $x \in (\bigcup E)(L)$

par définition de \bigcup

car $I(L) = \bigcup\{I(y) \mid y \in L\}$

en permutant les "il existe"

par définition de \bigcup

d'après la propriété 8.3.8

car $(\bigcup E)(L) = \bigcup\{(\bigcup E)(y) \mid y \in L\}$

Propriété 8.3.10 Soient X et Y deux ensembles et f une fonction de l'ensemble Y^X dans lui-même. Soient $Z \subset Y^X$ et $x \in X$.

Alors $\{(f(y))(x) \mid y \in Z\} = \{g(x) \mid g \in \{f(y) \mid y \in Z\}\}$

PREUVE. En effet :

$$u \in \{(f(y))(x) \mid y \in Z\}$$

si et seulement si il existe $y \in Z$ tel que $u = (f(y))(x)$

si et seulement si il existe $y \in Z$ et il existe $g \in Y^X$ tels que $u = g(x)$ et $g = f(y)$

D'après les propriétés de f

si et seulement si il existe $g \in Y^X$ tel que $u = g(x)$ et il existe $y \in Z$ tel que $g = f(y)$

En permutant les quantificateurs

si et seulement si il existe $g \in Y^X$ tel que $u = g(x)$ et $g \in \{f(y) \mid y \in Z\}$

si et seulement si il existe $u \in \{g(x) \mid g \in \{f(y) \mid y \in Z\}\}$

Théorème 8.3.11 *La fonction f_S définie par un système d'équations S est continue.*

PREUVE. Soit $S = \langle V_T, V_N, d \rangle$ un système d'équations.

On montre que :

$$\boxed{\text{Pour tout } E \text{ ensemble totalement ordonné d'interprétations de } V_N \text{ sur } V_T, f_S(\bigcup E) = \bigcup \{f_S(I) \mid I \in E\}}$$

Puisqu'une suite croissante dénombrable est un cas particulier d'ensemble totalement ordonné, d'après la définition de la continuité 8.2.2, la propriété encadrée implique la continuité de f_S .

Pour montrer cette propriété, il suffit de vérifier que pour tout $A \in V_N$, $f_S(\bigcup E)(A) = (\bigcup \{f_S(I) \mid I \in E\})(A)$.

En effet :

$$\begin{aligned} (f_S(\bigcup E))(A) &= (\bigcup E)(d(A)) && \text{D'après la définition 8.3.6 de } f_S \\ &= \bigcup \{I(d(A)) \mid I \in E\} && \text{D'après la propriété 8.3.9 appliquée au langage } d(A) \\ &= \bigcup \{(f_S(I))(A) \mid I \in E\} && \text{D'après la définition 8.3.6 de } f_S \end{aligned}$$

Or, d'après la propriété 8.3.10, $\{(f_S(I))(A) \mid I \in E\} = \{J(A) \mid J \in \{f_S(I) \mid I \in E\}\}$.

On en déduit que $\bigcup \{(f_S(I))(A) \mid I \in E\} = \bigcup \{J(A) \mid J \in \{f_S(I) \mid I \in E\}\}$, d'où

$$\begin{aligned} f_S(\bigcup E)(A) &= \bigcup \{J(A) \mid J \in \{f_S(I) \mid I \in E\}\} \\ &= (\bigcup \{f_S(I) \mid I \in E\})(A) \end{aligned} \quad \text{Car pour tout ensemble } F \text{ d'interprétations, } \bigcup \{J(A) \mid J \in F\} =$$

■

Théorème 8.3.12 *Soit S un système d'équations sur l'ensemble V_T de terminaux et l'ensemble V_N de variables.*

Soit \perp la plus petite interprétation de V_N sur V_T : pour tout $A \in V_N$, $\perp(A) = \emptyset$.

La plus petite solution de S est $\bigcup \{f_S^i(\perp) \mid i \in \mathbb{N}\}$.

PREUVE. D'après le théorème ci-dessus 8.3.11, la fonction f_S est continue. La plus petite solution de S est le plus petit point fixe de f_S . D'après le théorème de Kleene, cette plus petite solution est $\bigcup \{f_S^i(\perp) \mid i \in \mathbb{N}\}$. ■

Exemple 8.3.13

1. Soit S le système de terminaux a, b , de variable A_0 et d'équation $A_0 = aA_0b + \varepsilon$. Alors :

$$\begin{aligned} f_S^1(\perp) &= \langle A_0 \mapsto \{\varepsilon\} \rangle \\ f_S^2(\perp) &= \langle A_0 \mapsto \{\varepsilon, ab\} \rangle \\ f_S^3(\perp) &= \langle A_0 \mapsto \{\varepsilon, ab, aabb\} \rangle \\ f_S^i(\perp) &= \langle A_0 \mapsto \{a^j b^j \mid j < i\} \rangle \\ \bigcup \{f_S^i(\perp) \mid i \in \mathbb{N}\} &= \langle A_0 \mapsto \{a^j b^j \mid j \in \mathbb{N}\} \rangle \end{aligned}$$

La dernière fonction est la plus petite solution (d'ailleurs unique) du système d'équations.

2. Soit S , le système d'équations avec deux terminaux a, b , deux variables A_0, A_1 et les équations :

$$\begin{aligned} A_0 &= aA_0 + bA_1 \\ A_1 &= aA_1b + \varepsilon \end{aligned}$$

On a alors :

$$\begin{aligned} f_S^1(\perp) &= \langle A_0 \mapsto \emptyset, A_1 \mapsto \{\varepsilon\} \rangle \\ f_S^2(\perp) &= \langle A_0 \mapsto \{b\}, A_1 \mapsto \{\varepsilon, ab\} \rangle \\ f_S^3(\perp) &= \langle A_0 \mapsto \{ab, b, bab\}, A_1 \mapsto \{\varepsilon, ab, aabb\} \rangle \\ f_S^4(\perp) &= \langle A_0 \mapsto \{aab, ab, abab, b, bab, baabb\}, A_1 \mapsto \{\varepsilon, ab, aabb\} \rangle \\ f_S^i(\perp) &= \langle A_0 \mapsto \{a^j b a^k b^k \mid j+k < i-1\}, A_1 \mapsto \{a^j b^j \mid j < i\} \rangle \\ \bigcup \{f_S^i(\perp) \mid i \in \mathbb{N}\} &= \langle A_0 \mapsto \{a^j b a^k b^k \mid j, k \in \mathbb{N}\}, A_1 \mapsto \{a^j b^j \mid j \in \mathbb{N}\} \rangle \end{aligned}$$

La dernière fonction est la plus petite solution (d'ailleurs unique) du système d'équations.



4 Équivalence entre langages algébriques et hors-contextes

Définition 8.4.1 Un langage est algébrique si et seulement il est une composante de la plus petite solution d'un système d'équations. Plus formellement soit L un langage sur V_T . Ce langage est algébrique si et seulement si il y a un ensemble de variables V_N , disjoint de V_T , un système d'équations S sur V_T et V_N , de plus petite solution I , et une variable $A \in V_N$ telle que $L = I(A)$. \diamond

Le théorème suivant montre qu'un langage est algébrique si et seulement si il est hors-contexte.

Définition 8.4.2 Soit $S = \langle V_T, V_N, d \rangle$ un système d'équations. Soit $R = \{A \rightarrow \alpha \mid A \in V_N, \alpha \in d(A)\}$ l'ensemble des règles associées au système S . Soit $A \in V_N$.

La grammaire $G_{S,A} = \langle V_T, V_N, R, A \rangle$ est la grammaire hors-contexte associée au système S et à la variable A . \diamond

En pratique le passage d'un système à un ensemble de règles hors-contextes consiste à remplacer le signe $=$ par la flèche \rightarrow et le plus par la barre verticale. Par exemple le système

$$\begin{aligned} A_0 &= aA_0 + bA_1 \\ A_1 &= aA_1b + \varepsilon \end{aligned}$$

est remplacé par les règles

$$\begin{aligned} A_0 &\rightarrow aA_0|bA_1 \\ A_1 &\rightarrow aA_1b|\varepsilon \end{aligned}$$

Théorème 8.4.3 Soit $S = \langle V_T, V_N, d \rangle$ un système d'équations.

La plus petite solution du système est l'interprétation I telle que pour tout $A \in V_N$, $I(A) = L(G_{S,A})$.

Autrement dit la composante de la variable A dans la plus petite solution du système est le langage engendré par la grammaire d'axiome A et dont les règles sont les règles associées au système.

PREUVE. Soit $V = V_T \cup V_N$ et R l'ensemble des règles associées au système S .

1. L'interprétation I est une solution du système S .

Par définition de I , il est clair que pour tout $\alpha \in V^*$, $I(\alpha) = \{x \in V_T^* \mid \alpha \Rightarrow_R^* x\}$.

(a) Pour tout $A \in V_N$, $I(d(A)) \subset I(A)$.

Soit $x \in I(d(A))$. Par définition de $I(d(A))$, il existe $\alpha \in d(A)$ telle que $x \in I(\alpha)$.

Par définition de R , $A \rightarrow \alpha$ est une règle de la grammaire $G_{S,A}$.

Puisque $x \in I(\alpha)$, d'après la propriété de $I(\alpha)$, $\alpha \Rightarrow_R^* x$

Par suite x est un mot engendré par la grammaire $G_{S,A}$, donc $x \in I(A)$.

(b) Pour tout $A \in V_N$, $I(A) \subset I(d(A))$.

Soit $x \in I(A)$. Par définition de I , le mot x est engendré par la grammaire $G_{S,A}$.

Donc il existe α tel que $A \rightarrow \alpha$ est une règle de R et $\alpha \Rightarrow_R^* x$.

Par définition de R , $\alpha \in d(A)$. D'après la propriété de $I(\alpha)$, $x \in I(\alpha)$.

Donc par définition de $I(d(A))$, $x \in I(d(A))$.

2. L'interprétation I est la plus petite solution du système S .

Soit J une autre solution du système. Soit $A \in V_N$, $x \in V_T^*$, n un entier.

Par induction sur n , on montre que pour tout n , si $A \Rightarrow_R^n x$ alors $x \in J(A)$.

Supposons que pour tout p où $p < n$, si $A \Rightarrow_R^p x$ alors $x \in J(A)$: c'est l'hypothèse d'induction.

(a) Soit $n = 0$. Montrons que si $A \Rightarrow_R^n x$ alors $x \in J(A)$.

Pour $n = 0$, on aurait $A = x$ ce qui est impossible car A est une variable et x une chaîne terminale. Donc l'implication est vraie.

(b) Soit $n > 0$. Supposons $A \Rightarrow_R^n x$.

Il y a une règle $A \rightarrow \alpha$ dans R et $\alpha \Rightarrow_R^{n-1} x$.

Soit $B_{(i \mid i \text{ de } 1 \text{ à } p)}$ la suite des symboles de α .

D'après la propriété fondamentale 7.1.1 des grammaires hors-contextes :

Pour i de 1 à p , $B_i \Rightarrow_R^{k_i} x_i$ où $n - 1 = \sum_i^p k_i$ et $x = x_1 \dots x_p$.

Quand $B_i \in V_T$, par définition des dérivations on a $B_i = x_i$ et par définition des interprétations, $J(B_i) = \{B_i\}$, donc $x_i \in J(B_i)$.

Quand $B_i \in V_N$, puisque $k_i < n$, l'hypothèse d'induction est applicable, on a de même $x_i \in J(B_i)$.

Par suite $x_1 \dots x_p \in J(B_1 \dots B_p)$, c'est-à-dire $x \in J(\alpha)$.

Puisque $\alpha \in d(A)$, il résulte que $x \in J(d(A))$ et puisque J est une solution du système S , $J(A) = J(d(A))$, donc $x \in J(A)$.

Par le principe d'induction, pour tout n , si $A \Rightarrow_R^n x$ alors $x \in J(A)$, donc $I \subset J$.
 Par suite I est bien la plus petite solution du système d'équations. ■

5 Transformation de systèmes d'équations

Définition 8.5.1 (Relation entre systèmes d'équations) Soit V_T un ensemble de terminaux et V_N un ensemble de variables. Soient $S = \langle V_T, V_N, d \rangle$ et $S' = \langle V_T, V_N, d' \rangle$ deux systèmes d'équations. S' est *conséquence* de S , si toute solution de S est solution de S' . Ces deux systèmes sont *équivalents* s'ils ont les mêmes solutions. ◇

5.1 Substitution simple

Définition 8.5.2 (Substitution simple) Le système S' est obtenu par substitution simple sur le système S s'il est obtenu en remplaçant dans une définition de S , *une* occurrence d'une variable par sa définition.

Plus formellement soient $S = \langle V_T, V_N, d \rangle$ et $S' = \langle V_T, V_N, d' \rangle$ deux systèmes. S' est obtenu par substitution simple sur S , s'il existe $A, B \in V_N$, $\alpha B \beta \in d(A)$ ¹ tel que :
 d' est identique à d sauf pour A et $d'(A) = (d(A) - \{\alpha B \beta\}) \cup \alpha d(B) \beta$. ◇

On va montrer que cette substitution change un système en un autre équivalent. On a besoin dans cette preuve d'une propriété sur les langages, que nous montrons ci-dessous.

Propriété 8.5.3 Soient L, M, N, P, Q , cinq langages sur le même vocabulaire vérifiant les égalités

$$P = L \cup M \cup QN$$

$$Q = L \cup M \cup PN$$

Il en résulte que $P = Q$

PREUVE. 1. Supposons $\varepsilon \in M$ et $\varepsilon \in N$.

Puisque $\varepsilon \in M$ et $\varepsilon \in N$, on a $Q \subset M \cup QN$. D'après la première égalité $M \cup QN \subset P$. Donc $Q \subset P$.

De façon analogue, grâce à la deuxième égalité, on a $P \subset Q$, donc $P = Q$.

2. Supposons $\varepsilon \notin M$ ou $\varepsilon \notin N$.

On montre par induction sur x que $x \in P$ si et seulement si $x \in Q$.

(a) Supposons $x = \varepsilon$.

Supposons $x \in P$. Puisque tout mot de $M \cup QN$ n'est pas vide, d'après la première égalité $x \in L$ et d'après la deuxième $x \in Q$.

Réciproquement, de façon analogue si $x \in Q$ alors $x \in P$.

1. Rappelons que $d(A)$ est la définition de A

(b) Supposons $x \neq \varepsilon$.

On admet l'hypothèse d'induction que tout mot, de longueur inférieure à celle de x , est élément de P si et seulement si il est élément de Q .

i. Supposons $x \in L$. Des deux égalités, il résulte immédiatement que $x \in P$ si et seulement si $x \in Q$.

ii. Supposons $x \notin L$.

Supposons $x \in P$.

D'après la première équation $x \in MQN$, donc $x = uvw$ avec $u \in M, v \in Q, w \in N$.

Puisque $\varepsilon \notin M$ ou $\varepsilon \notin N$, la longueur de v est inférieure à celle de x , donc par hypothèse d'induction $v \in P$.

Par suite $x = uvw \in MPQ$, donc d'après la deuxième égalité $x \in Q$.

De façon analogue, $x \in Q$ implique $x \in P$.

Donc $x \in P$ si et seulement si $x \in Q$. Par le principe d'induction, c'est vrai pour tout mot x , donc $P = Q$. ■

Théorème 8.5.4 Soient deux systèmes d'équations dont l'un est obtenu par une substitution simple appliquée à l'autre. Ces deux systèmes sont équivalents.

PREUVE. Soient $S = \langle V_T, V_N, d \rangle$ et $S' = \langle V_T, V_N, d' \rangle$ deux systèmes. S' est obtenu par substitution simple sur S . Donc il existe $A, B \in V_N, \alpha B \beta \in d(A)$ tel que :
 d' est identique à d sauf pour A et $d'(A) = (d(A) - \{\alpha B \beta\}) \cup \alpha d(B) \beta$.

1. Soit I solution de S . Montrons que c'est également une solution de S' .

Il suffit de vérifier que $I(A) = I(d'(A))$.

$$\begin{aligned}
 I(A) &= I(d(A)) && \text{Car } I \text{ est solution de } S \\
 &= I(d(A) - \{\alpha B \beta\}) \cup I(\alpha B \beta) && \text{Car } d(A) = (d(A) - \{\alpha B \beta\}) \cup \{\alpha B \beta\} \\
 &= I(d(A) - \{\alpha B \beta\}) \cup I(\alpha d(B) \beta) && \text{Car } I(B) = I(d(B)) \text{ puisque } I \text{ est solution de } S \\
 &= I(d'(A)) && \text{Par définition de } d'(A)
 \end{aligned}$$

2. Soit I une solution de S' . Montrons que c'est également une solution de S .

Il suffit de vérifier que $I(A) = I(d(A))$.

Puisque I est solution de S' , $I(A) = I(d'(A)) = I(d(A) - \{\alpha B \beta\}) \cup I(\alpha d(B) \beta)$.

(a) Supposons $A \neq B$.

Puisque seule l'équation définissant A est modifiée, $d(B) = d'(B)$.

Puisque I est solution de S' , $I(B) = I(d'(B)) = I(d(B))$.

Donc en remplaçant $I(d(B))$ par la valeur qui lui est égale $I(B)$ dans la valeur de $I(A)$, on obtient :

$$I(A) = I(d(A) - \{\alpha B \beta\}) \cup I(\alpha B \beta) = I(d(A)).$$

Donc I est solution de S .

(b) Supposons $A = B$.

On a alors :

$$I(A) = I(d(A) - \{\alpha A \beta\}) \cup I(\alpha)I(d(A))I(\beta) \quad \text{Car } I \text{ est solution de } S' \text{ et } d'(A) = (d(A) - \{\alpha A \beta\}) \cup$$

$$I(d(A)) = I(d(A) - \{\alpha A \beta\}) \cup I(\alpha)I(A)I(\beta) \quad \text{Car } d(A) = (d(A) - \{\alpha A \beta\}) \cup \{\alpha A \beta\}$$

D'après la proposition 8.5.3 en instanciant L, M, N, P, Q respectivement par les ensembles $I(d(A) - \{\alpha A \beta\}), I(\alpha), I(\beta), I(A)$ et $I(d(A))$, on obtient $I(A) = I(d(A))$

Donc I est solution de S . ■

5.2 Substitution générale

Définition 8.5.5 (Substitution générale) Le système S' est obtenu par substitution générale sur le système S s'il est obtenu en remplaçant dans les définitions de S , des occurrences de variables par leur définition.

Plus formellement soient $S = \langle V_T, V_N, d \rangle$ et $S' = \langle V_T, V_N, d' \rangle$ deux systèmes, S' est obtenu par substitution générale sur le système S si :

Pour tout $A \in V_N$ $d'(A) = \bigcup \{E_\alpha \mid \alpha \in d(A)\}$ et

tout E_α est de la forme $\beta_0 d(B_1) \beta_1 \dots d(B_n) \beta_n$ où $n \geq 0, B_1, \dots, B_n \in V_N$ et $\alpha = \beta_0 B_1 \beta_1 \dots B_n \beta_n$. ◊

Notons qu'une substitution simple est un cas particulier d'une substitution générale.

La substitution générale, contrairement à la substitution simple, ne préserve pas nécessairement l'ensemble des solutions d'un système d'équations. Cependant, comme on le prouve ci-dessous, elle préserve la plus petite solution.

Considérons l'exemple suivant. Soit le système S de terminaux a, b et de variables A, B et d'équations :

$$A = B + a$$

$$B = A + b$$

En remplaçant à droite des équations, A et B par leur définition, on obtient le système S' suivant :

$$A = A + a + b$$

$$B = B + a + b$$

L'interprétation $A = a + b + \varepsilon, B = a + b$ est solution de S' mais n'est pas une solution de S . Donc la substitution générale ne préserve pas toutes les solutions.

Par contre $A = a + b, B = a + b$ est la plus petite solution des deux systèmes.

Propriété 8.5.6 Soit S' un système obtenu par substitution générale sur le système S . Toute solution de S est solution de S' .

PREUVE. Nous reprenons, pour les deux systèmes, les notations de la définition 8.5.5.

Soit $E_\alpha = \beta_0 d(B_1) \beta_1 \dots d(B_n) \beta_n$ où $n \geq 0, B_1, \dots, B_n \in V_N$ et $\alpha = \beta_0 B_1 \beta_1 \dots B_n \beta_n$.

Si I est une solution de S , alors pour tout i de 1 à n , $I(B_i) = I(d(B_i))$ et par suite $I(E_\alpha) = I(\alpha)$.

Donc pour tout $A \in V_N$, par définition de $d'(A)$, $I(d(A)) = I(d'(A))$.

Puisque I est solution de S , pour tout $A \in V_N$, $I(A) = I(d(A))$, donc aussi $I(A) = I(d'(A))$.

Par suite I est aussi solution de S' . ■

Propriété 8.5.7 Soit S' un système obtenu par substitution générale sur le système S . La plus petite solution de S est inférieure ou égale à la plus petite solution de S' pour l'ordre sur les interprétations défini comme l'extension de l'ordre de l'inclusion entre ensembles.

PREUVE. Nous reprenons, pour les deux systèmes, les notations de la définition 8.5.5.

Soient respectivement I et I' les plus petites solutions de S et S' .

Rappelons que f_S et $f_{S'}$ sont les fonctions définies respectivement par S et S' .

D'après le théorème 8.3.11 ces deux fonctions sont continues, donc d'après la preuve du théorème de Kleene les suites $f_S^i(\perp)$ et $f_{S'}^i(\perp)$ sont croissantes, et d'après le théorème 8.3.12, $I = \bigcup \{f_S^i(\perp) \mid i \in \mathbb{N}\}$, et $I' = \bigcup \{f_{S'}^i(\perp) \mid i \in \mathbb{N}\}$.

1. Soit I une interprétation telle que $I \subset f_S(I)$. Montrons que $f_S(I) \subset f_{S'}(I)$.
 Puisque $I \subset f_S(I)$, par définition de f_S , pour tout $A \in V_N$, $I(A) \subset I(d(A))$.
 Soit A une variable et x une chaîne terminale telle que $x \in (f_S(I))(A)$. Il suffit de montrer que $x \in (f_{S'}(I))(A)$.
 Puisque $x \in (f_S(I))(A) = I(d(A))$, il existe $\alpha \in d(A)$ tel que $x \in I(\alpha)$.
 Puisque S' est obtenue par substitution générale, on a :
 $\alpha = \beta_0 B_1 \beta_1 \dots B_n \beta_n$ où pour i de 1 à n , B_i est une variable et $\beta_0 d(B_1) \beta_1 \dots d(B_n) \beta_n \subset d'(A)$.
 Puisque pour tout $A \in V_N$, $I(A) \subset I(d(A))$, il en résulte que pour i de 1 à n , $I(B_i) \subset I(d(B_i))$.
 Puisque la concaténation de langages est croissante, on en déduit que $I(\alpha) \subset I(\beta_0 d(B_1) \beta_1 \dots d(B_n) \beta_n)$.
 Puisque $\beta_0 d(B_1) \beta_1 \dots d(B_n) \beta_n \subset d'(A)$, nous concluons que $I(\alpha) \subset I(d'(A))$.
 Comme $x \in I(\alpha)$, on a $x \in I(d'(A))$; de plus, par définition de $f_{S'}$, $I(d'(A)) = (f_{S'}(I))(A)$, donc $x \in (f_{S'}(I))(A)$.
 2. Montrons par induction sur i que pour tout $i \in \mathbb{N}$, $f_S^i(\perp) \subset f_{S'}^i(\perp)$.
 Pour $i = 0$ c'est trivial car $f_S^0(\perp) = \perp = f_{S'}^0(\perp)$.
 Supposons que $f_S^i(\perp) \subset f_{S'}^i(\perp)$.
 On a rappelé ci-dessus que $f_S^i(\perp)$ est une suite croissante, donc $f_S^i(\perp) \subset f_S(f_S^i(\perp))$.
 Par suite on peut appliquer le résultat 1 du premier cas à l'interprétation $f_S^i(\perp)$, donc :
 (*) $f_S(f_S^i(\perp)) \subset f_{S'}(f_S^i(\perp))$.
 Puisque toute fonction continue est croissante d'après la propriété 8.2.3, $f_{S'}$ est croissante. Comme par hypothèse d'induction $f_S^i(\perp) \subset f_{S'}^i(\perp)$, il résulte de la croissance de $f_{S'}$ que :
 (**) $f_{S'}(f_S^i(\perp)) \subset f_{S'}(f_{S'}^i(\perp))$.
 Donc d'après (*) et (**), on a $f_S^{i+1}(\perp) \subset f_{S'}^{i+1}(\perp)$.
- Puisque pour tout $i \in \mathbb{N}$, $f_S^i(\perp) \subset f_{S'}^i(\perp)$, que I est le plus petit majorant de la suite $f_S^i(\perp)$ et I' le plus petit majorant de la suite $f_{S'}^i(\perp)$, il en résulte que $I \subset I'$. ■

Théorème 8.5.8 Soit S' un système obtenu par substitution générale sur le système S . Les plus petites solutions de S et de S' sont identiques.

PREUVE. Soient respectivement I et I' les plus petites solutions de S et S' .
D'après la propriété 8.5.6, I est aussi solution de S' donc $I' \subset I$.
D'après la propriété 8.5.7, $I \subset I'$. Donc $I = I'$. ■

Chapitre 9

Transformations des grammaires hors-contexte

1 Rappels du chapitre précédent

Dans ce chapitre on présente des transformations de grammaires, préservant les langages engendrés et utiles pour rédiger des programmes efficaces d'analyse, c'est-à-dire des programmes pour reconnaître si un mot est engendré par une grammaire.

La justification de certaines de ces transformations repose sur des résultats du chapitre précédent qui seront brièvement rappelés sous une forme qui devrait éviter la lecture de ce chapitre précédent à ceux qui souhaitent dans un premier temps faire l'impasse sur ces preuves.

1.1 Grammaires et équations

Une grammaire hors-contexte $G = \langle V_T, V_N, S, R \rangle$ peut être vue comme un système d'équations, avec une équation par non-terminal. Soit $V = V_T \cup V_N$.

Soit A un non-terminal (que l'on appelle aussi une variable) et soit $A \rightarrow \alpha_1 | \alpha_2 | \dots | \alpha_n$ où $n > 0$ et $\alpha_1, \alpha_2, \dots, \alpha_n \in V^*$ l'ensemble des règles d'entête A . Cet ensemble de règles est vu comme une équation définissant A que l'on l'écrit $A = \alpha_1 + \alpha_2 + \dots + \alpha_n$.

Un système d'équations est la donnée d'un triplet $\langle V_T, V_N, d \rangle$, où V_T est l'ensemble des terminaux, V_N est l'ensemble des variables, d est l'ensemble des équations entre une variable et une somme de mots dont les lettres sont des terminaux ou des variables.

Les *valeurs des variables* sont des langages sur le vocabulaire terminal V_T .

Une *interprétation du système* est une suite d'égalités entre les variables du système et des valeurs.

Une interprétation est une *solution* du système, si les valeurs obtenues en remplaçant chaque variable par sa valeur dans les deux membres de chaque équation du système, sont égales.

On ordonne ainsi les interprétations : soient I et J deux interprétations d'un système, I est inférieure ou égale à J (ce qu'on note $I \subset J$) si et seulement si pour toute variable A du système la valeur de A donnée par I est un sous-ensemble de la valeur de A donnée par J .

Exemple 9.1.1 Soit la grammaire G de vocabulaire terminal a, b , de vocabulaire non-terminal A, B , d'axiome A et de règles :

$$A \rightarrow aA|B$$

$$B \rightarrow \varepsilon|aBb$$

On peut voir G comme un système d'équations :

$$A = aA + B$$

$$B = \varepsilon + aAb$$

L'interprétation $A = \{ab, a, aba\}, B = \{b^n a \mid n \in \mathbb{N}\}$ n'est pas une solution du système.

Par contre l'interprétation $A = \{a^n b^p \mid n \geq p\}, B = \{a^n b^n \mid n \geq 0\}$ est une solution de ce système d'équations, qui n'a d'ailleurs qu'une seule solution. ♣

Il y a des systèmes d'équations qui ont plus qu'une solution. Par exemple le système $A = A + a + b$ de variable A et de terminaux a, b possède une plus petite solution $A = \{a, b\}$. Mais $A = L$ où L est un langage quelconque tel que $a, b \in L$ est aussi une solution.

La relation entre les solutions et les grammaires est résumée par le résultat suivant.

Théorème 9.1.2 Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte. Une interprétation est la plus petite solution du système associé à la grammaire si et seulement si pour tout non-terminal A , la valeur de A dans l'interprétation est le langage engendré par le non-terminal A avec les règles de G .

Pour les courageux, qui veulent lire la preuve, voir le théorème 8.4.3 du chapitre 10.

1.2 Des substitutions, qui préservent les langages

Définition 9.1.3 Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte et $V = V_T \cup V_N$. On appelle définition du non-terminal A et l'on note par $d(A)$ l'ensemble des parties droites des règles d'entête A .

La grammaire $G' = \langle V_T, V_N, S, R' \rangle$ est obtenue par substitution générale sur G , si les règles de G' sont obtenues en remplaçant dans les parties droites de règles des variables par leurs définitions.

Plus précisément pour chaque règle $A \rightarrow \alpha$ de la grammaire G , il existe $n \geq 0$, des mots $\beta_0, \beta_1, \dots, \beta_n$ de V^* , des variables $B_1, \dots, B_n \in V_N$ tel que

$$\text{— } \alpha = \beta_0 B_1 \beta_1 \dots B_n \beta_n$$

$$\text{— } d'(A) \text{ la définition de } A \text{ dans } R' \text{ est l'ensemble } \beta_0 d(B_1) \beta_1 \dots d(B_n) \beta_n \quad \diamond$$

Théorème 9.1.4 Soit G' obtenue par substitution générale sur la grammaire G . Les deux grammaires sont équivalentes.

Pour les courageux, qui veulent lire la preuve, voir le théorème 8.5.8 du chapitre 10.

1.3 Treillis fini et calcul de points fixes

On rappelle qu'un treillis est un ensemble ordonné dans lequel tout sous-ensemble fini non vide a une plus petite borne supérieure et une plus petite borne inférieure.

Théorème 9.1.5 *Soit E un treillis fini ayant un plus petit élément \perp . Soit n le nombre d'éléments de E .*

Soit $f : E \mapsto E$ une fonction croissante.

La suite $f^i(\perp)$ est une suite croissante.

Le plus petit point fixe de f est $f^j(\perp)$ où j est le plus petit entier tel que $j < n$ et $f^j(\perp) = f^{j+1}(\perp)$.

Pour les courageux, qui veulent lire la preuve, voir le paragraphe 2.3.

2 Réduction d'une grammaire hors-contexte

Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte. Soit $V = V_T \cup V_N$.

2.1 Calcul de l'ensemble des symboles productifs

Le non-terminal $A \in V_N$ est productif si et seulement si il existe une chaîne terminale x telle que $A \Rightarrow^* x$. Un non-terminal improductif ne figure dans aucune dérivation dont l'extrémité est une chaîne terminale.

L'ensemble des symboles productifs peut aussi être défini par induction comme étant le plus petit ensemble de non-terminaux tel que A est productif si et seulement si il y a une règle $A \rightarrow \alpha$ dans la grammaire, telle que toutes les variables de α sont productives. Prouvons que ces deux manières de définir l'ensemble des productifs sont équivalentes.

Théorème 9.2.1 *Soit $f : \mathcal{P}(V_N) \mapsto \mathcal{P}(V_N)$ la fonction suivante :*

Pour tout $E \subset V_N$, $f(E) = \{A \in V_N \mid \exists \alpha \in (V_T \cup E)^ \text{ tel que } A \rightarrow \alpha \in R\}$.*

L'ensemble des symboles productifs est le plus petit point fixe de f

PREUVE. Soit P l'ensemble des symboles productifs.

1. L'ensemble P des symboles productifs vérifie $f(P) = P$.
 - (a) Soit $A \in f(P)$. Donc il existe une règle $A \rightarrow \alpha$ dont tous les symboles de α sont des terminaux ou produisent des chaînes terminales. Par composition des dérivations A produit une chaîne terminale, donc $A \in P$.
 - (b) Soit $A \in P$. Il existe α et une chaîne terminale x tel que $A \rightarrow \alpha \in R$ et $\alpha \Rightarrow^* x$. Par décomposition des dérivations, chaque symbole de α est un terminal ou un non-terminal produisant une chaîne terminale. Donc $\alpha \in (V_T \cup P)^*$ et par suite $A \in f(P)$.

2. P est la plus petite solution.

Soit Q une solution quelconque. Il suffit de vérifier que $P \subset Q$.

Montrons par induction sur i que :

(*) pour tout non-terminal A et toute chaîne terminale x tel que $A \Rightarrow^i x$, on a $A \in Q$.

Pour $i = 0$, la propriété est vérifiée car $A \Rightarrow^i x$ est impossible. Soit $i > 0$. Supposons, par induction, que pour tout $j < i$, pour tout non-terminal A et toute chaîne terminale x tel que $A \Rightarrow^j x$ on a $A \in Q$.

Soit le non-terminal A et la chaîne terminale x tels que $A \Rightarrow^i x$.

Puisque $i > 0$, il existe α tel que $A \rightarrow \alpha \in R$ et $\alpha \Rightarrow^{i-1} x$.

Par décomposition des dérivations, chaque non-terminal de α engendre en au plus $i - 1$ pas une chaîne terminale, donc par hypothèse d'induction est élément de Q .

Par suite $\alpha \in (V_T \cup Q)^*$, donc $A \in f(Q)$. Puisque $Q = f(Q)$, on a $A \in Q$.

Par application du principe d'induction, la propriété (*) est vraie pour tout entier i .

Soit $A \in P$. Il existe i et une chaîne terminale x telle que $A \Rightarrow^i x$.

D'après la propriété (*), vraie pour tout i , on a $A \in Q$ ■

Le théorème ci-dessus nous donne un moyen de calculer l'ensemble P des non-terminaux productifs. En effet f est une fonction croissante sur le treillis fini $\mathcal{P}(V_N)$, ordonné par l'inclusion et dont le plus petit élément est \emptyset . Puisque P est le plus petit point fixe de f , d'après le théorème 9.1.5, $P = f^j(\emptyset)$ où j est le plus petit entier tel que $f^j(\emptyset) = f^{j+1}(\emptyset)$.

D'après ce même théorème, $f^i(\emptyset)$ est une suite croissante d'ensembles. Soit n le nombre de non-terminaux, puisque $f^i(\emptyset)$ est une suite croissante, on a $j < n$.

Donc la valeur de P est obtenue en appliquant au plus n fois la fonction f .

Exemple 9.2.2 Soit la grammaire ayant un terminal a , les non-terminaux A, B, C, D, E et de règles $A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow a, D \rightarrow E, E \rightarrow EA$.

On calcule la suite $f^i(\emptyset)$ jusqu'à la stabilité de la suite :

$$f^0(\emptyset) = \emptyset$$

$$f^1(\emptyset) = \{D\}$$

$$f^2(\emptyset) = \{D, C\}$$

$$f^3(\emptyset) = \{D, C, B\}$$

$$f^4(\emptyset) = \{D, C, B, A\}$$

$$f^5(\emptyset) = \{D, C, B, A\}$$

Donc le non-terminal E est improductif et les autres terminaux sont productifs. ♣

Exercice 9.2.3 Montrer que $f^i(\emptyset)$ est l'ensemble des symboles produisant une chaîne terminale avec un arbre de dérivation de hauteur au plus égale à i .

2.2 Calcul de l'ensemble de l'ensemble des symboles accessibles

Un non-terminal B est *accessible* à partir du non-terminal A si et seulement si il existe deux chaînes $\alpha, \beta \in V^*$ telles que $A \Rightarrow^* \alpha B \beta$.

Un non-terminal est accessible relativement à une grammaire si et seulement si il est accessible

à partir de l'axiome de la grammaire.

Évidemment un non-terminal inaccessible ne figure dans aucune dérivation à partir de l'axiome, donc est inutile.

Soit d la relation de dépendance sur V_N définie par :

AdB si et seulement il existe deux chaînes $\alpha, \beta \in V^*$ telles que $A \rightarrow \alpha B \beta \in R$.

Notons d^* la fermeture reflexive et transitive de la relation d . Il est évident que B est *accessible* à partir du non-terminal A si et seulement si $A d^* B$.

Comme la relation d est une relation sur l'ensemble fini des non-terminaux, il y a des algorithmes pour la calculer.

Nous donnons ci-dessus l'algorithme de Roy-Warshall, qui est applicable à *tout* ensemble fini.

Soit A_i où i va de 1 à n , une liste sans répétition des éléments de V_N .

Notons d_k la relation suivante : $A_i d_k A_j$ si et seulement il y a un chemin (pour la relation d) entre A_i et A_j dont les sommets *intermédiaires* sont d'indice au plus égal à k .

Il est (presque) évident que ces relations vérifient :

1. $A_i d_0 A_j$ si et seulement si $A_i d A_j$ ou $i = j$
2. $d^* = d_n$ car n est le plus grand indice possible
3. Soit $k > 0$. $A_i d_k A_j$ si et seulement si $A_i d_{k-1} A_j$ ou $(A_i d_{k-1} A_k$ et $A_k d_{k-1} A_j)$ car s'il y a un chemin entre A_i et A_j passant par des sommets d'indice au plus k , il y a un chemin entre A_i et A_j passant une seule fois par un tel sommet.

Cela donne un calcul de d^* en un temps de l'ordre de n^3 , car il suffit de calculer chacune des n relations d_k , où $k > 0$, et que le calcul de d_k nécessite de calculer les n^2 couples de cette relation.

2.3 Construction d'une grammaire réduite

Définition 9.2.4 Une grammaire hors-contexte est réduite si et seulement si tous ses symboles sont productifs et accessibles. \diamond

Théorème 9.2.5 Soit G une grammaire hors-contexte telle que $L(G) \neq \emptyset$. On peut construire une grammaire réduite équivalente à G .

PREUVE. Soit $G = \langle V_T, V_N, S, R \rangle$.

1. Élimination des improductifs.

Soit P l'ensemble des symboles productifs de la grammaire.

Puisque $L(G)$ n'est pas vide, P comporte au moins l'axiome S .

Soit R' l'ensemble des règles de G ne comportant pas de symboles improductifs et soit $G' = \langle V_T, P, S, R' \rangle$.

On montre aisément que pour tout $A \in P$ et toute chaîne terminale $x : A \Rightarrow_G^* x$ si et seulement si $A \Rightarrow_{G'}^* x$.

Puisque S est l'axiome de G' , il en résulte que les deux grammaires sont équivalentes.

Puisque P est l'ensemble des non-terminaux de G' , il en résulte que tous les non-terminaux de G' sont productifs.

2. Élimination des inaccessibles.

Soit Q l'ensemble des symboles accessibles de la grammaire G' .

Soit R'' l'ensemble des règles de G' ne comportant pas de symboles inaccessibles et soit $G'' = \langle V_T, Q, S, R'' \rangle$.

On montre aisément les propriétés suivantes de G''

(a) pour tout $\alpha \in V^*$, si $S \Rightarrow_{G'}^* \alpha$ alors $S \Rightarrow_{G''}^* \alpha$.

La preuve est une preuve facile mais fastidieuse sur la longueur des dérivations.

(b) pour tout $\alpha \in V^*$, si $S \Rightarrow_{G''}^* \alpha$ alors $S \Rightarrow_{G'}^* \alpha$.

Ce résultat résulte de ce que toute règle de G'' est une règle de G' .

(c) pour tout $A \in Q$ et toute chaîne terminale x , si $A \Rightarrow_{G'} x$ alors $A \Rightarrow_{G''} x$.

La preuve est une preuve facile mais fastidieuse sur la longueur des dérivations.

D'après la propriété 2a, tous les symboles accessibles dans G' , le sont dans G'' , donc tous les non-terminaux de G'' sont accessibles.

D'après la propriété 2c et le fait que tous les non-terminaux de G' sont productifs, il résulte que tous les non-terminaux de G'' sont productifs.

Donc la grammaire G'' est réduite.

Des propriétés 2a et 2b, il résulte que les grammaires G' et G'' sont équivalentes.

Puisque les grammaires G et G' sont équivalentes, il en résulte que les grammaires G et G'' sont équivalentes.

Puisque l'on a vu au paragraphe 2.1 comment calculer P , l'ensemble des symboles productifs de G , et que l'on a vu au paragraphe 2.2 comment calculer Q , l'ensemble des symboles accessibles de G' , on sait comment construire une grammaire G'' réduite et équivalente à G . ■

Remarque On peut résumer ainsi la transformation d'une grammaire en une grammaire réduite équivalente, enlever d'abord les non-terminaux improductifs (et les règles qui en comportent) puis enlever les non-terminaux inaccessibles (et les règles qui en comportent). Si on procède dans l'ordre inverse, il n'est pas certain que la grammaire obtenue est réduite comme le montre l'exemple suivant.

Soit G la grammaire de terminaux a, b, c , de non-terminaux S, A, C, D, E , d'axiome S et de règles $S \rightarrow A, S \rightarrow a, A \rightarrow CD, C \rightarrow c, D \rightarrow A, E \rightarrow b$.

Le non-terminal E est le seul symbole inaccessible. En l'enlevant, on obtient la grammaire G' de non-terminaux S, A, C, D , d'axiome S et de règles $S \rightarrow A, S \rightarrow a, A \rightarrow CD, C \rightarrow c, D \rightarrow A$.

Les non-terminaux A, D , sont les seuls symboles improductifs. En les enlevant, on obtient la grammaire G'' de non-terminaux S, C , d'axiome S et de règles $S \rightarrow a, C \rightarrow c$.

La grammaire obtenue comporte le symbole inaccessible C : elle n'est pas réduite.

3 Élimination des ϵ -règles

Savoir éliminer les ϵ -règles n'a de nos jours aucun intérêt pratique, car les analyseurs savent les utiliser. Mais cela permet de justifier ce qui avait été dit sans preuve au chapitre 6 : tout

langage hors-contexte ne comportant pas la chaîne vide, est sous-contexte. En effet les règles hors-contextes, dont la partie droite n'est pas vide, sont aussi des règles sous-contextes, comme on le voit dans la classification des grammaires, page 70 (chapitre 6).

Théorème 9.3.1 Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte. On peut construire une grammaire hors-contexte $G' = \langle V_T, V_N, S, R' \rangle$ sans ε -règle telle que $L(G') = L(G) - \{\varepsilon\}$.

PREUVE. On montre d'abord comment construire l'ensemble des non-terminaux engendrant la chaîne vide, puis on détaille la construction de G' , enfin on montre que $L(G') \subset L(G) - \{\varepsilon\}$ et $L(G) - \{\varepsilon\} \subset L(G')$.

1. Construction de E l'ensemble des non-terminaux engendrant la chaîne vide.

Cette construction est analogue à celle de l'ensemble de symboles productifs étudiée au paragraphe 2.1, aussi nous laissons au lecteur les justifications plus précises de cette construction.

Soit $f : \mathcal{P}(V_N) \mapsto \mathcal{P}(V_N)$ la fonction suivante :

Pour tout $X \subset V_N$, $f(X) = \{A \in V_N \mid \exists \alpha \text{ tel que } A \rightarrow \alpha \text{ et } \alpha \in X^*\}$.

Il est clair que f est croissante et que E est le plus petit point fixe de f .

Donc $E = f^j(\emptyset)$ où j est le plus petit indice tel que $f^j(\emptyset) = f^{j+1}(\emptyset)$.

De plus, puisque la suite $f^i(\emptyset)$ est croissante, j est inférieur au nombre de non-terminaux de la grammaire.

2. Construction de R' .

R' est obtenu en effaçant, dans les parties droites des règles de G , de toutes les manières possibles les non-terminaux produisant la chaîne vide.

Plus formellement R' est l'ensemble de règles $A \rightarrow B_1 \dots B_p$ où

— $p > 0$ et pour i de 1 à p , $B_i \in V$

— il existe une règle de G s'écrivant $A \rightarrow \alpha_1 B_1 \dots \alpha_p B_p \alpha_{p+1}$ où $\alpha_1, \dots, \alpha_p, \alpha_{p+1} \in E^*$.

3. $L(G') \subset L(G) - \{\varepsilon\}$.

Puisque G' est sans ε -règle, la chaîne vide n'est pas élément de $L(G')$, aussi il suffit de montrer, par induction sur i , que :

Pour tout non-terminal A et toute chaîne terminale x , si $A \Rightarrow_{G'}^i x$ alors $A \Rightarrow_G^* x$
--

Pour $i = 0$, la propriété est "trivialement" vraie, car il est impossible que $A = x$.

Supposons $i > 0$ et admettons l'hypothèse d'induction, que la propriété est vérifiée pour toute dérivation de longueur inférieure à i .

Supposons $A \Rightarrow_{G'}^i x$. Puisque $i > 0$, cette dérivation s'écrit :

$A \Rightarrow_{G'} B_1 \dots B_p \Rightarrow_{G'}^{i-1} x$ où $B_1, \dots, B_p \in V$.

En utilisant la propriété fondamentale des langages hors-contexte (voir chapitre 7, théorème 7.1.1), ainsi que l'hypothèse d'induction, il existe des chaînes terminales x_1, \dots, x_p telles que

— $x = x_1 \dots x_p$

— pour j de 1 à p , $B_j \Rightarrow_G^* x_j$

Par définition de G' , il existe $\alpha_1, \dots, \alpha_p, \alpha_{p+1} \in E^*$ telles que $A \rightarrow \alpha_1 B_1 \dots \alpha_p B_p \alpha_{p+1}$ est une règle de G .

Puisque E est l'ensemble des non-terminaux produisant la chaîne vide dans la grammaire G , toutes les chaînes $\alpha_1, \dots, \alpha_p, \alpha_{p+1}$ produisent la chaîne vide dans cette grammaire et par composition des dérivations, on a : $A \Rightarrow_G^* x$.

4. $L(G) - \{\epsilon\} \subset L(G')$.

Il suffit de montrer, par induction sur i , que :

Pour tout non-terminal A et toute chaîne terminale non vide x , si $A \Rightarrow_G^i x$ alors $A \Rightarrow_{G'}^* x$

Pour $i = 0$, la propriété est "trivialement" vraie.

Supposons $i > 0$ et admettons l'hypothèse d'induction, que la propriété est vérifiée pour toute dérivation de longueur inférieure à i .

Supposons $A \Rightarrow_G^i x$. Puisque $i > 0$, cette dérivation s'écrit :

$A \Rightarrow_G B_1 \dots B_p \Rightarrow_G^{i-1} x$ où $B_1, \dots, B_p \in V$.

D'après la propriété fondamentale des langages hors-contexte, il existe pour j de 1 à p , des entiers n_j et des chaînes x_j tel que :

(*) $B_j \Rightarrow_G^{n_j} x_j$ et $x = x_1 \dots x_p$ et $i - 1 = n_1 + \dots + n_p$.

Soient k_1, \dots, k_q la suite croissante des indices des chaînes x_j non vides.

Par définition des règles de G' et de ces indices :

(**) $A \rightarrow B_{k_1} \dots B_{k_q}$ est une règle de G' ,

obtenue en supprimant les non-terminaux B_j qui engendrent des chaînes vides.

Puisque, d'après la propriété (*), pour tout j de 1 à p , $n_j < i$, il résulte, en appliquant l'hypothèse d'induction à cette même propriété que :

(***) Pour j de 1 à q , $B_{k_j} \Rightarrow_{G'}^* x_{k_j}$

D'après les propriétés (*),(**) et (***) , en composant les dérivations, on a : $A \Rightarrow_{G'}^* x$.

Par induction la propriété encadrée ci-dessus est donc vérifiée pour tout i . ■

Exemple 9.3.2 Soit la grammaire de terminaux a, b , de non-terminaux S, A, B, C , d'axiome S et de règles :

$S \rightarrow ABC, A \rightarrow BB|\epsilon, B \rightarrow CC|a, C \rightarrow AA|b$

Tous les non-terminaux produisent la chaîne vide, donc la grammaire équivalente sans ϵ -règles a comme règles :

$S \rightarrow ABC|BC|AC|AB|A|B|C$

$A \rightarrow BB|B$

$B \rightarrow CC|C|a$

$C \rightarrow AA|A|b$

Cet exemple met en évidence que l'enlèvement des règles produisant la chaîne vide peut augmenter considérablement le nombre de règles. ♣

Théorème 9.3.3 Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte. Si $L(G)$ n'est pas vide, on peut construire une grammaire hors-contexte $G' = \langle V_T, V_N, S, R' \rangle$ réduite et sans ϵ -règle telle que $L(G') = L(G) - \{\epsilon\}$.

PREUVE. On élimine d'abord les ε -règles, comme il a été indiqué ci-dessus, on obtient alors une grammaire

$G_1 = \langle V_T, V_N, S, R_1 \rangle$ sans ε -règle telle que $L(G_1) = L(G) - \{\varepsilon\}$.

On élimine ensuite les non-terminaux improductifs puis les non-terminaux inaccessibles. Ces éliminations ne faisant qu'enlever des règles, la grammaire obtenue $G_2 = \langle V_T, V_N, S, R_2 \rangle$ est réduite, équivalente à G_1 et sans ε -règle.

Par suite en posant $R' = R_2$, $G' = G_2$, la grammaire $G' = \langle V_T, V_N, S, R' \rangle$ est réduite et vérifie $L(G') = L(G) - \{\varepsilon\}$. ■

4 Élimination des 1-règles

Rappelons qu'une 1-règle est une règle de la forme $A \rightarrow B$ où B est un non-terminal. L'élimination n'a plus qu'un intérêt historique et doit être vu comme un exercice sur les transformations de grammaire.

Théorème 9.4.1 *Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte. On peut construire une grammaire hors-contexte équivalente sans 1-règle $G' = \langle V_T, V_N, S, R' \rangle$.*

PREUVE. On montre comment construire G' , ensuite on prouve $L(G') \subset L(G)$ puis $L(G) \subset L(G')$.

1. Construction de G' .

Soit r la relation sur V_N définie par : ArB si et seulement si $A \rightarrow B$ est une règle de G .

On a donné au paragraphe 2.2, l'algorithme de Roy-Warshall qui permet de calculer facilement r^* la fermeture transitive et reflexive de r .

On pose : $R' = \{A \rightarrow \alpha \mid \text{il existe } B \rightarrow \alpha \in R \text{ et } \alpha \notin V_N \text{ et } Ar^*B\}$.

Puisque r^* est calculable, il en est de même de R' .

2. $L(G') \subset L(G)$.

On donne uniquement l'argument essentiel de la preuve, toute règle de R' peut être remplacée par une dérivation dans G , c'est-à-dire si $A \rightarrow \alpha \in R'$ alors $A \Rightarrow_G^* \alpha$.

3. $L(G) \subset L(G')$.

On prouve par induction sur i que :

si A est un non-terminal et x une chaîne terminale telle que $A \Rightarrow_G^i x$ alors $A \Rightarrow_{G'}^* x$.

Par hypothèse d'induction, supposons la propriété vraie pour toute dérivation de longueur inférieure à i .

Supposons $A \Rightarrow_G^i x$. Il y a une dérivation de A en x , qui s'écrit $\alpha_0, \dots, \alpha_i$ où $\alpha_0 = A$ et $\alpha_i = x$.

Puisque A est un non-terminal et que x est une chaîne terminale, il y a un indice j entre 0 et $i-1$ tel que $\alpha_0, \dots, \alpha_j$ sont des non-terminaux et que la chaîne α_{j+1} n'est pas un non-terminal.

Par définition de R' , $A \rightarrow \alpha_{j+1}$ est une règle de R' .

Puisque $\alpha_0, \dots, \alpha_i$ est une dérivation de la grammaire G , on a $\alpha_{j+1} \Rightarrow_G^{i-(j+1)} x$.

Puisque $i - (j + 1) < i$, en appliquant la propriété fondamentale des grammaires hors-contexte et l'hypothèse d'induction, on obtient : $\alpha_{j+1} \Rightarrow_{G'}^* x$.

Par suite $A \Rightarrow_{G'}^* x$.

Donc la propriété encadrée ci-dessus est vraie pour tout i , d'où $L(G) \subset L(G')$. ■

Exemple 9.4.2 Soit la grammaire de terminaux $+, *, (,)$, de non-terminaux E, T, F , d'axiome E et de règles

$$E \rightarrow E + T | T$$

$$T \rightarrow T * F | F$$

$$F \rightarrow (E) | a$$

On a : $Er^*E, Er^*T, Er^*F, Tr^*T, Tr^*F, Fr^*F$.

Par suite les règles de la grammaire équivalente sans 1-règle sont

$$E \rightarrow E + T | T * F | (E) | a$$

$$T \rightarrow T * F | (E) | a$$

$$F \rightarrow (E) | a$$

On constate que cette grammaire a bien plus de règles.

Appelons expression un mot engendré par E , terme un mot engendré par T et facteur un mot engendré par F .

La nouvelle grammaire ne met plus en évidence ni que tout terme est une expression, ni que tout facteur est un terme. ♣

Théorème 9.4.3 Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte. Si $L(G)$ n'est pas vide, on peut construire une grammaire hors-contexte $G' = \langle V_T, V_N, S, R' \rangle$ résuite, sans ε -règle et sans 1-règle telle que $L(G') = L(G) - \{\varepsilon\}$.

Une grammaire ayant ces propriétés est appelée une grammaire propre.

PREUVE. Il suffit d'appliquer dans l'ordre les transformations suivantes : élimination des ε -règles, élimination des 1-règles, élimination des improductifs, élimination des inaccessibles.

En effet puisque l'élimination des 1-règles n'ajoute pas de nouvelle partie droite de règle, la grammaire construite par la deuxième élimination n'a pas ε -règles. Il en est de même des deux dernières éliminations : la grammaire finale G' est donc propre.

Puisque ces différentes éliminations change une grammaire en une autre grammaire équivalente à ε près, on a bien $L(G') = L(G) - \{\varepsilon\}$. ■

5 Forme normale de Chomsky

La forme normale de Chomsky n'a plus qu'un intérêt historique. Aussi comme le paragraphe précédent, il faut voir ce paragraphe comme l'exposé de méthodes pour prouver l'équivalence de grammaires.

Définition 9.5.1 (Forme Normale de Chomsky) Une grammaire hors-contexte est sous forme normale de Chomsky si toutes ses règles sont de l'une des deux formes suivantes :

- $A \rightarrow BC$ où A, B, C sont des non-terminaux
- $A \rightarrow a$ où a est un terminal

◇

Théorème 9.5.2 Soit G une grammaire hors-contexte. On peut construire une grammaire G' sous forme normale de Chomsky telle que $L(G') = L(G) - \{\varepsilon\}$.

PREUVE. D'après le théorème 9.4.3, pour toute grammaire hors-contexte, on peut construire une grammaire propre équivalente à ε près. On peut donc supposer que la grammaire G est sans ε -règle et sans 1-règle.

Posons $G = \langle V_T, V_N, S, R \rangle$ et soit $V = V_T \cup V_N$.

La grammaire $G' = \langle V_T, V'_N, [S], R' \rangle$ est construite ainsi

- Les symboles de V'_N sont obtenus en mettant entre crochets les éléments de V et les suffixes de longueur au moins deux des parties droites de règles de R , autrement dit :

$$V'_N = \{[B] \mid B \in V\} \cup \{[\alpha] \mid \text{il existe } \beta \text{ tel que } |\alpha| \geq 2, A \rightarrow \beta\alpha \in R\}$$

- Les règles de R' sont

$$\begin{aligned} R' = & \{[A] \rightarrow [B_1][B_2 \dots B_p] \mid A \rightarrow B_1 B_2 \dots B_p \in R, B_1, B_2, \dots, B_p \in V, p \geq 2\} \\ & \cup \{[B_1 B_2 \dots B_p] \rightarrow [B_1][B_2 \dots B_p] \mid [B_1 B_2 \dots B_p] \in V'_N, p \geq 2\} \\ & \cup \{[a] \rightarrow a \mid a \in V_T\} \\ & \cup \{[A] \rightarrow a \mid A \rightarrow a \in R \text{ avec } a \in V_T\} \end{aligned}$$

La grammaire G' est sous forme normale de Chomsky. Montrons qu'elle est équivalente à G .

1. Montrons que $L(G) \subset L(G')$.

Il suffit de prouver, par induction sur i , que :

$$\boxed{\text{si } A \text{ est un non-terminal et } x \text{ une chaîne terminale tel que } A \Rightarrow_G^i x \text{ alors } [A] \Rightarrow_{G'}^* x.}$$

Pour $i = 0$, c'est trivial.

Pour $i > 0$, supposons la propriété vraie pour toute dérivation de longueur inférieure à i .

Supposons $A \Rightarrow_G^i x$. Puisque $i > 0$, cette dérivation s'écrit :

$$A \Rightarrow_G B_1 \dots B_p \Rightarrow_G^{i-1} x \text{ où } B_1, \dots, B_p \in V.$$

D'après la propriété fondamentale des langages hors-contexte, il existe pour j de 1 à p , des entiers n_j et des chaînes x_j tel que :

$$(*) B_j \Rightarrow_G^{n_j} x_j \text{ et } x = x_1 \dots x_p \text{ et } i - 1 = n_1 + \dots + n_p.$$

On considère deux cas suivant la valeur de p .

- (a) $p = 1$.

Vu que G ne comporte pas de 1-règle, B_1 est un terminal égal à x . Donc $[A] \rightarrow x$ est une règle de G' donc $[A]$ engendre x (en un pas) dans la grammaire G' .

- (b) $p > 1$.

Par définition de R' , il est clair que $[A] \Rightarrow_{G'}^* [B_1][B_2] \dots [B_p]$, d'ailleurs en exactement $p - 1$ pas.

On distingue deux cas suivant que B_i (où $1 \leq i \leq p$) est un terminal ou un non-terminal.

- i. si B_i est un terminal alors $B_i = x_i$ et, par définition des nouvelles règles, $[B_i] \rightarrow x_i \in R'$, donc $[B_i] \Rightarrow_{G'}^* x_i$.

ii. si B_i est un non-terminal, puisqu'il engendre x_i en moins de i pas, par hypothèse d'induction, on a aussi $[B_i] \Rightarrow_{G'}^* x_i$.

Par composition de ces dérivations, $[B_1][B_2] \dots [B_p] \Rightarrow_{G'}^* x_1 x_2 \dots x_n$.

Puisque $x = x_1 \dots x_p$ et que $[A] \Rightarrow_{G'}^* [B_1][B_2] \dots [B_p]$, il en résulte que $[A] \Rightarrow_{G'}^* x$.

2. Montrons que $L(G') \subset L(G)$.

Nous présentons une preuve informelle comportant seulement l'argument essentiel.

Supposons que $[S]$ engendre la chaîne terminale x dans G' . Il y a une dérivation de G' , de x à partir de $[S]$. En effaçant les crochets de cette dérivation, on obtient une dérivation (avec des répétitions) de x (qui n'a pas de crochets) à partir de S dans la grammaire G .

Soyons plus formel. Soit $h : V_{N'} \mapsto V^*$ l'application qui efface les crochets, autrement dit :

Pour tout $\alpha \in V^*$ tel que $[\alpha] \in V_{N'}$ alors $h([\alpha]) = \alpha$.

Soit $V' = V_T \cup V_{N'}$. L'application h est étendue en un homomorphisme pour la concaténation de V'^* dans V^* en posant

— pour tout $a \in V_T$, $h(a) = a$

— pour tous $x, y \in V'^*$, $h(xy) = h(x)h(y)$

En examinant tous les règles de R' , on voit que si $[\alpha] \rightarrow \beta$ est une règle de R' alors soit $h([\alpha]) = h(\beta)$, soit $h([\alpha]) \rightarrow \beta$ est une règle de R . On en déduit facilement que $L(G') \subset L(G)$. ■

Exemple 9.5.3 On montre la transformation de grammaire ci-dessus. Soit la grammaire de terminaux a, b , de non-terminaux S, A, B , d'axiome S et de règles

$S \rightarrow aAB|BBBA$

$A \rightarrow BAB|a$

$B \rightarrow AS|b$

La nouvelle grammaire équivalente a comme non-terminaux $[a], [b], [S], [A], [B], [AB], [BBA], [BA]$, comme axiome $[S]$ et comme règles

$[S] \rightarrow [a][AB] \mid [B][BBA]$

$[a] \rightarrow a$

$[AB] \rightarrow [A][B]$

$[BBA] \rightarrow [B][BA]$

$[BA] \rightarrow [B][A]$

$[A] \rightarrow [B][AB] \mid a$

$[B] \rightarrow [A][S] \mid b$

Remarquez que le non-terminal $[b]$ est inutilisé. ♣

6 Grammaires hors-contexte récursives

Définition 9.6.1 Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte et $V = V_T \cup V_N$.

Un non-terminal A est

— *récursif* si et seulement si il existe $\alpha, \beta \in V^*$ tel que $A \Rightarrow^+ \alpha A \beta$.

- *récuratif à gauche* si et seulement si il existe $\alpha \in V^*$ tel que $A \Rightarrow^+ A\alpha$.
- *récuratif à droite* si et seulement si il existe $\alpha \in V^*$ tel que $A \Rightarrow^+ \alpha A$. ◇

Une grammaire hors-contexte est *récursive* (respectivement *récursive à gauche*, *récursive à droite* si elle contient un non-terminal récuratif (respectivement récuratif à gauche, récuratif à droite).

Propriété 9.6.2 *Le problème « la grammaire G est-elle récursive ? » est décidable.*

PREUVE. Il suffit de montrer que le problème «le non-terminal A de la grammaire G est-il récuratif?» est décidable.

Soit d la relation de dépendance sur V_N définie au paragraphe 2.2.

Il est clair que A est récuratif si et seulement si Ad^+A où d est la fermeture transitive de cette relation de dépendance.

Dans le paragraphe 2.2, on a vu comment calculer d^* , la fermeture transitive et réflexive de d . Le calcul de d^+ , la fermeture transitive de d , est analogue, il suffit de modifier la définition de d_0 par :

$A_i d_0 A_j$ si et seulement si $A_i d A_j$. ■

Les deux paragraphes suivants, on montre que pour toute grammaire hors-contexte, on peut construire une grammaire sans symbole récuratif à gauche. Cette transformation est appelée *l'élimination de la récursivité à gauche*. Il a plusieurs façons d'effectuer cette élimination. Nous avons choisi une méthode élémentaire (mais qui n'est pas la meilleure), effectuer une suite d'éliminations de règles *directement récuratives à gauche*, c'est-à-dire de règles de la forme $A \rightarrow A\alpha$. L'élimination de la récursivité gauche (respectivement droite) est encore actuellement une transformation importante, car les analyseurs descendants (respectivement ascendants) exigent des grammaires non-récuratives à gauche (respectivement non-récuratives à droites).

7 Élimination des règles directement récurative à gauche

Définition 9.7.1 Un non-terminal A est directement récuratif à gauche dans une grammaire si la grammaire comporte une règle de la forme $A \rightarrow A\alpha$. ◇

Théorème 9.7.2 *Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte et $V = V_T \cup V_N$.*

Soit A un non-terminal directement récuratif à gauche et soit A' un nouveau non-terminal, autrement dit $A' \notin V$.

Soit $G' = \langle V_T, V_N \cup \{A'\}, S, R' \rangle$ la grammaire où l'ensemble R' est obtenu en remplaçant les A -règles (c'est-à-dire les règles d'entête A) par

$$\begin{aligned} & \{A \rightarrow \beta \mid A \rightarrow \beta \in R \text{ et } \beta \text{ ne commence pas par } A\} \\ & \cup \{A \rightarrow \beta A' \mid A \rightarrow \beta \in R \text{ et } \beta \text{ ne commence pas par } A\} \\ & \cup \{A' \rightarrow \alpha \mid A \rightarrow A\alpha \in R\} \\ & \cup \{A' \rightarrow \alpha A' \mid A \rightarrow A\alpha \in R\}. \end{aligned}$$

On a les résultats suivants :

- Les deux grammaires sont équivalentes.
- A n'est pas directement récursif à gauche dans G' .
- Si G ne comporte pas la règle $A \rightarrow A$, alors A' n'est pas directement récursif à gauche dans G' .

PREUVE. Il est évident, par construction de G' , que, dans la nouvelle grammaire, A n'est pas récursif à gauche et A' non plus, à condition que $A \rightarrow A$ ne soit pas une règle de G .

On montre que $L(G) \subset L(G')$, en donnant uniquement l'argument essentiel, et on laisse au lecteur la preuve (analogue) de l'inclusion inverse.

Dans le début d'une dérivation gauche à partir de A d'une chaîne terminale il y a un premier pas qui ne commence pas par A , donc il existe $p \geq 0$ tel que le début de cette dérivation s'écrit :

Pour i de 1 à p , $A\alpha_{i-1} \dots \alpha_1 \Rightarrow A\alpha_i\alpha_{i-1} \dots \alpha_1$ (par la règle $A \rightarrow A\alpha_i$ de G)

$A\alpha_p\alpha_{p-1} \dots \alpha_1 \Rightarrow \beta\alpha_p \dots \alpha_1$ par la règle $A \rightarrow \beta$ où β ne commence pas par A .

Vu le remplacement des récursions directes gauches par des récursions directes droites, on a aussi dans G' :

$A \Rightarrow_{G'}^* \beta\alpha_p \dots \alpha_1$. En effet :

1. Si $p = 0$, alors $A \rightarrow \beta$, est une règle commune à G et G' .

2. Si $p > 0$, alors dans G' il y a la dérivation droite :

$A \Rightarrow \beta A'$

Pour i de p à 2, $\beta\alpha_p \dots \alpha_{i+1} A' \Rightarrow \beta\alpha_p \dots \alpha_{i+1} \alpha_i A'$ (par la règle $A' \rightarrow \alpha_i A'$ de R')

$\beta\alpha_p \dots \alpha_2 A' \Rightarrow \beta\alpha_p \dots \alpha_2 \alpha_1$ (par la règle $A' \rightarrow \alpha_1$ de R')

On en déduit que si le non-terminal A engendre une chaîne terminale x par dérivation gauche dans G , alors A engendre x dans G' .

D'après la propriété 7.2.14 du chapitre 7, si le non-terminal A engendre une chaîne terminale x dans G , alors il engendre aussi x par dérivation gauche.

Par suite toute chaîne terminale engendrée par un non-terminal de G est aussi engendrée par ce même non-terminal dans G' . Donc $L(G) \subset L(G')$.

On prouve de façon analogue, en remplaçant les dérivations gauches par des dérivations droites, que $L(G') \subset L(G)$. ■

Exemple 9.7.3 Soit la grammaire de terminaux a, b , de non-terminal S et de règles :

$S \rightarrow SS|aSb|\epsilon$.

En éliminant la récursion gauche directe, on obtient une grammaire d'axiome S et de règles

$S \rightarrow aSbS'|S'|aSb|\epsilon$

$S' \rightarrow SS'|S$

La récursion directe gauche a été éliminée, mais les non-terminaux S et S' sont récursifs à gauche. ♣

8 Élimination de la récursivité à gauche

On élimine la récursion à gauche, car les analyseurs descendants (ceux qui analysent les mots en essayant de construire une dérivation à partir de l'axiome) exigent des grammaires non récur-

sives à gauches.

Nous donnons ci-dessous deux exemples modèles où l'élimination de la récursion gauche est effectuée simplement en la remplaçant par une récursion droite.

Exemple 9.8.1 Rappelons la grammaire des expressions arithmiques.

$$E \rightarrow E + T | T$$

$$T \rightarrow T * F | F$$

$$F \rightarrow (E) | a$$

On peut éliminer la récursion à gauche simplement en la remplaçant par

$$E \rightarrow T + E | T$$

$$T \rightarrow F * T | F$$

$$F \rightarrow (E) | a$$

La nouvelle grammaire, sans récursion gauche, est bien plus courte et plus descriptive que celle résultant des méthodes ci-dessous. ♣

Exemple 9.8.2 La grammaire suivante, décrivant des listes de a

$$L \rightarrow La | \varepsilon$$

est équivalente à la grammaire sans récursion gauche

$$L \rightarrow aL | \varepsilon$$

Cet exemple montre que, contrairement aux méthodes ci-dessous, l'élimination des ε -règles n'est pas un préalable indispensable à l'élimination de la récursion à gauche. ♣

Nous proposons ci-dessous un algorithme d'élimination de la récursion gauche, qui opère sur les grammaires sans ε -règle et sans 1-règle. Il existe d'autres algorithmes¹, qui n'obligent pas à respecter ces deux conditions et introduisent moins de non-terminaux en faisant l'analogie de l'élimination directe à gauche simultanément sur plusieurs non-terminaux : l'intérêt de ces algorithmes plus complexes est notamment de mieux respecter les qualités «descriptives» des grammaires, comme sur les exemples ci-dessus.

Algorithme 8.1 Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte sans ε -règle et sans 1-règle. Soit $V = V_T \cup V_N$ et A_1, \dots, A_n une liste sans répétition des non-terminaux de G .

On construit une suite $G_{(i|1 \leq i \leq n+1)}$ de grammaires ainsi définies.

$$G_1 = G$$

Soit $G_i = \langle V_T, V_{N_i}, S, R_i \rangle$. On montre comment, pour $i \leq n$, on obtient G_{i+1} à partir de G_i .

La construction comporte deux étapes

1. Pour j de 1 à i on construit la suite de grammaires $G_{i,j} = \langle V_T, V_{N_i}, S, R_{i,j} \rangle$.

$$G_{i,1} = G_i.$$

Pour $j < i$, la grammaire $G_{i,j+1}$ est obtenu en remplaçant, chaque règle de $G_{i,j}$ de la forme $A_i \rightarrow A_j \alpha$ par l'ensemble de règles $\{A_i \rightarrow \beta \alpha \mid A_j \rightarrow \beta \text{ est une règle de } G_{i,j}\}$.

1. Voir par exemple la thèse de Jérôme Bordier
Méthode pour la mise au point de grammaires LL(1)
Université Joseph-Fourier - Grenoble I (1971-01-30), Jean-Claude Boussard (Dir.)
<http://tel.archives-ouvertes.fr/docs/00/04/81/34/PDF/tel-00009479.pdf>

2. Si A_i n'est pas directement récursif à gauche dans $G_{i,i}$ alors $G_{i+1} = G_{i,i}$.
3. Si A_i est directement récursif à gauche dans $G_{i,i}$ alors G_{i+1} est construite ainsi :
 Soit A'_i un nouveau non-terminal : $V_{N_{i+1}} = V_{N_i} \cup \{A'_i\}$.
 On remplace les A_i -règles par l'ensemble des règles
 $\{A_i \rightarrow \beta \mid A_i \rightarrow \beta \in R_{i,i} \text{ et } \beta \text{ ne commence pas par } A_i\}$
 $\cup \{A_i \rightarrow \beta A'_i \mid A_i \rightarrow \beta \in R_{i,i} \text{ et } \beta \text{ ne commence pas par } A_i\}$
 $\cup \{A'_i \rightarrow \alpha \mid A_i \rightarrow A_i \alpha \in R_{i,i}\}$
 $\cup \{A'_i \rightarrow \alpha A'_i \mid A_i \rightarrow A_i \alpha \in R_{i,i}\}$.

Exemple 9.8.3 Soit la grammaire G_1 de terminaux a, b, c, d, e , de non-terminaux A_1, A_2 et de règles

$$A_1 \rightarrow A_2 a | b$$

$$A_2 \rightarrow A_2 c | A_1 d | e$$

$G_2 = G_1$, car A_1 n'est pas directement récursif.

$G_{2,2}$ a comme règle :

$$A_1 \rightarrow A_2 a | b$$

$$A_2 \rightarrow A_2 c | A_2 a d | b d | e \quad \text{En remplaçant } A_1 \text{ par sa définition}$$

La grammaire G_3 ci-dessous est obtenue par élimination de la récursion gauche directe sur A_2 :

$$A_1 \rightarrow A_2 a | b$$

$$A_2 \rightarrow b d | e | b d A'_2 | e A'_2$$

$$A'_2 \rightarrow c A'_2 | a d A'_2 | c | a d$$



Théorème 9.8.4 Soit G une grammaire sans ε -règle et sans 1-règle. On peut construire une grammaire équivalente à G non récursive à gauche.

PREUVE. On adopte les notations de l'algorithme 8.1.

Il est évident que toutes les grammaires construites sont équivalentes. En effet les transformations de grammaires utilisées sont des substitutions générales et des éliminations de la récursion gauche directe, et les théorèmes 9.1.4 et 9.7.2 montrent que ces transformations changent une grammaire en une autre équivalente.

Nous montrons, par induction, que la grammaire G_{n+1} n'est pas récursive à gauche.

Soit $P(i, j)$ la propriété suivante de la grammaire $G(i, j)$:

- Toutes les A_k -règles où $1 \leq k \leq n$ ont des parties droites commençant par un terminal, ou de la forme $A_l \alpha$ où α a un premier symbole élément de V .
- Toutes les A_k -règles où $1 \leq k < i$ ont des parties droites commençant par un terminal ou de la forme $A_l \alpha$ où $k < l$.
- Toutes les règles de la forme $A_i \rightarrow A_k \alpha$ où sont telles que $k \geq j$.
- La partie droite de toutes les règles d'entête un nouveau non-terminal, commencent par un élément de V .

Nous montrons que pour tout i, j , $P(i, j)$ est vraie.

1. Si $1 \leq i \leq n$ et $j < i$ et si $P(i, j)$ est vraie alors $P(i, j+1)$ est vraie.

Supposons que $1 \leq i \leq n$, $j < i$ et que la propriété $P(i, j)$ est vérifiée.

Puisque la grammaire $G(i, j+1)$, est obtenue en remplaçant toutes règles de la forme

$A_i \rightarrow A_j \alpha$ par

$\{A_i \rightarrow \beta \alpha \mid A_j \rightarrow \beta \in R_{i,j}\}$, et que, d'après la propriété $P(i, j)$, toutes les chaînes β commencent par un terminal ou s'écrivent $A_k \gamma$ où $k > j$ et γ commence par un symbole de V , alors tous les nouvelles règles de $G(i, j + 1)$ sont de la forme $A_i \rightarrow \delta$ où δ commence par un terminal ou est de la forme $A_k \eta$ où $k \geq j + 1$ et η commence par un symbole de V .

Par suite la propriété $P(i, j + 1)$ est vérifiée.

2. Puisque $G_{1,1} = G_1 = G$, et que G est sans ε -règle et sans 1-règle, alors la propriété $P(1, 1)$ est vérifiée.
3. Pour $1 \leq i \leq n$ alors $P(i, 1)$ implique $P(i + 1, 1)$.

Supposons que $1 \leq i \leq n$ et que la propriété $P(i, 1)$ vérifiée.

Puisque $P(i, 1)$ est vraie et que pour $j < i$, $P(i, j)$ implique $P(i, j + 1)$, alors par induction sur j , $P(i, i)$ est vraie.

On examine deux cas suivant que le non-terminal A_i est ou n'est pas directement récursif à gauche dans $G_{i,i}$.

- (a) Supposons que A_i n'est pas directement récursif à gauche.

Par définition $G_{i+1,1} = G_{i+1} = G_{i,i}$.

Puisque la propriété $P(i, i)$ est vérifiée et, vu l'absence de récursion directe gauche sur A_i , toutes les A_i -règles où $1 \leq k < i$ ont des parties droites commençant par un terminal ou de la forme $A_k \alpha$ où $i < k$. Donc la propriété $P(i + 1, 1)$ est vérifiée.

- (b) Supposons que A_i est directement récursif à gauche.

La grammaire $G_{i+1,1} = G_{i+1}$ est obtenu par élimination de la récursion directe gauche, autrement dit les A_i règles de $R_{i,i}$ sont remplacées par l'ensemble de règles

$\{A_i \rightarrow \beta \mid A_i \rightarrow \beta \in R_{i,i} \text{ et } \beta \text{ ne commence pas par } A_i\}$

$\cup \{A_i \rightarrow \beta A'_i \mid A_i \rightarrow \beta \in R_{i,i} \text{ et } \beta \text{ ne commence pas par } A_i\}$

$\cup \{A'_i \rightarrow \alpha \mid A_i \rightarrow A_i \alpha \in R_{i,i}\}$

$\cup \{A'_i \rightarrow \alpha A'_i \mid A_i \rightarrow A_i \alpha \in R_{i,i}\}$.

D'après la propriété $P(i, i)$, les chaînes α commencent par un élément de V , donc toutes A'_i règles commencent par un élément de V .

D'après la propriété $P(i, i)$, les chaînes β commencent par un terminal ou s'écrivent $A_j \gamma$ où $j > i$ et γ commence par un symbole de V .

Donc tous les nouvelles règles de $G_{i+1} = G(i + 1, 1)$ sont de la forme $A_i \rightarrow \delta$ où δ commence par un terminal ou sont de la forme $A_k \eta$ où $k \geq j + 1$ et η commence par un symbole de V .

Par suite $P(i + 1, 1)$ est aussi vérifiée.

4. La grammaire G_{n+1} n'est pas récursive à gauche.

On a vu ci-dessus que $P(1, 1)$ est vraie et que pour tout $1 \leq i \leq n$, $P(i, 1)$ implique $P(i + 1, 1)$. Donc par induction la propriété $P(n + 1, 1)$ est vérifiée.

Par suite la grammaire G_{n+1} , où $G_{n+1} = G_{n+1,1}$ vérifie :

- (a) Toute partie droite d'un A_i -règle (où $1 \leq i \leq n$) commence par un terminal ou par A_j où $i < j$.
- (b) Tout nouveau non-terminal a une partie droite commençant par un élément de V

De ces deux propriétés, il résulte que G_n n'est pas récursive à gauche. En effet

— Supposons que $A_i \Rightarrow^+ \alpha$.

Par induction sur la longueur de cette dérivation, α commence que par un terminal ou un non-terminal A_j où $j > i$. Donc les non-terminaux A_i ne sont pas récursifs à gauche.

— Supposons que $A'_i \Rightarrow^+ \alpha$.

Puisque les parties droite des A'_i -règles commencent par un élément de B , il existe un élément $B \in V$ et des chaînes β, γ, δ telles que :

* $A'_i \Rightarrow B\beta \Rightarrow^* \gamma\delta$

* $B \Rightarrow^* \gamma$ et $\beta \Rightarrow^* \delta$ et $\alpha = \gamma\delta$

Puisque B engendre ou est égal à γ , alors d'après ce qui vient d'être démontré, γ , donc aussi α , commence par un symbole de V . Donc les non-terminaux A'_i ne sont pas récursifs à gauche.

Par suite la grammaire G_{n+1} n'est pas récursive à gauche. ■

Notons une conséquence immédiate du théorème. Toute grammaire est équivalente au mot vide près à une grammaire non récursive à gauche. Il suffit d'éliminer dans l'ordre les ε -règles, les 1-règles, puis d'appliquer l'algorithme 8.1 ci-dessus.

9 Forme normale de Greibach

Définition 9.9.1 Une grammaire est sous la forme normale de Greibach si toutes ses règles sont de la forme

$A \rightarrow aB_1 \dots B_p$ où $p \geq 0$, a est un terminal et les B_i sont des non-terminaux. ◇

Algorithme 9.1 Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte sans ε -règle, et non récursive à gauche. L'algorithme transforme G en une grammaire équivalente dont toutes les parties droites de règles commencent par un terminal.

Considérons la relation g suivante entre les symboles de V_N :

$A g B$ si et seulement si il existe une chaîne α telle que $A \Rightarrow^+ B\alpha$.

Cette relation est un ordre partiel : en effet, elle est évidemment transitive, et, puisque la grammaire n'est pas récursive à gauche, elle est irreflexive.

Par suite il existe un ordre total², c'est-à-dire une liste sans répétition des symboles de V_N , A_1, \dots, A_n prolongeant l'ordre partiel c'est à dire tel que si $A_i g A_j$ alors $i < j$.

Si $A_i \rightarrow A_j\alpha$ est une règle de G , alors $A_i g A_j$ donc $j > i$. Puisque G ne comporte pas d' ε -règles, les autres règles de G commencent par un terminal.

Une fois numérotés les non-terminaux dans l'ordre indiqué, on construit par des substitutions générales une grammaire dont toutes les règles commencent par un terminal.

Pour i de n à 1, on construit la suite de grammaires $G_i = \langle V_T, V_N, S, R_i \rangle$ de la façon suivante :

— $G_n = G$

2. Cet ordre est calculable par l'algorithme du tri topologique

- pour $n \geq i \geq 2$ la grammaire G_{i-1} est obtenue en remplaçant chaque règle de G_i de la forme $A_{i-1} \rightarrow A_j \alpha$ par $\{A_{i-1} \rightarrow \beta \alpha \mid A_j \rightarrow \beta \in R_i\}$.

Théorème 9.9.2 Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte sans ε -règle, sans 1-règle, non réursive à gauche. Soit $V = V_T \cup V_N$. On peut construire une grammaire équivalente dont toutes les parties droites de règle commencent par un terminal.

PREUVE. On adopte les notations de l'algorithme 9.1.

Il est évident que toutes les grammaires construites sont équivalentes. En effet les transformations de grammaires utilisées sont des substitutions générales et le théorème 9.1.4 montre que ces transformations changent une grammaire en une autre équivalente.

Nous montrons que toutes les parties droites des règles de la grammaire G_1 commencent par un terminal.

Soit $P(i)$ la propriété suivante de la grammaire G_i :

- (a) pour tout $j < i$, toutes les parties droites de A_j -règles commencent par un terminal ou sont de la forme $A_k \alpha$ où $k < j$.
- (b) pour $i \leq j$, toutes les parties droites des A_j -règles commencent par un terminal.

1. $P(n)$ est vérifiée.

D'après l'ordre adopté des non-terminaux, toutes les règles de G commencent par un terminal ou sont de la forme $A_i \rightarrow A_j \alpha$ où $j > i$.

Comme il y a n non-terminaux, les A_n règles commencent par un terminal.

Puisque $G_n = G$, $P(n)$ est vérifiée.

2. Pour $i \geq 2$, $P(i)$ implique $P(i-1)$.

Supposons $i \geq 2$ et la propriété $P(i)$ vérifiée.

Toute A_{i-1} -règle de G_i qui ne commence pas par un terminal, commence par un non-terminal A_j où $i \leq j$. Par construction, A_j est remplacé par les parties droites des A_j -règles et d'après $P(i)$, ces parties droites commencent par un terminal.

Donc après ces remplacements, la propriété $P(i-1)$ est vérifiée.

3. Par induction puisque $P(n)$ est vérifiée et puisque, pour tout $i \geq 2$, $P(i)$ implique $P(i-1)$, il résulte que $P(1)$ est vérifiée. Donc tous les parties droites de règles commencent par un terminal. ■

Théorème 9.9.3 Pour toute grammaire hors-contexte, on peut construire une grammaire équivalente à ε près sous forme normale de Greibach.

PREUVE. Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte et $V = V_T \cup V_N$.

Soit A_1, \dots, A_n une liste sans répétitions des non-terminaux de cette grammaire.

La construction de la grammaire sous forme normale de Greibach, est faite en plusieurs étapes dans l'ordre ci-dessous :

1. Élimination des ε -règles décrite au paragraphe 3.

2. Élimination des 1-règles décrite au paragraphe 4.
Puisque l'élimination des 1-règles n'introduit pas d' ϵ -règles, la grammaire obtenue après cette étape, est sans ϵ -règle et sans 1-règle.
3. Élimination de la récursion à gauche par l'algorithme 8.1.
Comme résultat de cet algorithme, on obtient une grammaire $G' = \langle V_T, V_{N'}, S, R' \rangle$ avec $V_{N'} = V_N \cup E$ ou E est un ensemble de nouveaux non-terminaux et les règles de R' ayant les propriétés suivantes :
 - (a) La partie droite des règles ayant pour entête un nouveau non-terminal commence par un élément de V
 - (b) Toute partie droite d'une A_i règles (où $1 \leq i \leq n$) commencent par un terminal ou par un non-terminal A_j où $j > i$.

Soit B_1, \dots, B_p une liste sans répétition des nouveaux non-terminaux. Il en résulte que l'ordre total utilisé dans l'algorithme 9.1 n'a pas besoin d'être calculé, il peut être défini par la liste $B_1, \dots, B_p, A_1, \dots, A_n$ dans laquelle tout non terminal A précède tout non-terminal B tel que $A \rightarrow B$ (c'est-à-dire tels que il existe une chaîne α telle que $A \Rightarrow^+ B\alpha$).
4. Transformation de la grammaire G' en une grammaire dont tous les parties droites de règle commencent par un terminal. Il suffit d'appliquer l'algorithme 9.1, avec l'ordre total des non-terminaux déjà défini ci-dessus.
5. Passage à la forme normale de Greibach. Il suffit de :
 - (a) Remplacer chaque règle de la forme $A \rightarrow a\alpha$ par $A \rightarrow a\alpha'$ où α' est obtenu en remplaçant chaque terminal b par un nouveau non-terminal b' .
Par exemple la règle $A \rightarrow aBbaC$ où a et b sont des terminaux est remplacée par la règle $A \rightarrow aBb'a'C$ où a' et b' sont des nouveaux non-terminaux (avec un nouveau non-terminal pour chaque terminal).
 - (b) Pour chaque terminal a , ajouter la règle $a' \rightarrow a$

Il est évident que cette transformation change une grammaire en une autre équivalente. ■

Chapitre 10

Conditions d'unicité des systèmes d'équation

Ce chapitre est un complément du chapitre qui utilise les systèmes d'équation pour donner des définitions inductives des langages hors-contextes. Nous nous intéressons aux conditions nécessaires et suffisantes *calculables* pour qu'un système d'équation ait une seule solution.

Ce chapitre est un hommage à Bruno Courcelle, voir <http://www.labri.fr/perso/courcell/>, qui est l'auteur de cette étude intéressante et trop souvent oubliée.

Dans le premier paragraphe, on définit un système strict et on montre qu'un tel système a une seule solution. Dans le deuxième paragraphe, on étudie des conditions nécessaires et suffisantes pour qu'un système ait une seule solution : un système riche¹ a une seule solution si et seulement si on peut le transformer par substitutions en un système strict et cette condition est calculable.

1 Système strict

Définition 10.1.1 (système strict) La définition d'une variable d'un système est stricte si tous ses mots sont vides ou comportent un terminal. Un système d'équations est strict si toutes les définitions de ses variables sont strictes. ◇

Exemple 10.1.2 Le système $S = aSb + \varepsilon$ est strict.

Le système $S = SS + aSb + \varepsilon$ n'est pas strict. ♣

Théorème 10.1.3 *Un système strict a une seule solution.*

PREUVE. La preuve n'est qu'une généralisation de l'exemple étudié au paragraphe 1 du chapitre 10.

Soit $S = \langle V_T, V_N, d \rangle$ un système strict d'équations et $V = V_T \cup V_n$.

Soit I la plus petite solution du système et J une solution quelconque. Supposons que ces deux

1. Un système riche est défini en 10.2.2

solutions soient distinctes.

Il existe x une chaîne terminale *de longueur minimale* et un non terminal A tel que $x \in J(A)$ et $x \notin I(A)$, autrement dit, pour toute chaîne terminale y de longueur inférieure à x et tout non terminal B , $x \in J(B)$ si et seulement si $x \in I(B)$.

Donc il existe α un mot de la définition de A tel que $x \in J(\alpha)$ et $x \notin I(\alpha)$.

Le mot α n'est pas une chaîne terminale, car dans ce cas, nous aurions $I(\alpha) = J(\alpha) = \{\alpha\}$. Donc $\alpha = B_1 B_2 \dots B_n$ où $B_1, B_2, \dots, B_n \in V$, $n \geq 1$ et, puisque le système est strict, au moins un de ces symboles est un terminal et un autre un non terminal.

Par définition des interprétations, il existe une suite x_1, x_2, \dots, x_n de chaînes terminales telles que, pour i de 1 à n , $x_i \in J(B_i)$ et $x = x_1 x_2 \dots x_n$.

Soit B_i est un terminal. Alors $J(B_i) = I(B_i) = \{x_i\}$

Soit B_i est un non terminal. Puisque que le système est strict, une des chaînes x_1, x_2, \dots, x_n est un symbole terminal, donc la longueur de x_i est inférieure (strictement) à celle de x . Et, vu que la chaîne x est la chaîne la plus courte pour laquelle existe un non terminal A avec $x \in J(A)$ et $x \notin I(A)$, on a $x_i \in I(B_i)$.

Par suite $x \in I(\alpha)$, donc $x \in I(A)$: on obtient une contradiction. Donc $I = J$ et le système n'a qu'une solution. ■

2 Conditions nécessaires et suffisantes d'unicité

Soit S un système d'équations, transformé par des substitutions générales² en un système strict S' . D'après la propriété 8.5.6 du chapitre 10, toute solution de S est une solution de S' . D'après le théorème 10.1.3, un système strict n'a qu'une solution. Donc S n'a qu'une solution.

Exemple 10.2.1 Le système d'équations S suivant n'est pas strict :

$$E = T + E \underline{+} T$$

$$T = F + T * F$$

$$F = a + b + (E)$$

Cependant on peut le transformer en un système strict S' par des substitutions générales :

$$E = a + b + (E) + T * F + E \underline{+} T$$

$$T = a + b + (E) + T * F$$

$$F = a + b + (E)$$

Donc S n'a qu'une solution. ♣

On peut se demander si le fait pour un système d'être transformable par des substitutions générales en un système strict est une condition nécessaire et suffisante d'unicité des solutions. Un exemple nous montre que ce n'est pas tout à fait le cas.

Soit le système $S = SS + a + \varepsilon$. Par des substitutions générales, il est impossible de transformer ce système en un système strict. Lorsque le vocabulaire terminal ne comprend que la lettre a ,

2. On rappelle qu'une telle substitution consiste à remplacer dans les définitions du système, des occurrences de variables par leur définition

l'unique solution de ce système est $S = a^*$. Par contre lorsque le vocabulaire terminal comprend les deux lettres a et b , le système a les deux solutions $S = a^*$ et $S = (a + b)^*$.

Définition 10.2.2 (Système riche) Un système d'équations est riche, si et seulement si, l'ensemble des mots de la plus petite solution du système est *strictement* inclus dans l'ensemble des chaînes sur le vocabulaire terminal du système. \diamond

Lorsque l'on considère le système $S = SS + a + \varepsilon$ comme ayant le vocabulaire terminal $\{a\}$, il n'est pas riche. Mais il est riche, si l'on considère que son vocabulaire terminal est $\{a, b\}$. On montre dans la suite qu'un système *riche* a une seule solution si et seulement si il est transformable par des substitutions générales en un système strict.

Définition 10.2.3 (Variables potentiellement strictes) Soit $S = \langle V_T, V_N, d \rangle$ un système d'équations et $V = V_T \cup V_N$. Soit E l'ensemble de variables engendrant la chaîne vide au moyen des règles du système. On a vu, au paragraphe 3 du chapitre 9, comment calculer cet ensemble.

Soit $f : \mathcal{P}(V_N) \mapsto \mathcal{P}(V_N)$ la fonction suivante :

pour tout $X \subset V_N$, $f(X) = \{A \in V_N \mid \forall \alpha \in d(A), \alpha \in X^* \cup V^* V_T V^* \cup V^* (X - E) V^*\}$.

La fonction f est une fonction croissante sur le treillis fini $\mathcal{P}(V_N)$ ordonné par l'inclusion. Puisqu'on sait calculer E , la fonction f est calculable. En suivant l'exemple du calcul de l'ensemble des symboles productifs donné au paragraphe 2.1 du chapitre 9, cette fonction a un plus petit point fixe F calculable ainsi :

Soit n le nombre de variables du système, $F = f^i(\emptyset)$ où i est le plus petit entier inférieur à n tel que $f^i(\emptyset) = f^{i+1}(\emptyset)$.

F est l'ensemble des variables potentiellement strictes du système : ce nom vient, comme on le voit ci-dessous, de ce que F comporte les variables strictes du système ainsi que les variables qui peuvent devenir strictes par substitution. \diamond

Lemme 10.2.4 Soit S un système d'équations et F l'ensemble des variables potentiellement strictes du système. Si F est l'ensemble des variables de S , alors S est transformable par substitutions en un système strict.

PREUVE. Soit $S = \langle V_T, V_N, d \rangle$ un système d'équations et $V = V_T \cup V_N$.

Soit F l'ensemble des variables potentiellement strictes du système.

Soit $f : \mathcal{P}(V_N) \mapsto \mathcal{P}(V_N)$ la fonction suivante :

pour tout $X \subset V_N$, $f(X) = \{A \in V_N \mid \forall \alpha \in d(A), \alpha \in X^* \cup V^* V_T V^* \cup V^* (X - E) V^*\}$.

On sait que F est le plus petit point fixe de f et qu'il existe un entier i tel que $F = f^i(\emptyset)$.

Pour tout entier j , posons $F_j = f^j(\emptyset)$.

Soit $S_0 = S$ et S_{i+1} le système ainsi obtenu : dans les définitions de S_i , on remplace toutes les occurrences de variables par leur définitions dans S_i .

Montrons que :

(*) pour tout entier j , l'ensemble F_j est inclus dans l'ensemble des variables strictes du système S_j .

Pour $j = 0$ c'est évident puisque F_0 est l'ensemble vide.

Supposons que F_j est inclus dans l'ensemble des variables strictes du système S_j et montrons

qu'il en est de même en remplaçant j par $j + 1$.

Soit $A \in F_{j+1}$. On examine deux cas.

1. Supposons $A \in F_j$. Par hypothèse d'induction, A est strict dans S_j . Puisque S_{j+1} est obtenu par substitution sur S_j , la variable A reste stricte dans S_{j+1} .
2. Supposons $A \notin F_j$. Par définition de F_{j+1} , toutes les règles $A \rightarrow \alpha$ sont telles que :
 $\alpha \in F_i^* \cup V^*V_TV^* \cup V^*(F_i - E)V^*$.
 Soit $A \rightarrow \beta$ une règle de S_{j+1} , obtenue en remplaçant dans α les variables par leurs définitions dans S_j . Montrons que β est vide ou comporte un terminal. On examine trois cas suivant l'ensemble dont α est élément.
 - (a) Soit $\alpha \in F_i^*$. Par hypothèse de récurrence toute variable de α est stricte dans S_i , donc est remplacée par la chaîne vide ou des chaînes comportant un terminal. Par suite β est vide ou comporte un terminal.
 - (b) Soit $\alpha \in V^*V_TV^*$. Alors β comporte un terminal, puisqu'il y en a déjà un dans α .
 - (c) Soit $\alpha \in V^*(F_i - E)V^*$. Donc α comporte une variable $X \in F_i - E$. Par récurrence cette variable est stricte dans S_j , donc les mots de sa définition sont vides ou comprennent un terminal. Puisque cette variable ne produit pas la chaîne vide, ses parties droites de règles comportent toutes un terminal, qui, en remplaçant X par sa définition dans S_j , se retrouve dans β .

Donc la variable A devient stricte dans S_{j+1} .

Supposons que F est l'ensemble des variables du système S .

On sait qu'il existe un entier i tel que $F = F_i$. D'après la propriété (*), pour tout j , F_j est inclus dans l'ensemble des variables strictes de S_j .

Puisque l'ensemble des variables ne change pas, toutes les variables de S_i sont strictes. Donc le système S_i , obtenu en transformant S par des substitutions, est strict. ■

Lemme 10.2.5 *Soit $S = \langle V_T, V_N, d \rangle$ un système d'équations et F l'ensemble des variables potentiellement strictes du système. Soit $\bar{F} = V_N - F$ l'ensemble complémentaire de F . Pour toute variable A , on a :*

1. $A \in F$ si et seulement si pour tout $\alpha \in d(A)$, on a : $\alpha \in F^* \cup V^*V_TV^* \cup V^*(F - E)V^*$.
2. $A \in \bar{F}$ si et seulement si il existe une chaîne $\alpha \in d(A)$, telle que : $\alpha \in (E \cup \bar{F})^* \bar{F} (E \cup \bar{F})^*$

PREUVE. Soit $f : \mathcal{P}(V_N) \mapsto \mathcal{P}(V_N)$ la fonction suivante :

pour tout $X \subset V_N$, $f(X) = \{A \in V_N \mid \forall \alpha \in d(A), \alpha \in X^* \cup V^*V_TV^* \cup V^*(X - E)V^*\}$.

On sait que F est le plus petit point fixe de f , donc $f(F) = F$, ce qui est équivalent à la première propriété.

La négation de la condition $\alpha \in F^* \cup V^*V_TV^* \cup V^*(F - E)V^*$ est la conjonction des 3 propriétés

- (a) $\alpha \notin F^*$, autrement dit un des symboles de α n'est pas dans F , c'est-à-dire $\alpha \in V^*\bar{F}V^*$.
- (b) $\alpha \notin V^*V_TV^*$, autrement dit tous les symboles de α sont des variables, c'est-à-dire $\alpha \in V_N^*$.
- (c) $\alpha \notin V^*(F - E)V^*$, autrement dit tous les symboles de α sont éléments de $\bar{F} - \bar{E}$ où

$$\bar{F} - \bar{E} = \overline{F \cap \bar{E}} = E \cup \bar{F}$$

Puisque tous les éléments de $E \cup \bar{F}$ sont des variables, la condition (c) implique la condition (b), et donc la conjonction des propriétés (a, b, c) est équivalente à la conjonction de la propriété (b) «un des symboles de α est élément de \bar{F} » et de la propriété (c), «tous les éléments de α sont éléments de $E \cup \bar{F}$ ».

Par suite la négation de la condition $\alpha \in F^* \cup V^* V_T V^* \cup V^* (F - E) V^*$ est $\alpha \in (E \cup \bar{F})^* \bar{F} (E \cup \bar{F})^*$. Donc la propriété 2 est obtenue en remplaçant dans la propriété 1 les deux membres de l'équivalence par leur négations : elle est donc équivalente à la propriété 1. ■

Lemme 10.2.6 *Soit S un système riche d'équations (voir définition 10.2.2). Soit F l'ensemble des variables potentiellement strictes du système. Si F n'est pas l'ensemble de toutes les variables du système, alors S a plusieurs solutions.*

PREUVE. Soit $S = \langle V_T, V_N, d \rangle$ et I la plus petite solution de S . Puisque S est riche, il y a une chaîne terminale w qui n'appartient pas à $\bigcup \{I(A) \mid A \in V_N\}$.

Soit $\bar{F} = V_N - F$, le complément de F et J l'interprétation suivante définie pour toute variable A par

- si $A \in \bar{F}$ alors $J(A) = \{\varepsilon, w\}$
 - si $A \in E \cap F$ alors $J(A) = \{\varepsilon\}$.
- On rappelle que E est l'ensemble des variables engendrant ε grâce aux règles du système.
- si $A \in F - E$ alors $J(A) = \emptyset$.

Soit f_S la fonction définie par le système S (voir la définition 8.3.6 du chapitre 10). On montre tout d'abord que $J \subset f_S(J)$, c'est-à-dire que pour tout $A \in V_N$, $J(A) \subset f_S(J)(A)$.

On examine trois cas suivant la définition de J :

1. Soit $A \in \bar{F}$. $J(A) = \{\varepsilon, w\}$.

D'après le cas 2 du lemme 10.2.5, il existe une chaîne $\alpha \in d(A)$ telle que $\alpha \in (E \cup \bar{F})^* \bar{F} (E \cup \bar{F})^*$.

Notons que tous les symboles de α sont des variables. Soit B une telle variable.

Puisque $B \in E \cup \bar{F}$ et que $E \cup \bar{F} = (E \cap F) \cup \bar{F}$, d'après les deux premiers cas de la définition de J , on a $\varepsilon \in J(B)$.

D'autre part il y a symbole C de α tel que $C \in \bar{F}$, donc par définition de J , $J(C) = \{\varepsilon, w\}$.

Par suite $\{\varepsilon, w\} \subset J(\alpha)$, donc $J(A) \subset f_S(J)(A)$.

2. Soit $A \in E \cap F$. $J(A) = \{\varepsilon\}$.

Puisque $A \in E$, il existe $\alpha \in d(A) \cap E^*$. D'après le cas 1 du lemme 10.2.5, on a nécessairement $\alpha \in F^*$. Donc toutes les variables B de α sont éléments de $E \cap F$, et, par définition de J , $J(B) = \{\varepsilon\}$.

Par suite $\{\varepsilon\} \subset J(\alpha)$, donc $J(A) \subset f_S(J)(A)$.

3. Soit $A \in F - E$. $J(A) = \emptyset$ donc $J(A) \subset f_S(J)(A)$.

Soit alors l'interprétation $K = \bigcup \{f_S^i(J) \mid i \geq 0\}$.

Puisque $I \subset f_S(J)$ et que f_S est une fonction continue (voir le théorème 8.3.11 du chapitre 10), on déduit, en remplaçant dans la preuve du théorème de Kleene (voir 8.2.4 du chapitre 10) l'élément \perp par J , que K est un point fixe de f_S , c'est-à-dire une solution du système S .

Par hypothèse, \bar{F} n'est pas vide. Soit $A \in \bar{F}$. De ce que $J \subset K$, on déduit que $w \in K(A)$. Puisque

$w \notin \bigcup \{I(A) \mid A \in V_N\}$, on sait que $w \notin I(A)$.
 Donc le système S a deux solutions distinctes. ■

Théorème 10.2.7 *Soit S un système d'équations qui est riche et F l'ensemble de ses variables potentiellement strictes. Les trois propriétés suivantes sont équivalentes :*

- (1) S n'a qu'une solution.
- (2) F est l'ensemble des variables de S
- (3) S est transformable par substitution en un système strict

PREUVE. (1) implique (2) : c'est la contraposée du lemme ci-dessus 10.2.6.

(2) implique (3) : c'est le lemme 10.2.4.

(3) implique (1) : Supposons que S est transformable par substitutions en un système strict S' . D'après le théorème 8.5.6 du chapitre 10, toute solution de S est solution de S' . D'après le théorème 10.1.3, le système strict S' n'a qu'une solution. Donc S n'a qu'une solution. ■

3 Problèmes de décision

Un problème est décidable, s'il y a un algorithme pour le résoudre. Nous renvoyons à l'ouvrage [?] pour plus de précisions sur ces notions.

3.1 Le problème «un système est-il riche ?» est indécidable

Système riche : rappel

Soit $S = \langle V_T, V_N, d \rangle$ un système et $V = V_T \cup V_N$. V_N est l'ensemble des variables et $d : V_N \mapsto \mathcal{P}(V)$ donne la définition de chaque variable.

L'ensemble R_S des règles de S est l'ensemble des règles $A \rightarrow \alpha$ telles que $\alpha \in d(A)$.

Soit A une variable. On note $G_{S,A}$ la grammaire $\langle V_T, V_N, R_S, A \rangle$ et $L(G_{S,A})$ le langage qu'elle engendre.

Le système S est riche si et seulement si $\bigcup \{L(G_{S,A}) \mid A \in V_N\}$ est un sous-ensemble strict de V_T^* .

3.2 Indécidabilité du problème «un système est-il riche ?»

Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte.

Le théorème 8.11 p203 du livre [?], nous dit que le problème « $L(G) = V_T^*$?» est indécidable.

Nicolas Peltier (<http://membres-liglab.imag.fr/peltier/>) a trouvé une réduction de ce problème au problème «un système est-il riche ?» : autrement dit, si le problème «un système est-il riche ?» était décidable, le problème « $L(G) = V_T^*$?» serait décidable.

Comme ce dernier problème n'est pas décidable, il en est de même du problème «un système est-il riche ?». On présente ci-dessous les principes de cette réduction.

Soit $G = \langle V_T, V_N, S, R \rangle$ une grammaire hors-contexte.

À cette grammaire on associe le système $S' = \langle V'_T, V'_N, d \rangle$ suivant :

1. V'_T est le vocabulaire terminal V_T auquel on ajoute la *nouvelle* lettre e .
2. V'_N est l'ensemble de variables V_N auquel on ajoute les nouvelles variables A, B, C, D, E .
3. Les règles de S' sont les suivantes
 - (a) Les règles de R
 - (b) $A = eS$.
Autrement dit A engendre l'ensemble des mots $eL(G)$.
 - (c) B engendre l'ensemble des mots ne commençant pas par le *nouveau* terminal e .
Les règles pour B sont : $B = \varepsilon + \{a \mid a \in V_T\}(C + CeB + CEB)$
 C engendre V_T^* et E engendre ee^+ .
 - (d) $C = \varepsilon + \{a \mid a \in V_T\}C$
 - (e) $E = ee + eE$
 - (f) D engendre l'ensemble des mots comportant au moins deux e .
Les règles pour D sont : $D = CeCeB + CeCEB$

On laisse au lecteur le soin de vérifier, qu'avec les règles ci-dessus, B engendre l'ensemble des mots ne commençant pas par e et D engendre l'ensemble des mots comportant au moins deux e .

Notons $U(S')$, l'union des langages engendrés par chacune des variables de S' .

D'après les propriétés de A, B, C, D, E , on a : $L(G_{S',C}) \subset L(G_{S',B})$ et $L(G_{S',E}) \subset L(G_{S',D})$.

Puisque toutes les variables de la grammaire G engendrent des mots sans e , donc éléments de $L(G_{S',B})$, il en résulte que : $U(S') = L(G_{S',A}) \cup L(G_{S',B}) \cup L(G_{S',D})$.

Montrons que $L(G) = V_T^*$ si et seulement si $U(S') = V_T'^*$ (c'est-à-dire si S' n'est pas riche).

1. Supposons $L(G) = V_T^*$.
D'après cette hypothèse et le fait que $A = eS$, on a : $U(S') = eV_T'^* \cup L(G_{S',B}) \cup L(G_{S',D})$.
Soit $w \in V_T'^*$.
 - (a) $w = \varepsilon$, donc $w \in L(G_{S',B})$ et par suite $w \in U(S')$
 - (b) w a une seule occurrence de e . On a deux cas suivant que cette occurrence est ou n'est pas la première lettre de w .
 - i. e est la première lettre de w . Donc $w \in eV_T'^*$ et par suite $w \in U(S')$.
 - ii. e n'est pas la première lettre de w . Donc $w \in L(G_{S',B})$ et par suite $w \in U(S')$.
 - (c) w a au moins deux occurrences de e .
Puis que D engendre l'ensemble des mots comportant au moins deux e , $w \in L(G_{S',D})$
et par suite $w \in U(S')$.

Par suite $w \in U(S')$, donc $U(S') = V_T'^*$.
2. Supposons $U(S') = V_T'^*$.
Soit $w \in V_T'^*$. Puisque ew n'est engendré ni par B , ni par D , $ew \in eL(G)$. Par suite $w \in L(G)$, donc $L(G) = V_T^*$

Le problème «un système a-t-il une seule solution ?» est relativement décidable

Soit S un système riche. Le problème « S a-t-il une seule solution ?» est décidable puisque, d'après le théorème ci-dessus 10.2.7, on peut le résoudre en calculant l'ensemble des variables potentiellement strictes de S .

Soit $S = \langle V_T, V_N, d \rangle$ un système quelconque. Si l'on est pas certain que ce système est riche, on peut le transformer en un système riche, en ajoutant à V_T des lettres qui ne figurent pas dans les règles de S .

Troisième partie

Annexes

Annexe A

Rappels sur l'induction

1 Compléments sur les relations d'ordre

Définition 1.1.1 Etant donné un ensemble E , une relation R sur E est une *relation d'ordre* si elle est :

- Réflexive : $\forall x \in E, xRx$;
- Antisymétrique : $\forall x, y \in E, (xRy \text{ et } yRx) \Rightarrow x = y$;
- Transitive : $\forall x, y, z \in E$, si xRy et yRz , alors xRz .

A toute relation d'ordre peut être associée une relation d'ordre strict.

Une relation R' sur E est une *relation d'ordre strict* si elle est :

- Irréflexive : $\forall x \in E, \neg(xR'x)$;
- Transitive.

◇

Exemple 1.1.2 Voici quelques exemples de relations d'ordre et des relations d'ordre strict associées.

1. La relation \leq est une relation d'ordre sur \mathbb{N} et sur \mathbb{Z} , à laquelle est associée la relation d'ordre strict $<$.
2. La relation \preceq définie sur $\mathbb{N} \times \mathbb{N}$ par : $(x, y) \preceq (x', y')$ si et seulement si $(x < x')$ ou bien $(x = x' \text{ et } y \leq y')$ est une relation d'ordre : c'est l'*ordre produit sur $\mathbb{N} \times \mathbb{N}$* . C'est également une relation d'ordre sur $\mathbb{Z} \times \mathbb{Z}$. L'ordre lexicographique strict \prec sur $\mathbb{N} \times \mathbb{N}$ est défini par : $(x, y) \prec (x', y')$ si et seulement si $(x < x')$ ou bien $(x = x' \text{ et } y < y')$
3. La relation \preceq définie sur \mathbb{N} par : $x \preceq y$ si et seulement si x divise y est une relation d'ordre. **Cette relation est-elle également une relation d'ordre sur \mathbb{Z} ?**
4. Etant donné un ensemble E , la relation d'inclusion \subseteq est une relation d'ordre sur l'ensemble des parties de E . La relation d'ordre strict qui lui est associée est l'inclusion stricte \subset .
5. Etant donné un vocabulaire V , la relation \preceq définie sur V^* par : $x \preceq y$ si et seulement si x est un préfixe de y est une relation d'ordre. C'est l'*ordre préfixe sur V^** . L'ordre préfixe strict \prec associé est défini par : $x \prec y$ si et seulement si x est un préfixe de y *distinct de y* .

6. La relation \preceq définie sur V^* par : $x \preceq y$ si et seulement si $|x| \leq |y|$ n'est pas une relation d'ordre (*Pourquoi ?*). Par contre, la relation \prec définie par : $x \prec y$ si et seulement si $|x| < |y|$ est une relation d'ordre stricte. ♣

Définition 1.1.3 Etant donnée une relation d'ordre \preceq sur un ensemble E , un élément $x \in E$ est un élément *minimal* dans E si aucun élément de E n'est plus petit que lui. Formellement, x est minimal dans E si $\forall y \in E, y \preceq x \Rightarrow y = x$. ◇

Exercice 1.1.4 Quelle différence y a-t-il entre la définition d'un élément minimal donnée ci-dessus et la suivante : " x est minimal dans E si $\forall y \in E, x \preceq y$ " ?

Exemple 1.1.5 Pour l'ensemble $\mathbb{N} \setminus \{0, 1\}$ muni de la relation de divisibilité, tous les nombres premiers sont minimaux. ♣

Définition 1.1.6 (Ordre bien fondé) Etant donné un ensemble E , un ordre strict \prec sur E est *bien fondé* si et seulement si tout sous-ensemble non-vide de E admet un élément minimal. ◇

Le théorème suivant fournit une définition équivalente d'un ordre bien fondé :

Théorème 1.1.7 L'ordre \prec est bien fondé si et seulement s'il n'existe aucune suite infinie décroissante dans E .

PREUVE. Supposons que \prec est bien fondé, et prouvons par contradiction qu'il n'existe aucune suite infinie décroissante dans E . Supposons qu'il existe une telle suite $u_1 \succ u_2 \succ \dots \succ u_n \succ \dots$ alors l'ensemble $X = \{u_i \mid i \geq 0\}$ serait un sous-ensemble de E ne contenant évidemment aucun élément minimal, ce qui contredit le fait que \prec est bien fondé.

Pour prouver la réciproque, nous supposons que \prec n'est pas bien fondé, et construisons une suite infinie décroissante dans E . Si \prec n'est pas bien fondé, alors il existe un ensemble $X \subseteq E$ non vide sans élément minimal. Soit $u_1 \in X$, alors, nécessairement, il existe $u_2 \in X$ tel que $u_2 \prec u_1$. Puis, toujours parce que X n'admet pas d'élément minimal, il existe $u_3 \in X$ tel que $u_3 \prec u_2$. En réitérant ce procédé, on construit une suite $u_1 \succ u_2 \succ \dots$ infinie décroissante dans E . ■

On voit pourquoi seul un ordre strict peut être bien fondé : si \preceq est une relation d'ordre (non strict), on peut toujours construire la suite infinie décroissante $x \succeq x \succeq x \succeq \dots$

Exemple 1.1.8 Les relations suivantes sont des ordres bien fondés :

- L'ordre $<$ sur \mathbb{N} ;
- La relation de division sur \mathbb{N} : $x \prec y$ si et seulement si $x|y$, et $x \neq y$;
- L'ordre lexicographique strict sur $\mathbb{N} \times \mathbb{N}$;
- L'ordre préfixe strict sur V^* . ♣

Exemple 1.1.9 Les relations suivantes ne sont **pas** des ordres bien fondés :

- L'ordre $<$ sur \mathbb{Z} ;
- L'ordre lexicographique strict sur $\mathbb{Z} \times \mathbb{Z}$;
- La relation d'inclusion stricte sur les parties de \mathbb{N} (considérer les ensembles $F_k = \{i \in \mathbb{N} \mid i \geq k\}$). ♣

Il peut être difficile de prouver qu'un ordre est bien fondé. Le lemme suivant fournit une condition suffisante pour qu'un ordre soit bien fondé.

Lemme 1.1.10 *Soit E un ensemble muni d'un ordre $<$, et soit E' un ensemble muni d'un ordre bien fondé \prec . S'il existe une fonction h de E vers E' telle que $\forall x, y \in E, x < y \Rightarrow h(x) \prec h(y)$, alors l'ordre $<$ est également bien fondé.*

PREUVE. Supposons que $<$ n'est pas un ordre bien fondé. Il existe donc une suite $(u_n)_{n \geq 0}$ d'éléments de E , infinie et strictement décroissante. Considérons alors la suite d'éléments de E' $(u'_n)_{n \geq 0}$, où pour tout $n \in \mathbb{N}$, $u'_n = h(u_n)$. Alors, par hypothèse, pour tout $n \in \mathbb{N}$, $u'_n \prec u'_{n+1}$, ce qui signifie que la suite $(u'_n)_{n \geq 0}$ est une suite infinie décroissante dans E' . Ceci est impossible puisque \prec est un ordre bien fondé. ■

2 Induction bien fondée

Une preuve par induction bien fondée est employée pour démontrer qu'une propriété P est vraie pour tous les éléments d'un ensemble E . Cette preuve peut se décomposer en les étapes suivantes :

- Définir un ordre strict bien fondé \prec sur E ;
- Choisir un élément $x \in E$ et supposer que pour tout $y \prec x$, la propriété P est vraie pour y ;
- Démontrer que la propriété P est vraie pour x également.

Formellement, étant donnée une propriété P , si pour l'ordre bien fondé \prec sur E , on peut prouver que :

$$\forall x \in E, ((\forall y \in E, (y \prec x) \Rightarrow P(y)) \Rightarrow P(x)),$$

alors on peut déduire que $\forall x \in E, P(x)$.

Théorème 1.2.1 *Le principe d'induction bien fondée est correct.*

Ce théorème peut se résumer ainsi : si une propriété P est vraie pour un élément $x \in E$ à chaque fois qu'elle est également vraie pour tous les éléments $y \prec x$, alors cette propriété est vraie pour tous les éléments $x \in E$.

PREUVE. Nous prouvons le théorème par l'absurde en supposant qu'une propriété P est vraie pour un élément $x \in E$ à chaque fois qu'elle est également vraie pour tous les éléments $y \prec x$, mais qu'il existe des éléments de E pour lesquels la propriété est fausse. Soit M l'ensemble des éléments pour lesquels la propriété est fausse : $M = \{x \in E \mid \neg P(x)\}$. Par hypothèse, M est non-vide, et comme \prec est un ordre bien fondé, M admet un élément minimal m .

- Comme m est un élément de M , par définition, m ne vérifie pas la propriété P .

- Comme m est un élément minimal de M , tous les éléments plus petits que m doivent être dans $E \setminus M$ et doivent donc vérifier la propriété P . Autrement dit, $\forall y \in E, y \prec m \Rightarrow P(y)$. Comme on a supposé que P est vraie pour un élément x à chaque fois qu'elle est également vraie pour tous les éléments $y \prec x$, on en déduit que m vérifie la propriété P , et on obtient une contradiction. ■

Exemple 1.2.2 Prouvons que tout entier naturel ≥ 2 est produit de nombres premiers. Soit $P(n)$ la propriété “ n est le produit de nombres premiers”, nous cherchons à prouver que pour tout $n \geq 2$, $P(n)$ est vraie. Pour cela, nous munissons $\mathbb{N} \setminus \{0, 1\}$ de la relation de division, qui est un ordre bien fondé. Les éléments minimaux pour cet ordre sont les nombres premiers, et il est clair que la propriété P est vraie pour ces éléments. Supposons que n n'est pas un élément minimal, il existe donc $p \geq 2$, différent de n tel que p divise n . Soit q tel que $n = pq$, alors $q \geq 2$ est également distinct de n . Ces deux éléments étant plus petits que n , par l'hypothèse d'induction, ils vérifient tous deux la propriété P , et sont donc des produits de nombres premiers. Donc, n est également un produit de nombres premiers. Par le principe d'induction bien fondée, on conclut que tout entier naturel ≥ 2 est le produit de nombres premiers. ♣

3 Induction structurelle

Le principe d'induction structurelle est un cas particulier de l'induction bien fondée, qui est employé quand l'ensemble E est défini par induction. Dans ce cas, E est donc défini comme étant le plus petit ensemble contenant un certain nombre d'éléments, les atomes, et qui est fermé pour un certain nombre d'opérations, appelées des constructeurs. Il est alors possible de munir E d'une relation d'ordre strict canonique : pour tout $x, y \in E$, $x \prec y$ si et seulement si y peut être obtenu en appliquant un nombre fini de constructeurs à x . Formellement, \prec est la *fermeture transitive* de la relation R définie pour tout $x, y \in E$ par : xRy si et seulement s'il existe un constructeur $f : E^k \rightarrow E$ et des éléments $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_k \in E$ tels que $y = f(x_1, \dots, x_{i-1}, x, x_{i+1}, \dots, x_k)$. Cet ordre est bien fondé (*Preuve ?*), et ses éléments minimaux sont les atomes de E .

Quand l'ensemble E est défini par induction, il est possible de montrer que tous les éléments de E satisfont une propriété P de la façon suivante :

Cas de base : vérifier que tous les atomes de E satisfont la propriété P ;

Induction : vérifier que pour tout constructeur $f : E^k \rightarrow E$, si les éléments x_1, \dots, x_k satisfont P , alors l'élément $f(x_1, \dots, x_k)$ satisfait également cette propriété.

Si ces deux étapes sont validées, il est possible de conclure que tous les éléments de E satisfont P .

Théorème 1.3.1 *Le principe d'induction structurelle est correct.*

PREUVE. La preuve de ce résultat est très simple : soit M l'ensemble des éléments qui satisfont la propriété P . Comme le cas de base du principe d'induction structurelle est vérifié, tous les atomes de E sont dans M . Comme l'hypothèse d'induction est également vérifiée, pour tout

constructeur f et pour tous les éléments x_1, \dots, x_k dans M , l'élément $f(x_1, \dots, x_k)$ est également dans M . On en déduit que $E \subseteq M$, ce qui prouve que $\forall x \in E, P(x)$ est vrai. ■

Ce résultat aurait également pu être prouvé en démontrant que toute preuve par induction structurelle peut être traduite en une preuve par induction bien fondée, en se servant de l'ordre sur les constructeurs mentionné dans l'introduction de ce paragraphe.

Exemple 1.3.2 Soit la grammaire hors-contexte $S \rightarrow aSbS \mid bSaS \mid \epsilon$, et montrons que tout mot du langage L engendré par cette grammaire contient autant de a que de b .

Preuve par induction structurelle. Le langage L engendré par la grammaire peut être défini inductivement comme suit :

- ϵ est dans L ;
- Si x et y sont dans L , alors $axby$ et $bxay$ sont dans L .

Ce langage est donc défini à partir d'un atome et de deux constructeurs. Appliquons le principe d'induction structurelle :

- ϵ contient autant de a que de b .
- Si x et y contiennent tous deux autant de a que de b , alors il est clair que $axby$ et $bxay$ contiennent également autant de a que de b .

D'où le résultat.

Preuve par induction bien fondée. Nous prenons comme ordre bien fondé la longueur des dérivations, qui correspond au nombre de règles de la grammaire appliquées à S .

- Si $S \Rightarrow^1 x$, alors nécessairement $x = \epsilon$ (l'atome du langage), et ϵ vérifie bien la propriété.
- Supposons que $S \Rightarrow^n x$, avec $n > 1$. Alors

$$\begin{aligned} S &\Rightarrow aSbS \Rightarrow^{n-1} aubv, \text{ ou bien} \\ S &\Rightarrow bSaS \Rightarrow^{n-1} buav. \end{aligned}$$

Supposons que $S \Rightarrow aSbS \Rightarrow^{n-1} aubv$, l'autre cas est similaire. Alors on a $S \Rightarrow^p u$ et $S \Rightarrow^q v$, où $p + q = n - 1$. Par l'hypothèse d'induction, u et v contiennent autant de a que de b , ce qui prouve que ceci est également le cas pour $aubv$. ♣

Annexe B

Eléments de corrections

Ce chapitre contient des propositions de solutions aux exercices, rédigées par des étudiants de l'Ensimag. L'intérêt serait que les coquilles soit corrigées dans ce chapitre, et qu'il soit complété et étendu d'année en année.

1 Langages, relations

Exercice 1.1.11 Montrons d'abord que si ρ implique σ , alors $\bar{\rho} \cup \sigma = E \times E$. Comme ρ et σ sont des relations sur E , il est clair que $\bar{\rho} \subseteq E$ et $\sigma \subseteq E$, on a donc $\bar{\rho} \cup \sigma \subseteq E$. Il s'agit maintenant de prouver que $E \subseteq \bar{\rho} \cup \sigma$. Soient $x, y \in E$, il faut montrer que $(x, y) \in \bar{\rho} \cup \sigma$. Si $(x, y) \in \bar{\rho}$, le résultat est démontré. Sinon, on a nécessairement $(x, y) \in \overline{(\bar{\rho})}$, et comme $\overline{(\bar{\rho})} = \rho$, on en déduit que $(x, y) \in \rho$. Comme ρ implique σ par hypothèse, on a bien le résultat.

Montrons maintenant que si $\bar{\rho} \cup \sigma = E \times E$, alors ρ implique σ . Ceci est évident : si $(x, y) \in \rho$, alors par définition $(x, y) \notin \bar{\rho}$ et donc, nécessairement $(x, y) \in \sigma$, d'où le résultat.

Exercice 1.2.8 Soit V un vocabulaire non vide. Nous allons montrer la propriété par induction sur n .

Cas de base : $V^0 = \{\varepsilon\}$, on a donc $|V^0| = 1 = (\#V)^0$.

Induction Soit $n \in \mathbb{N}$, et supposons que la propriété est vérifiée.

$$\begin{aligned} V^{n+1} &= \{w_1 \dots w_{n+1} \mid \forall i w_i \in V\} \\ &= \{ww_{n+1} \mid w \in V^n, w_{n+1} \in V\} \end{aligned}$$

$$\text{D'où } \#(V^{n+1}) = \#(V^n)\#(V) = (\#V)^n(\#V) = (\#V)^{n+1}$$

Conclusion Par propriété de récurrence on peut donc conclure :

$$\boxed{\forall n \in \mathbb{N}, \#(V^n) = (\#V)^n}$$

Exercice 1.2.14

1. Soit V un vocabulaire non vide, $n \in \mathbb{N}$ et $w \in V^n$.

Remarquons tout d'abord que $\forall i \in \{0 \dots n\}$ il existe toujours au moins une sous-chaine/un préfixe/un suffixe de longueur i . De plus en notant a un élément de V , la chaîne $w = a \dots a$ contient exactement une sous-chaine/un préfixe/un suffixe de longueur i , $\forall i$.

On en déduit que :

toute chaîne de longueur $n + 1$ admet au minimum n sous-chaînes/préfixes/suffixes.

Idée pour le maximum : $\sum n = \frac{(N+1)N}{2}$.

2. Commençons par démontrer le résultat pour une chaîne de longueur 4 exactement. Si la chaîne contient deux éléments égaux consécutifs, alors la sous chaîne de longueur 1 correspondante est non vide et apparaît de manière consécutive dans la chaîne. Sinon, la chaîne est forcément de la forme $abab$ ou $baba$ et contient donc une sous chaîne de longueur 2 (ab ou ba) qui apparaît de manière consécutive dans la chaîne. On étend le résultat à toute chaîne de longueur supérieure ou égale à 4 en remarquant qu'elle contient toujours une sous chaîne de longueur 4, et que toute sous chaîne d'une sous chaîne d'une chaîne est également une sous chaîne de cette chaîne.

Exercice 1.2.21 On a $|\varepsilon| = 0$ et $\forall (u, v) \in (V^*)^2 \quad |uv| = |u| + |v|$

L'application longueur de chaîne est donc bien par définition un homomorphisme de $\langle V^*, \cdot, \varepsilon \rangle$ dans $\langle \mathbb{N}, +, 0 \rangle$.

Exercice 1.2.30

1. Soit $L \in \mathcal{P}(V^*)$

$\emptyset.L = \{w \in V^* \mid w = uv, u \in \emptyset, v \in L\} = \emptyset$ car il n'existe aucun u tel que $u \in \emptyset$.

De même, on montre que $L.\emptyset = \emptyset$

2. Si $V = \emptyset$ on a :

$$\begin{aligned} \mathcal{P}(V^*) &= \bigcup_{i \geq 0} L^i \\ &= \{\varepsilon\} \cup \left(\bigcup_{i \geq 1} L^i \right) \\ &= \{\varepsilon\} \cup L \cdot \left(\bigcup_{i \geq 0} L^i \right) \\ &= \{\varepsilon\} \cup \emptyset \cdot \left(\bigcup_{i \geq 0} L^i \right) \\ &= \{\varepsilon\} \text{ d'après la question précédente} \end{aligned}$$

Exercice 4.1.3

1. Soit \mathcal{E} un ensemble de langages V fermé par la concaténation, l'étoile, et la réunion, et contenant les langages finis.

\mathcal{E} contenant les langages finis il contient alors par fermeture l'ensemble des langages réguliers. D'où $\{LangagesReguliers\} \subseteq \mathcal{E}$

2. Soit L_1 un langage contenant L et ϵ , stable par concaténation.

On note $\mathcal{P}(n) = \forall k \leq n, L^k \subset L_1$ et nous allons raisonner par récurrence.

Initialisation : $L^0 = \{\epsilon\}$ d'où $\mathcal{P}(0)$.

Récurrence On suppose $\mathcal{P}(n)$.

L_1 étant stable par concaténation, et contenant L et $\{L^k \mid k \leq n\}$ on a $L \cdot \{L^k \mid k \leq n\} = \{L^k \mid k \leq (n+1)\} \subset L_1$ soit $\mathcal{P}(n+1)$.

Par principe de récurrence, $L^* \subset L_1$, ce qui achève la démonstration.

3. — $(L^*)^* = \bigcup_{n \geq 0} (L^*)^n$ d'où en particulier $L^* \subset (L^*)^*$ ($n=1$).

Soit $w \in (L^*)^n$. Alors $w = w_1 w_2 \dots w_n$ tq $\forall i w_i \in L^*$.

Par définition on a alors $\forall w_i \exists n_i, tq w = v_{(i,1)} \dots v_{(i,n_i)}$ et $\forall j v_j \in L$.

Alors $w = v_{(1,1)} \dots v_{(1,n_1)} \dots v_{(n,1)} v_{(n,n_n)} \in L^{(\sum_{i=1}^n n_i)} \subset L^*$.

D'où $(L^*)^* \subset L^*$ et par double inclusion

$$\boxed{(L^*)^* = L^*}$$

— Si $\epsilon \in L$ alors $(L + \epsilon) = L$ est on a bien le résultat demandé.

Si $\epsilon \notin L$ alors comme $L \subset (L + \epsilon)$ on a trivialement $L^* \subset (L + \epsilon)^*$.

De plus, soit $w = v_1 \dots v_n \in (L + \epsilon)^*, tq \forall i v_i \in (L + \epsilon)$

En supprimant tous les epsilons, on obtient $w = v'_1 \dots v'_m \in L^m \subset L^*$ avec $0 < m \leq n$ ou bien $w = \epsilon \in L^0 \subset L^*$.

D'où $(L + \epsilon)^* \subset L^*$, et donc par double inclusion.

$$\boxed{(L + \epsilon)^* = L^*}$$

— Soit $w \in L^* \cdot L^*$. Alors $w = \epsilon \cdot w \in L^0 \cdot L^* \subset L^* \cdot L^*$. D'où $L^* \subset L^* \cdot L^*$.

Soit $w \in L^* \cdot L^*, tq w = uv, u \in L^n, v \in L^m$.

Alors $w \in L^{n+m} \subset L^*$, d'où $L^* \cdot L^* \subset L^*$.

Par double inclusion on a alors

$$\boxed{L^* \cdot L^* = L^*}$$

4. Pas encore corrigé

5. Pas encore corrigé

6. (a) Avec $L = \emptyset$, et $M \neq \emptyset$ l'équation se réécrit $\emptyset = M$ (\emptyset élément absorbant) qui n'as pas de solution.

(b) Soit L et M deux langages. On pose $X_0 = L^* M$.

On a alors $L \cdot X_0 + M = L \cdot L^* M + M = L \cdot L^* M + \epsilon \cdot M = L \cdot L^* M + L^0 \cdot M = (L \cdot L^* + L^0) \cdot M = L^* \cdot M$

Soit,

$$\boxed{X_0 = L \cdot X_0 + M, \text{ pour } X_0 = L^* M}$$

(c) Pas encore corrigé.

(d) Pas encore corrigé.

7. Pas encore corrigé

2 Systèmes de réécriture, grammaires

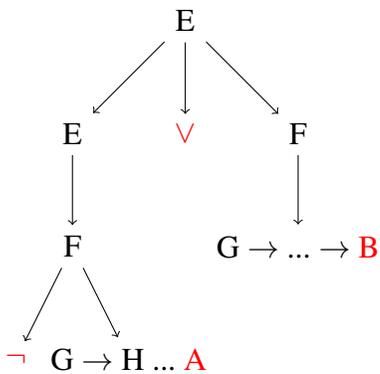
Exercice 6.2.6

1. $S \rightarrow aSa|bSb|\epsilon$
2. $S \rightarrow aSb|A|B$
 $A \rightarrow aA|a$
 $B \rightarrow bB|b$

Exercice 6.2.7

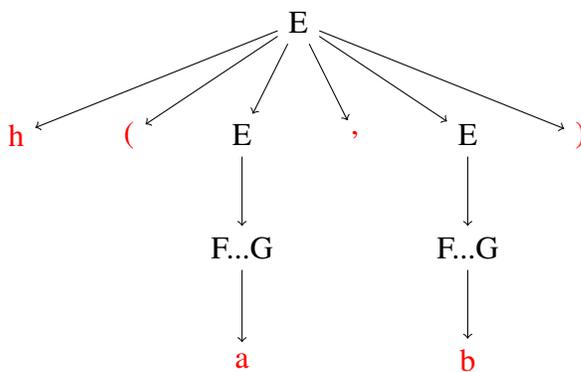
1. $E \rightarrow E \vee E|F$
 $F \rightarrow F \wedge F|G$
 $G \rightarrow \neg G|H$
 $H \rightarrow H \rightarrow H|I$
 $I \rightarrow L|(E)$
 $L \rightarrow AL|\dots|ZL|A|\dots|Z$

Exemple :



2. $E \rightarrow h(E, E)|F$
 $F \rightarrow f(F)|G|g(F)$
 $G \rightarrow a|b$

Exemple :



3. $G \rightarrow a|b|E = E|E < E$

4. $A \rightarrow A \vee A | B$
 $B \rightarrow B \wedge B | C$
 $C \rightarrow \neg C | D$
 $D \rightarrow D \rightarrow D | E$
 $E \rightarrow \forall X E | \exists X E | F$
 $F \rightarrow h(F, F) | G$
 $G \rightarrow f(G) | g(G) | H$
 $H \rightarrow a | b | X | F = F | F < F | (A)$
 $X \rightarrow xX | yX | zX | x | y | z$

Exercice 6.2.8

1. Sous-contexte.
2. $S \Rightarrow_1 aSA \Rightarrow_1 abSBA \Rightarrow_1 abcBA \Rightarrow_3 abcbA \Rightarrow_6 abcAb \Rightarrow_2 abcab$
3. Preuve par induction, en montrant la double inclusion...

Exercice 6.2.9

1. Le langage engendré par la grammaire $S \rightarrow bSS | a$ est $L(G) = \{b^n a^{n+1} | n \geq 0\}$.
2. Le langage engendré est $L(G) = \{\#a^{2^n}\# | n \geq 0\}$.

3 Grammaires hors-contexte

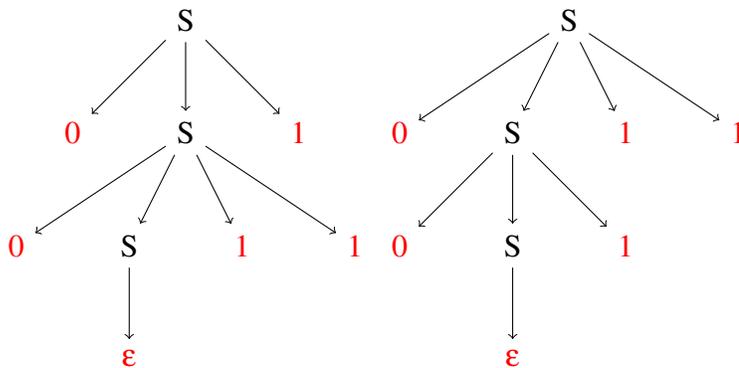
Exercice 7.5.2

1. Cette grammaire est ambiguë, car pour un même mot nous avons deux arbres de dérivations différents.
 $S \Rightarrow A \Rightarrow a$ et $S \Rightarrow B \Rightarrow a$
2. Une grammaire équivalente simple est : $S \rightarrow a$.

Exercice 7.5.3

1. Prenons le mot $\omega = 00111$. Nous obtenons les deux dérivations différentes suivantes :
 $S \Rightarrow 0S1 \Rightarrow 00S111 \Rightarrow 00111$ et $S \Rightarrow 0S11 \Rightarrow 00S111 \Rightarrow 00111$.

Illustration :

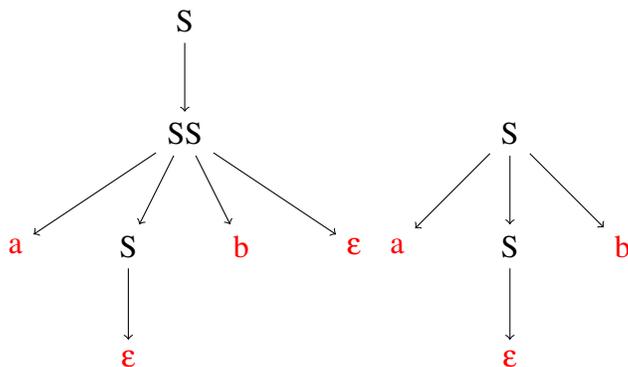


2. Preuve par induction sur la longueur de la dérivation...
3. ?
4. $S \rightarrow 0S1|A$
 $A \rightarrow 0A11|\epsilon$.

Exercice 7.5.4

1. Posons $\omega = ab$. Nous avons les dérivations suivantes :
 $S \Rightarrow SS \Rightarrow aSb\epsilon \Rightarrow ab = \omega$ et $S \Rightarrow aSb \Rightarrow a\epsilon b \Rightarrow ab = \omega$.

Illustration :

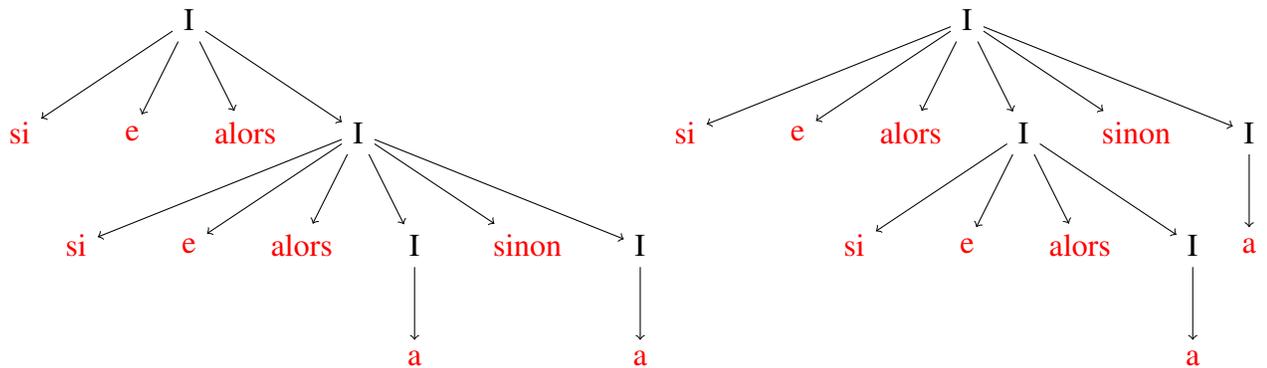


2. ?
3. ?

Exercice 7.5.6

1. On a la grammaire suivante : $I \rightarrow \text{si } e \text{ alors } I \mid \text{si } e \text{ alors } I \text{ sinon } I \mid a$.

Illustration :



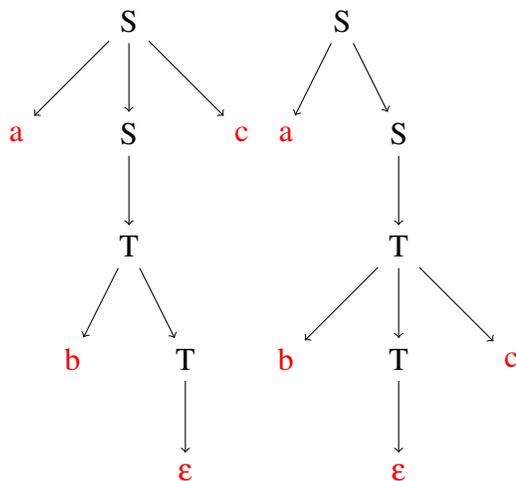
”si e alors si e alors a sinon a” est un mot obtenu par deux arbres différents, cela prouve bien que la grammaire est ambiguë.

2. $I \rightarrow \text{si } e \text{ alors } I \mid A$
 $A \rightarrow \text{si } e \text{ alors } A \text{ sinon } A \mid a$.

Exercice 7.5.7

1. $S \rightarrow aSc \mid aS \mid T$
 $T \rightarrow bTc \mid bT \mid \epsilon$

Illustration :



Le mot ”abc” se dérive de deux manières différentes, la grammaire est donc ambiguë.

2. Par induction, montrer que T engendre des mots de la forme $b^q c^n$ où $q \geq n$, puis en déduire le résultat.

$$\begin{aligned} 3. \quad & S \rightarrow aSc|A \\ & A \rightarrow aA|T \\ & T \rightarrow bTc|T' \\ & T' \rightarrow bT'|\epsilon \end{aligned}$$