ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Jacques Zahnd

AUTOMATES ET CALCULABILITÉ I

2002 - 2003

TABLE DES MATIÈRES

Chapitre 1 Préliminaires

1.1 Introduction

Ce chapitre présente un certain nombre de notions mathématiques générales utilisées dans la suite de ce cours et plus généralement en informatique théorique. Une partie de cette présentation, surtout dans la section 1.3, n'est qu'un rappel de notions de théorie des ensembles déjà étudiées dans le cours de logique élémentaire. Ces rappels sont donnés en général sans démonstrations, mais avec suffisamment d'explications pour que ce texte ne soit pas un simple recueil de formules, mais puisse être lu comme une brêve introduction.

La section 1.2 traite surtout de notations qui sont d'un usage très répandu mais qui ont été omises dans le cours de logique élémentaire. Il s'agit principalement d'un rappel et d'un complément sur les collections. Les sections 1.4 (ensembles ordonnés), 1.5 (récurrence), 1.6 (monoïdes) contiennent l'essentiel des notions nouvelles de théorie des ensembles par rapport à la matière du cours de logique élémentaire.

1.2 Notations logiques et ensemblistes

1.2.1. Quantificateurs typés

Dans les exposés mathématiques, les quantificateurs universels et existentiels $\forall x$, $\exists x$ sont très souvent *typés*. Cela veut dire que leur variable x est déclarée comme étant d'un certain *type*. Considérons par exemple les deux propositions

pour tout nombre réel x, $x^2 \geq 0$;
il existe un nombre réel x tel que $x^2 = 2$.

Dans ces deux phrases, on ne dit pas seulement « pour tout x » et « il existe un x », mais « pour tout *nombre réel* x » et « il existe un *nombre réel* x ». La notation que nous utiliserons souvent à la place de ces deux phrases est la suivante :

$\forall x \in \mathbb{R} : x^2 \geq 0$;
$\exists x \in \mathbb{R} : x^2 = 2$.

Ces notations se lisent : pour tout x, élément de $\mathbb{R}$, $x^2 \geq 0$; il existe un x, élément de $\mathbb{R}$, tel que $x^2 = 2$.

Les quantificateurs typés peuvent être considérés comme une notation abrégée. Par exemple, les deux assertions ci-dessus sur les nombres réels peuvent s'exprimer de la manière suivante, en utilisant des quantificateurs ordinaires, non typés :

$\forall x(x \in \mathbb{R} \Rightarrow x^2 \geq 0)$;
$\exists x(x \in \mathbb{R} \text{ et } x^2 = 2)$.

Définition générale de la notation. Si $P(x)$ est une proposition quelconque, considérée comme exprimant une propriété d'un objet x, et si A désigne un ensemble, les expressions

$$\forall x \in A : P(x), \qquad \exists x \in A : P(x) \tag{1}$$

sont des abréviations des propositions

$$\forall x(x \in A \Rightarrow P(x)), \qquad \exists x(x \in A \text{ et } P(x)). \tag{2}$$

Elles signifient respectivement que tout élément de A possède la propriété P ou qu'il existe un élément de A qui possède cette propriété.

Dans les notations abrégées (1), nous omettrons parfois les parenthèses extérieures de la proposition $P(x)$ lorsque celle-ci est une proposition composée. Par exemple, au lieu de

$$\exists x \in \mathbb{R} : (x \geq 0 \text{ et } x^2 = a),$$

nous écrirons parfois simplement $\exists x \in \mathbb{R} : x \geq 0 \text{ et } x^2 = a$.

Formules logiques. Les quantificateurs typés obéïssent à des formules semblables à celles des quantificateurs ordinaires. Nous mentionnons ici les deux plus importantes :

$$\begin{aligned} &\text{non}(\forall x \in A : P(x)) \Leftrightarrow (\exists x \in A : \text{non} P(x)). \\ &\text{non}(\exists x \in A : P(x)) \Leftrightarrow (\forall x \in A : \text{non} P(x)). \end{aligned} \tag{3}$$

La démonstration de ces équivalences, en utilisant la définition (2) des abréviations (1), est un exercice facile de logique élémentaire [A].

Une formule de théorie des ensembles. Si $A = \varnothing$, la proposition $x \in A$ est fausse et, par conséquent, la proposition $x \in A \Rightarrow P(x)$ est *vraie* (logique élémentaire), quelle que soit la proposition $P(x)$. On démontre grâce à cela la formule suivante :

$$A = \varnothing \Rightarrow (\forall x \in A : P(x)). \tag{4}$$

Nous dirons que si $A = \varnothing$, la proposition $\forall x \in A : P(x)$ est *trivialement vraie*. Il est trivialement vrai d'affirmer que tout élément de l'ensemble vide possède une certaine propriété, n'importe laquelle.

1.2.2. Collections simples

Une proposition $P(x)$, exprimant une certaine propriété d'un objet x, est dite *collectivisante en* x s'il existe un ensemble E tel que l'on ait

$$\forall x (x \in E \Leftrightarrow P(x)).$$

Cela signifie que l'ensemble E a pour éléments tous les objets x qui possèdent la propriété P et seulement ces objets-là. Cet ensemble E est alors appelé *l'ensemble des* x *tels que* $P(x)$, ce qui se note

$$E = \{x \mid P(x)\}.$$

Par définition, un objet a appartient à cet ensemble si et seulement s'il possède la propriété P en question. Autrement dit, on a la formule

$$\forall a : \quad a \in \{x \mid P(x)\} \Leftrightarrow P(a). \tag{1}$$

Un terme de la forme $\{x \mid P(x)\}$ est appelé une *collection*, plus exactement une *collection simple*, par opposition à une forme de collection plus générale introduite au §1.2.3. On rappelle que la variable x ne figure pas librement dans le terme $\{x \mid P(x)\}$. Le préfixe « $\{x \mid$ » lie la variable x comme un quantificateur.

Il y a des propositions $P(x)$ qui ne sont pas collectivisantes en x et pour lesquelles le terme $\{x \mid P(x)\}$ n'a pas de sens. Pour de telles propositions $P(x)$, la formule (1) est fausse. On ne fait donc jamais usage d'un terme de la forme $\{x \mid P(x)\}$ pour une proposition $P(x)$ qui n'est pas collectivisante en x. Mais nous admettrons toujours cette propriété sans la démontrer.

Une proposition de la forme ($x \in A$ et $P(x)$) est toujours collectivisante en x. Cela fait partie des axiomes de la théorie des ensembles. La collection

$$\{x \mid x \in A \text{ et } P(x)\}$$

est l'ensemble des éléments x de A qui vérifient $P(x)$. On a l'équivalence

$$a \in \{x \mid x \in A \text{ et } P(x)\} \Leftrightarrow (a \in A \text{ et } P(a)). \tag{2}$$

Le terme $\{x \mid x \in A \text{ et } P(x)\}$ est souvent abrégé de la manière suivante :

$$\{x \in A \mid P(x)\}.$$

Lire : l'ensemble des x, éléments de A, tels que $P(x)$. La proposition $a \in \{x \in A \mid P(x)\}$ équivaut donc à $a \in A$ et $P(a)$.

1.2.3. Collections générales

Une collection simple (1.2.2) est une expression de la forme $\{\ldots \mid \ldots\}$ dans laquelle, à gauche de la barre |, il y a seulement une *variable*, et à droite de la barre une proposition. Dans une collection *générale* $\{\ldots \mid \ldots\}$, il peut y avoir à gauche de la barre un *terme quelconque*, et pas seulement une variable.

Exemple 1. Considérons les ensembles de nombres $A = \{0, 1\}$ et $B = \{2, 4, 6\}$. On peut vouloir introduire la notation $A + B$ pour désigner l'ensemble de tous les nombres de la forme $x + y$, avec $x \in A$ et $y \in B$, à savoir

$$A + B = \{0 + 2, 0 + 4, 0 + 6, 1 + 2, 1 + 4, 1 + 6\} = \{2, 4, 6, 3, 5, 7\}. \tag{1}$$

Cette première égalité est une manière de définir l'ensemble $A + B$ qui n'est possible que parce que A et B sont des ensembles finis et pas trop grands. Cependant l'idée de construire un ensemble en prenant toutes les sommes possibles $x + y$ d'un élément x de A et d'un élément y de B peut s'appliquer à des ensembles de nombres A, B quelconques, finis ou infinis. Pour ne pas énumérer toutes les combinaisons comme dans (1), on écrit

$$A + B = \{x + y \mid x \in A \text{ et } y \in B\}. \tag{2}$$

Le membre de droite de cette égalité est ce que nous appelons une collection générale. C'est un terme qui se lit :

l'ensemble des $x + y$ tels que $x \in A$ et $y \in B$.

À gauche de la barre verticale figure le terme $x + y$. Il reste à définir précisément le sens exact de cette notation, pour des ensembles de nombres A et B quelconques. Pour cela, en considérant l'exemple (1), nous voyons que nous pouvons caractériser l'ensemble $A + B$ en disant ceci : un nombre z appartient à cet ensemble si et seulement s'il existe un nombre $x \in A$ et un nombre $y \in B$ tels que $z = x + y$. Autrement dit, l'ensemble $A + B$ est caractérisé par l'équivalence

$$z \in A + B \Leftrightarrow \exists x \exists y (z = x + y \text{ et } x \in A \text{ et } y \in B). \tag{3}$$

Il revient au même de poser

$$A + B = \{z \mid \exists x \exists y (z = x + y \text{ et } x \in A \text{ et } y \in B)\}. \tag{4}$$

On définit ainsi l'ensemble $A + B$ au moyen d'une collection simple (§1.2.2). Le membre de droite de (4) constitue la définition de celui de (2). C'est ce qui est exprimé dans la définition générale qui suit. Notons encore que les variables x et y ne figurent pas librement dans le terme $\{x + y \mid x \in A \text{ et } y \in B\}$. Ce sont des variables locales de ce terme.

Définition 1. Lorsqu'on définit un ensemble E au moyen d'une égalité

$$E = \{T(x_1, \ldots, x_n) \mid P(x_1, \ldots, x_n)\} \tag{5}$$

dans laquelle $T(x_1, \ldots, x_n)$ est un terme dépendant des variables $x_1, \ldots, x_n$ et $P(x_1, \ldots, x_n)$ est une proposition, le membre de droite de cette égalité est un terme qui se lit : l'ensemble des $T(x_1, \ldots, x_n)$ tels que $P(x_1, \ldots, x_n)$. Par définition, l'égalité (5) équivaut à

$$\forall z : \quad z \in E \Leftrightarrow \exists x_1 \ldots \exists x_n \big(z = T(x_1, \ldots, x_n) \text{ et } P(x_1, \ldots, x_n)\big). \tag{6}$$

Cela suppose évidemment que la proposition

$$\exists x_1 \ldots \exists x_n \big(z = T(x_1, \ldots, x_n) \text{ et } P(x_1, \ldots, x_n)\big)$$

est collectivisante en z. En vertu de la définition 1, au lieu de (5), on peut écrire aussi

$$E = \big\{z \,\big|\, \exists x_1 \ldots \exists x_n \big(z = T(x_1, \ldots, x_n) \text{ et } P(x_1, \ldots, x_n)\big)\big\}.$$

Les variables $x_1, \ldots, x_n$ ne figurent pas librement dans le membre de droite de (5).

Exemple 2. Nous considérons à nouveau deux ensembles de nombres A, B comme dans l'exemple 1, ainsi que l'ensemble $A + B$ défini par (2). Nous voulons noter ici une propriété intuitivement évidente de l'ensemble $A + B$, à savoir que l'implication suivante est vraie :

$$(x \in A \text{ et } y \in B) \Rightarrow x + y \in A + B. \tag{7}$$

Cette proposition se démontre facilement au moyen de la formule (3). En effet, supposons que $x \in A$ et $y \in A$ et posons $z = x + y$. On a aussitôt $z = x + y$ et $x \in A$ et $y \in B$, donc $\exists x \exists y(z = x + y \text{ et } x \in A \text{ et } y \in B)$, d'où $z \in A + B$, donc $x + y \in A + B$. Voir [A] pour une mise en forme de cette démonstration.

Remarquons que l'implication réciproque de (7) est fausse en général. Par exemple, dans le cas des ensembles $A = \{0, 1\}$ et $B = \{2, 4, 6\}$, on a

$$3 + 1 \in A + B \text{ et non}(3 \in A \text{ et } 1 \in B).$$

La proposition $3 + 1 \in A + B$ est vraie parce qu'il existe des nombres x et y tels que $3 + 1 = x + y$ et $x \in A$ et $y \in B$. Mais ces nombres x et y ne sont pas 3 et 1, mais 0 et 4.

La vérité de l'implication (7), en tant que conséquence de (2), est un cas particulier du théorème général suivant, dont la démonstration [A] généralise celle de (7).

Théorème 1. Si $E = \{T(x_1, \ldots, x_n) \mid P(x_1, \ldots, x_n)\}$, alors

$$P(x_1, \ldots, x_n) \Rightarrow T(x_1, \ldots, x_n) \in E.$$

Nous devrions parler plutôt d'un *schéma* de théorème, parce que, dans cet énoncé, $T(x_1, \ldots, x_n)$ et $P(x_1, \ldots, x_n)$ représentent un terme et un proposition indéterminés. Notons que l'implication réciproque $T(x_1, \ldots, x_n) \in E \Rightarrow P(x_1, \ldots, x_n)$ est fausse en général, comme nous l'avons remarqué dans l'exemple 2. Mais elle est vraie pour certains termes $T(x_1, \ldots, x_n)$ particuliers. Voir plus loin, théorème 3.

Exemple 3. Nous continuons les exemples 1 et 2 en relevant une deuxième propriété intuitivement évidente de l'ensemble $A + B$, à savoir que, si C est un ensemble quelconque, on a l'équivalence

$$A + B \subset C \Leftrightarrow \forall x \forall y((x \in A \text{ et } y \in B) \Rightarrow x + y \in C). \tag{8}$$

Esquissons-en une démonstration informelle. Premièrement, sous l'hypothèse $A + B \subset C$, on démontre immédiatement l'implication $(x \in A \text{ et } y \in B) \Rightarrow x + y \in C$ au moyen de (7). Inversement, sous l'hypothèse $\forall x \forall y((x \in A \text{ et } y \in B) \Rightarrow x + y \in C)$, on démontre $A + B \subset C$ en faisant l'hypothèse $z \in A + B$ et en démontrant $z \in C$. L'hypothèse

$z \in A + B$ signifie que z est un nombre de la forme $x + y$ avec $x \in A$ et $y \in C$. Un tel nombre appartient à C en vertu de la première hypothèse. Formellement, ce raisonnement contient une élimination des quantificateurs $\exists x \exists y$ provenant de l'utilisation de (3) [A].

La vérité de l'équivalence (8), en tant que conséquence de (2), est un cas particulier du théorème général suivant, dont la démonstration [A] généralise celle que nous venons d'esquisser.

Théorème 2. Si $E = \{T(x_1, \ldots, x_n) \mid P(x_1, \ldots, x_n)\}$, alors

$$E \subset C \Leftrightarrow \forall x_1 \cdots \forall x_n \big(P(x_1, \ldots, x_n) \Rightarrow T(x_1, \ldots, x_n) \in C\big).$$

Remarquons que le théorème 1 peut se déduire du théorème 2. Il suffit de faire $C = E$ dans ce dernier et d'utiliser le fait que $E \subset E$ est vrai.

Théorème 3 (collections de couples, resp. de n-uples). Si $E = \{(x, y) \mid P(x, y)\}$, autrement dit si E est l'ensemble des couples (x, y) pour lesquels une certaine proposition $P(x, y)$ est vérifiée, alors on a l'équivalence

$$(x, y) \in E \Leftrightarrow P(x, y),$$

et pas seulement l'implication $P(x, y) \Rightarrow (x, y) \in E$ donnée par le schéma de théorème 1. Le théorème 3 peut se généraliser pour une collection de n-uples : si E est l'ensemble des n-uples $(x_1, \ldots, x_n)$ pour lesquels une proposition $P(x_1, \ldots, x_n)$ est vérifiée, autrement dit si $E = \{(x_1, \ldots, x_n) \mid P(x_1, \ldots, x_n)\}$, alors on a l'équivalence

$$(x_1, \ldots, x_n) \in E \Leftrightarrow P(x_1, \ldots, x_n).$$

1.3 Relations et fonctions

Cette section contient essentiellement un rappel de définitions et de formules du cours de logique élémentaire sur les graphes et les fonctions. Les démonstrations formelles sont omises, mais les déductions principales sont explicitées de telle manière que ce texte ne soit pas un simple recueil de formules mais puisse être lu aussi comme une brève introduction.

Pour le lecteur qui maîtrise cette matière, nous signalons ici les quelques différences de terminologie et de notation et les quelques notions et notations supplémentaires qui sont introduites dans cette section, et auxquelles il doit prêter attention :

Un graphe G est appelé une *relation*. Les ensembles $\mathbf{Pr}_1 G$, $\mathbf{Pr}_2 G$ sont notés $\mathrm{Dom} G$, $\mathrm{Prt} G$ (domaine et portée de G), comme pour une fonction. Un couple (a, b) est noté aussi $a \mapsto b$. Les notions nouvelles sont : la *notation lambda* des fonctions (1.3.17), *l'axiome de choix* pour une famille d'ensembles (1.3.20), et les *restrictions* d'une fonction (1.3.21).

1.3.1. Relations

Le terme de *relation* est très souvent utilisé pour désigner ce que nous avons appelé un *graphe* dans le cours de logique élémentaire, à savoir un ensemble de couples. Une relation R est donc un ensemble dont tous les éléments sont des couples. L'égalité $R = R'$ de deux relations est une égalité d'ensembles (de couples). Elle équivaut à l'assertion : quels que soient x et y,

$$(x, y) \in R \Leftrightarrow (x, y) \in R'.$$

L'inclusion $R \subset R'$ équivaut à $\forall x \forall y \big((x, y) \in R \Rightarrow (x, y) \in R'\big)$.

Notation. Lorsque R est une relation, on écrit souvent

$$x\,R\,y$$

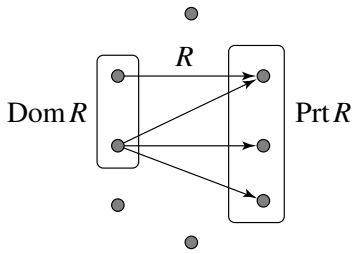
au lieu de $(x, y) \in R$. Les deux notations sont équivalentes. Par exemple, on peut écrire que deux relations R, R' sont égales si et seulement si, quels que soient x et y, on a $x\,R\,y \Leftrightarrow x\,R'\,y$.

1.3.2. Domaine et portée d'une relation

Étant donné une relation R, on dit qu'un objet x appartient au *domaine* de R s'il existe un objet y tel que le couple (x, y) appartienne à R, autrement dit tel que l'on ait $x\,R\,y$. Le domaine de R, noté $\mathrm{Dom}\,R$, est donc par définition l'ensemble des objets x qui vérifient la condition $\exists y(x\,R\,y)$. Symétriquement, l'ensemble appelé *portée* de R, ou $\mathrm{Prt}\,R$, est l'ensemble des objets y qui vérifient la condition $\exists x(x\,R\,y)$. Les ensembles $\mathrm{Dom}\,R$ et $\mathrm{Prt}\,R$ sont donc caractérisés par les propriétés suivantes de leurs éléments :

$$x \in \mathrm{Dom}\,R \Leftrightarrow \exists y(x\,R\,y).$$
$$y \in \mathrm{Prt}\,R \Leftrightarrow \exists x(x\,R\,y).$$

Cette définition est illustrée par le diagramme ci-dessous qui représente une relation R dans un ensemble fini de « points ». Les couples de points (x, y) appartenant à R sont les couples de points reliés par un flèche allant de x à y.



1.3.3. Relations dans un ensemble

Si R est une relation et si E est un ensemble tel que

$$R \subset E \times E,$$

on dit que R est une *relation dans* E. Cette inclusion signifie que chaque couple (x, y) appartenant à R est un couple d'éléments de E. Il revient au même de dire que l'on a

$$\mathrm{Dom}\,R \subset E \text{ et } \mathrm{Prt}\,R \subset E.$$

Si R est une relation dans E et si E' est un ensemble tel que $E \subset E'$, alors R est aussi une relation dans E'. Étant donné une relation R, il y a donc toujours une infinité d'ensembles E pour lesquels on peut dire que R est une relation dans E. Le plus petit de ces ensembles est l'ensemble $E = \mathrm{Dom}\,R \cup \mathrm{Prt}\,R$.

Exemple 1. L'ensemble des couples (x, y) d'entiers naturels tels que $x < y$, à savoir l'ensemble

$$R = \{(x, y) \mid (x, y) \in \mathbb{N} \times \mathbb{N} \text{ et } x < y\}, \tag{1}$$

est une relation dans l'ensemble $\mathbb{N}$. On l'appelle naturellement la relation *d'inégalité stricte* dans $\mathbb{N}$. En appliquant à cette collection de couples R le théorème 3 de 1.2.3, on a l'équivalence

$$(x, y) \in R \Leftrightarrow ((x, y) \in \mathbb{N} \times \mathbb{N} \text{ et } x < y). \tag{2}$$

Si x et y sont deux éléments de $\mathbb{N}$, autrement dit si la proposition $(x, y) \in \mathbb{N} \times \mathbb{N}$ est vraie, l'équivalence (2) se simplifie en

$$x\,R\,y \Leftrightarrow x < y.$$

Le domaine de cette relation R est l'ensemble $\mathbb{N}$ lui-même, alors que la portée de R est l'ensemble des entiers naturels non nuls, $\mathbb{N} - \{0\}$, ensemble complémentaire de $\{0\}$ par rapport à $\mathbb{N}$. Ces assertions sur $\mathrm{Dom}\,R$ et $\mathrm{Prt}\,R$ devraient être évidentes. (Voir [A] pour plus de précisions.)

NB. Il ne faut pas confondre les expressions $x\,R\,y$ et $x < y$. La variable R, dans l'expression $x\,R\,y$, n'est pas un symbole relationnel comme le symbole $<$, mais représente un ensemble de couples et l'expression $x\,R\,y$ est une abréviation de $(x, y) \in R$.

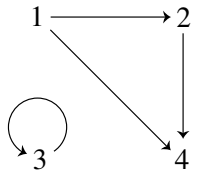
Notation abrégée. Au lieu de (1), on utilise souvent la notation

$$R = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid x < y\}.$$

Exemple 2. Le diagramme ci-dessous représente une relation R dans l'ensemble $E = \{1, 2, 3, 4\}$, à savoir l'ensemble de couples

$$R = \{(1, 2), (1, 4), (2, 4), (3, 3)\}.$$

Dans cet exemple, on a $\mathrm{Dom}\,R = \{1, 2, 3\}$ et $\mathrm{Prt}\,R = \{2, 3, 4\}$.

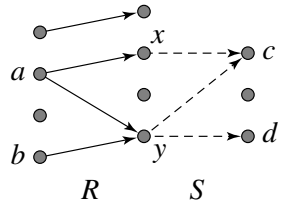


1.3.4. Composition de relations

Si R et S sont deux relations, la relation *composée* $S \circ R$ est définie comme l'ensemble des couples (a, b) qui vérifient la condition suivante : il existe un objet x tel que $(a, x) \in R$ et $(x, b) \in S$. Par définition, on a donc, quels que soient a, b :

$$\begin{aligned} a\,(S \circ R)\,b &\Leftrightarrow (a, b) \in S \circ R \\ &\Leftrightarrow \exists x\,(a\,R\,x \text{ et } x\,S\,b). \end{aligned}$$

Cette définition est illustrée dans le diagramme ci-dessous par deux relations R, S dans un ensemble de « points », les couples de la seconde étant représentés par des flèches en pointillé. La relation composée $S \circ R$, dans cet exemple, est l'ensemble de couples $\{(\mathtt{a}, \mathtt{c}), (\mathtt{a}, \mathtt{d}), (\mathtt{b}, \mathtt{c}), (\mathtt{b}, \mathtt{d})\}$.

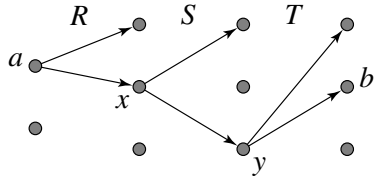


La composition de relations est *associative*, en ce sens que si R, S, T sont trois relations, on a

$$T \circ (S \circ R) = (T \circ S) \circ R.$$

Cette propriété de la composition de relations est illustrée par le diagramme ci-dessous, dans lequel on « voit » que les propositions $(a, b) \in T \circ (R \circ S)$, $(a, b) \in (T \circ R) \circ S$ sont équivalentes, parce qu'elles sont toutes deux équivalentes à

$$\exists x \exists y (a\, R\, x \text{ et } x\, S\, y \text{ et } y\, T\, b).$$



$$(a, b) \in (T \circ (R \circ S)) \Leftrightarrow (a, b) \in ((T \circ R) \circ S).$$

En vertu de cette associativité, on peut écrire $T \circ S \circ R$ sans parenthèses.

Notons finalement cette propriété évidente : si R et S sont des relations dans un ensemble E (1.3.3), alors la relation composée $S \circ R$ est aussi une relation dans E.

1.3.5. Exercice

Dans chacun des exemples du §1.3.3, décrire la relation $R \circ R$. Dans l'exemple 1, on donnera une expression de la forme

$$R \circ R = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid \dots\}$$

avec une proposition adéquate à la place des points … Dans l'exemple 2, on donnera une liste explicite des couples appartenant à $R \circ R$.

1.3.6. La relation vide

Pour prouver qu'un ensemble R est une relation (§1.3.1), on doit montrer que tout élément de cet ensemble est un couple, c'est-à-dire formellement que :

$$\forall z (z \in R \Rightarrow z \text{ est un couple}). \tag{1}$$

Dire que z est un couple signifie qu'il existe des objets a, b tels que $z = (a, b)$.

La proposition (1) est trivialement vraie pour l'ensemble particulier $R = \varnothing$, car la proposition $z \in \varnothing$ est fausse et par conséquent l'implication ($z \in \varnothing \Rightarrow z$ est un couple) est vraie (logique élémentaire : le faux implique tout). Par conséquent on peut affirmer que *l'ensemble $\varnothing$ est une relation*. On l'appelle naturellement la *relation vide*.

Le domaine et la portée de la relation $\varnothing$ sont évidemment vides. En outre, la relation vide est l'unique relation dont le domaine est vide, ainsi que l'unique relation dont la portée est vide. Autrement dit, pour qu'une relation R soit vide, il faut et il suffit que son domaine (resp. sa portée soit vide). Formellement, si R est une relation, on a

$$\begin{cases} R = \varnothing \Leftrightarrow \operatorname{Dom} R = \varnothing; \\ R = \varnothing \Leftrightarrow \operatorname{Prt} R = \varnothing. \end{cases} \tag{2}$$

Finalement, pour toute relation R, on a

$$R \circ \varnothing = \varnothing \circ R = \varnothing. \tag{3}$$

1.3.7. Relation opposée d'une relation

Si R est une relation, on appelle *relation opposée* de R et l'on note R^{-1} l'ensemble des couples (x, y) tels que $(y, x) \in R$. Par définition, on a donc

$$x\, R^{-1}\, y \Leftrightarrow y\, R\, x.$$

Dans la représentation d'une relation par un diagramme de points et de flèches, le diagramme de R^{-1} s'obtient à partir de celui de R en inversant le sens des flèches. On a évidemment

(1) $$\mathrm{Dom}(R^{-1}) = \mathrm{Prt}\,R \text{ et } \mathrm{Prt}(R^{-1}) = \mathrm{Dom}\,R.$$

La relation opposée d'une relation composée obéit à la formule suivante: si R et S sont des relations,

(2) $$(S \circ R)^{-1} = R^{-1} \circ S^{-1}.$$

1.3.8. La relation identique d'un ensemble

Pour tout ensemble E, la *relation identique de* E, notée Id_E, est l'ensemble des couples (x, x) tels que $x \in E$. La formule qui définit cette relation est la suivante:

(1) $$(x, y) \in \mathrm{Id}_E \Leftrightarrow (x \in E \text{ et } x = y).$$

On peut écrire naturellement $x\,(\mathrm{Id}_E)\,y$ au lieu de $(x, y) \in \mathrm{Id}_E$.

Il est clair que $(x, y) \in \mathrm{Id}_E$ implique $(x, y) \in E \times E$, donc que $\mathrm{Id}_E \subset E \times E$. Autrement dit, Id_E est une relation dans E (1.3.3). Cette relation est *neutre* pour l'opération de composition de relations dans E. Cela signifie que, pour toute relation R dans E, on a

(2) $$\mathrm{Id}_E \circ R = R \text{ et } R \circ \mathrm{Id}_E = R.$$

1.3.9. Fonctions

Une *fonction* F est une relation (ensemble de couples) qui satisfait à la condition:

(1) $$\forall x \forall y \forall y' : \quad (x\,F\,y \text{ et } x\,F\,y') \Rightarrow y = y'.$$

Cette propriété s'exprime en disant que pour tout objet x, il existe au plus un objet y tel que $(x, y) \in F$.

Si x est un objet qui appartient au domaine de F, alors, par définition du domaine d'une relation (1.3.2), il *existe* un objet y tel que $x\,F\,y$ et alors, en vertu de (1), il en existe un et un seul. On peut donc énoncer: si F est une fonction, alors

(2) $$x \in \mathrm{Dom}\,F \Rightarrow \exists! y (x\,F\,y).$$

1.3.10. Image d'un objet par une fonction

Si F est une fonction et si $x \in \mathrm{Dom}\,F$, alors l'unique objet y tel que $x\,F\,y$ est désigné traditionnellement par $F(x)$. On le note aussi $F\,@\,x$ pour mettre en évidence l'opération binaire qui est en jeu, à savoir l'opération d'application d'une fonction à un argument. La formule principale, relative à cette opération est la suivante:

$$x\,F\,y \Leftrightarrow (x \in \mathrm{Dom}\,F \text{ et } y = F(x)).$$

Si F est une fonction, l'objet $F(x)$ correspondant à un élément x de $\mathrm{Dom}\,F$ est appelé de diverses manières: l'image de x par F; la valeur de F correspondant à l'argument x; la valeur retournée par F pour la donnée x (langage des informaticiens), etc.

1.3.11. Portée d'une fonction

Une fonction étant un cas particulier de relation, les ensembles $\mathrm{Dom}\,F$ et $\mathrm{Prt}\,F$, pour une fonction F, sont définis comme au §1.3.2. En particulier, l'ensemble $\mathrm{Prt}\,F$ est caractérisé par $y \in \mathrm{Prt}\,F \Leftrightarrow \exists x (x\,F\,y)$. En utilisant la formule du §1.3.10, cette caractérisation de $\mathrm{Prt}\,F$ peut se reformuler

(1) $$y \in \mathrm{Prt}\,F \Leftrightarrow \exists x (x \in \mathrm{Dom}\,F \text{ et } y = F(x)).$$

Il revient au même de dire que $\mathrm{Prt}\,F = \{F(x) \mid x \in \mathrm{Dom}\,F\}$ (1.2.3, Déf. 1).

1.3.12. Exemples : fonctions finies

Un ensemble de couples qui comporte un seul élément, autrement dit un ensemble de la forme $F = \{(a, b)\}$, est une fonction. Nous utiliserons souvent la notation $a \mapsto b$ pour un couple (a, b) :

$$a \mapsto b \;=\; (a, b)$$

surtout lorsque ce couple est élément d'une fonction ou plus généralement d'une relation. Le domaine de la fonction $F = \{a \mapsto b\}$ est l'ensemble $\mathrm{Dom}\,F = \{a\}$ et l'on a $F(a) = b$.

Un ensemble de couples à deux éléments, $G = \{a_1 \mapsto b_1, a_2 \mapsto b_2\}$ tel que $a_1 \neq a_2$ est une fonction, quels que soient par ailleurs b_1 et b_2 (distincts ou non). On a $\mathrm{Dom}\,G = \{a_1, a_2\}$, $G(a_1) = b_1$, $G(a_2) = b_2$. Par contre, si $a_1 = a_2$ et $b_1 \neq b_2$, l'ensemble G n'est pas une fonction. Dans ce cas, G est simplement une *relation*.

Plus généralement, un ensemble de couples fini à n éléments, c'est-à-dire comprenant n couples distincts ($n \in \mathbb{N}$),

$$H = \{a_1 \mapsto b_1, \ldots, a_n \mapsto b_n\}$$

est une fonction si son domaine, $\mathrm{Dom}\,H = \{a_1, \ldots, a_n\}$, comporte aussi n éléments distincts, c'est-à-dire si $a_i \neq a_j$ pour $i \neq j$. Le fait que le nombre d'éléments de l'ensemble $\mathrm{Dom}\,H$ soit égal à celui de l'ensemble H entraîne en effet que pour tout élément a de $\mathrm{Dom}\,H$ il y a un seul couple $a_i \mapsto b_i$ de H tel que $a_i = a$, et l'on a donc $H(a_i) = b_i$ pour $i = 1, \ldots, n$.

Un cas particulier important auquel nous aurons affaire constamment dans la suite est celui d'un ensemble de couples fini de la forme

$$S = \{1 \mapsto b_1, \ldots, n \mapsto b_n\}$$

dont le domaine est l'ensemble $\{1, 2, \ldots, n\} = \{k \in \mathbb{N} \mid 1 \leq k \leq n\}$. Une telle fonction S sera appelée une *séquence* (chap. 2) et représentée par la notation $S = \langle\!\langle\, b_1, \ldots, b_n \,\rangle\!\rangle$. Pour tout $k \in \mathrm{Dom}\,S$, on a $S(k) = b_k$.

1.3.13. Égalité de deux fonctions

L'égalité $F = G$ de deux fonctions F, G est l'égalité de ces deux ensembles de couples, autrement dit l'équivalence $\forall x \forall y : x\,F\,y \Leftrightarrow x\,G\,y$. En appliquant à cette équivalence la formule du §1.3.10, on démontre facilement que pour deux fonctions F, G :

$$F = G \Leftrightarrow \big[\,\mathrm{Dom}\,F = \mathrm{Dom}\,G \text{ et } \forall x \in \mathrm{Dom}\,F :\; F(x) = G(x)\,\big].$$

1.3.14. Applications

Nous disons que F *est une application* d'un ensemble A dans un ensemble B, et nous écrivons $F : A \to B$, si F est une fonction et si l'on a $\mathrm{Dom}\,F = A$ et $\mathrm{Prt}\,F \subset B$. Cette définition peut se formuler :

$$\big(\,F : A \to B\,\big) \Leftrightarrow \big(\,F \text{ est une fonction et } \mathrm{Dom}\,F = A \text{ et } \mathrm{Prt}\,F \subset B\,\big).$$

Le prédicat « est une application » est un prédicat ternaire, qui demande trois arguments : F, A, B. On ne peut pas dire simplement « F est une application ». Par contre, le prédicat « est une fonction » est unaire. On peut dire « F est une fonction », le sens de cette assertion étant défini au §1.3.9.

Bien que l'expression $F : A \to B$ soit une proposition (ou assertion) et non un terme, il arrive souvent qu'on parle « d'une fonction $F : A \to B$ ». Cette tournure de langage est évidemment une manière abrégée de dire : *une fonction F telle que $F : A \to B$*.

Le sens de l'assertion $F : A \to B$ peut être explicité d'une autre manière en utilisant la formule 1.3.11 (1) qui caractérise l'ensemble $\mathrm{Prt}\, F$, à savoir :

$$\big(F : A \to B\big) \Leftrightarrow \big(F \text{ est une fonction et } \mathrm{Dom}\, F = A \text{ et } (\forall x \in A : F(x) \in B)\big).$$

1.3.15. La fonction vide

L'ensemble $\varnothing$ est non seulement une relation (1.3.6) mais encore une fonction (1.3.9) car, pour $F = \varnothing$, l'implication

$$((x, y) \in F \text{ et } (x, y') \in F) \Rightarrow y = y'$$

est trivialement vraie, puisque la proposition $((x, y) \in \varnothing$ et $(x, y') \in \varnothing)$ est fausse.

Nous parlerons donc aussi de l'ensemble $\varnothing$ comme étant la *fonction vide*. D'après 1.3.6 (2), cette fonction est l'unique fonction dont le domaine est vide, ainsi que l'unique fonction dont la portée est vide.

Étant donné un ensemble B quelconque, on peut dire que la fonction $\varnothing$ est une application de l'ensemble $\varnothing$ dans B, puisque $\mathrm{Dom}\varnothing = \varnothing$ et $\mathrm{Prt}\varnothing = \varnothing \subset B$. On a donc $\varnothing : \varnothing \to B$. D'autre part la fonction vide est l'unique fonction qui est une application de $\varnothing$ dans B, puisque $F : \varnothing \to B$ implique $\mathrm{Dom}\, F = \varnothing$, donc $F = \varnothing$. On a donc

$$(F : \varnothing \to B) \Leftrightarrow F = \varnothing.$$

1.3.16. Fonctions constantes

Une fonction F est dite *constante* si l'on a $F(x) = F(x')$ quels que soient $x, x' \in \mathrm{Dom}\, F$. On peut exprimer cette propriété d'une autre manière en disant qu'il existe un objet a tel que $F(x) = a$ pour tout $x \in \mathrm{Dom}\, F$, ce qui peut s'exprimer encore par

$$\exists a : \ F = (\mathrm{Dom}\, F) \times \{a\}.$$

D'après cette formule, la fonction vide est une fonction constante, car pour $F = \varnothing$, l'égalité $F = (\mathrm{Dom}\, F) \times \{a\}$ est vraie pour n'importe quel objet a, puisque $\mathrm{Dom}\varnothing = \varnothing$ et $\varnothing \times B = \varnothing$ pour tout ensemble B.

1.3.17. La notation lambda

Cette notation des fonctions est peu utilisée par les mathématiciens, mais joue un rôle important en informatique, dans l'étude des langages de programmation dits « fonctionnels ». Considérons par exemple l'expression suivante, qui désigne une fonction bien déterminée :

(1)
$$\begin{array}{l}\text{la fonction } F \text{ de domaine } \mathbb{R} \text{ telle que,}\\ \text{pour tout } x \in \mathbb{R} : \ F(x) = x^2 + 1.\end{array}$$

En pratique, on utilise diverses formes plus ou moins abrégées de cette expression, par exemple en parlant de

$$\text{la fonction } F(x) = x^2 + 1 \ (x \in \mathbb{R}).$$

Ces expressions contiennent deux choses : d'une part un nom donné à cette fonction, à savoir le nom « F », et d'autre part une description de l'ensemble de couples qu'est cette fonction F. On peut exprimer ces deux choses de manière plus claire, c'est-à-dire en les séparant plus clairement, au moyen d'une simple *égalité* de la forme

$$F = \ldots$$

Il suffit que le membre de droite soit un terme décrivant précisément l'ensemble de couples en question. Or la fonction considérée est l'ensemble des couples $(x, x^2 + 1)$ tels que $x \in \mathbb{R}$.

Cela s'écrit

$$F = \{(x, x^2 + 1) \mid x \in \mathbb{R}\} \tag{2}$$

ou

$$F = \{x \mapsto x^2 + 1 \mid x \in \mathbb{R}\}$$

en utilisant la notation $a \mapsto b$ pour un couple (a, b).

La « notation λ » (lettre grecque lambda) n'est qu'une autre manière d'écrire le membre de droite de l'égalité (2), à savoir :

$$F = (\lambda x \in \mathbb{R} : x^2 + 1). \tag{3}$$

L'expression $(\lambda x \in \mathbb{R} : x^2 + 1)$ doit être considérée comme une simple variante (un peu abrégée) de l'expression $\{(x, x^2 + 1) \mid x \in \mathbb{R}\}$. Elle est d'autre part assez « parlante » en tant que description de fonction. L'égalité (3) peut se lire en effet de la manière suivante : F est la fonction de domaine $\mathbb{R}$ qui, à tout $x \in \mathbb{R}$, associe (ou fait correspondre) le nombre $x^2 + 1$.

Il faut noter que la variable x est liée par le préfixe λx, comme par un quantificateur. On a l'égalité suivante (par changement de variable liée) :

$$(\lambda x \in \mathbb{R} : x^2 + 1) = (\lambda y \in \mathbb{R} : y^2 + 1).$$

La définition générale de cette notation est donnée ci-dessous.

Définition 1. Si $\mathbf{T}$ est un terme quelconque, par exemple le terme $x^2 + 1$ de l'exemple précédent, on désigne par

$$(\lambda x \in A : \mathbf{T})$$

l'ensemble des couples $x \mapsto \mathbf{T}$ tels que $x \in A$. Autrement dit :

$$(\lambda x \in A : \mathbf{T}) = \{x \mapsto \mathbf{T} \mid x \in A\}.$$

Le plus souvent, la variable x figure librement dans le terme $\mathbf{T}$, comme dans l'exemple du terme $x^2 + 1$, mais pas toujours. Par exemple une fonction constante F (1.3.16) de domaine X telle que $F(x) = a$ pour tout $x \in X$ est l'ensemble de couples $\{x \mapsto a \mid x \in X\}$. Avec la notation lambda, elle s'écrit $F = (\lambda x \in X : a)$. Le terme $\mathbf{T}$ est dans ce cas la variable a, et la variable x ne figure pas dans $\mathbf{T}$.

Si F est une fonction et $A = \mathrm{Dom}\, F$, il est clair que F est l'ensemble des couples $x \mapsto F(x)$ tels que $x \in A$, autrement dit que

$$F = \{x \mapsto F(x) \mid x \in A\}.$$

Inversement, de cette égalité, on déduit que F est une fonction de domaine A. Compte tenu de la définition 1, on a donc

$$(F \text{ est une fonction et } \mathrm{Dom}\, F = A) \Leftrightarrow F = (\lambda x \in A : F(x)). \tag{4}$$

1.3.18. Familles

Une fonction F est appelée parfois une *famille*. Les deux mots sont synonymes. C'est une erreur de prendre le mot « famille » comme synonyme du mot « ensemble ». Lorsqu'une fonction F est appelée une famille, on utilise une terminologie et des notations particulières. L'ensemble $A = \mathrm{Dom}\, F$ est appelé *l'ensemble des indices* de la famille, et si $x \in A$, on écrit F_x au lieu de $F(x)$, d'où la dénomination « ensemble des indices » pour A. La λ-expression

$$(\lambda x \in A : F_x) \tag{1}$$

qui représente la fonction F (1.3.17(4)) s'écrit alors traditionnellement

$$(F_x)_{x \in A} \tag{2}$$

ou parfois $(F_x \mid x \in A)$.

La famille vide. Si $A = \emptyset$ l'expression $(F_x)_{x \in A}$ désigne l'unique fonction F dont le domaine est vide, à savoir la fonction vide, $F = \emptyset$ (1.3.15). On parle dans ce cas de la famille vide.

Suites. Une famille $(F_n)_{n \in \mathbb{N}}$, dont l'ensemble d'indices est l'ensemble $\mathbb{N}$ des entiers naturels — autrement dit une fonction F de domaine $\mathbb{N}$ — est appelée une *suite*. En analyse, une telle famille est souvent notée simplement (F_n), en omettant le « $n \in \mathbb{N}$ ». Cette omission ne doit pas faire oublier que la variable n ne figure pas librement dans le terme $(F_n)_{n \in \mathbb{N}}$ qui est abrégé par (F_n). L'expression $(F_n)_{n \in \mathbb{N}}$ est une variante de l'expression $(\lambda n \in \mathbb{N} : F_n)$, dans laquelle n ne figure pas librement.

1.3.19. Réunion d'une famille d'ensembles

Une famille d'ensembles $(A_i)_{i \in I}$ est une fonction A de domaine I (§1.3.18) telle que, pour tout $i \in I$, A_i est un ensemble. On appelle *réunion* d'une telle famille et l'on note

$$\bigcup_{i \in I} A_i$$

l'ensemble de tous les objets qui appartiennent à l'un au moins des ensembles A_i. Cela signifie plus précisément que l'on a :

$$x \in \bigcup_{i \in I} A_i \Leftrightarrow \exists i (i \in I \text{ et } x \in A_i). \tag{1}$$

Le membre de droite de cette équivalence s'abrège souvent $\exists i \in I : x \in A_i$ (1.2.1). La formule (1) permet de démontrer en particulier que

$$\bigcup_{i \in \emptyset} A_i = \emptyset. \tag{2}$$

1.3.20. L'axiome de choix

La proposition suivante est un axiome de la théorie des ensembles : Si $(A_i)_{i \in I}$ est une famille d'ensembles non vides, c'est-à-dire si l'on a

$$\forall i \in I : A_i \neq \emptyset,$$

alors il existe une fonction α de domaine I telle que, pour tout $i \in I$,

$$\alpha(i) \in A_i.$$

L'idée de cet axiome est que si tous les ensembles A_i sont non vides, alors on peut « construire » une telle fonction α en « choisissant », pour chaque $i \in I$ un élément de A_i que l'on désigne par $\alpha(i)$ et que l'on associe à i.

1.3.21. Restrictions d'une fonction

Si F est une fonction et si $A \subset \mathrm{Dom}\, F$, la fonction

$$(\lambda x \in A : F(x)),$$

autrement dit l'ensemble de couples $\{x \mapsto F(x) \mid x \in A\}$, est appelée la *restriction* de la fonction F à l'ensemble A, et notée souvent $F_{|A}$. On a donc

$$\mathrm{Dom}(F_{|A}) = A$$

et pour tout x : $$x \in A \Rightarrow F_{|A}(x) = F(x).$$

1.3.22. Composition de deux fonctions

Deux fonctions F, G étant des relations (1.3.9), la relation composée $G \circ F$ (1.3.4) est caractérisée par la formule

(1) $$x\,(G \circ F)\,z \;\Leftrightarrow\; \exists y(x\,F\,y \text{ et } y\,G\,z).$$

D'après 1.3.10, on peut reformuler (1) en remplaçant $x\,F\,y$ par ($x \in \mathrm{Dom}F$ et $y = F(x)$) et en remplaçant $y\,G\,z$ semblablement. On obtient alors:

(2) $$x\,(G \circ F)\,z \;\Leftrightarrow\; \big[\,x \in \mathrm{Dom}F \text{ et } F(x) \in \mathrm{Dom}G \text{ et } z = G(F(x))\,\big].$$

On en tire le théorème suivant.

Théorème. Si F et G sont des fonctions, alors

$$\begin{aligned}
&G \circ F \text{ est une fonction;}\\
&x \in \mathrm{Dom}(G \circ F) \;\Leftrightarrow\; (x \in \mathrm{Dom}F \text{ et } F(x) \in \mathrm{Dom}G\,);\\
&x \in \mathrm{Dom}(G \circ F) \;\Rightarrow\; (G \circ F)(x) = G(F(x)).
\end{aligned}$$

Si, en outre, $\mathrm{Prt}F \subset \mathrm{Dom}G$, alors

$$\mathrm{Dom}(G \circ F) = \mathrm{Dom}F.$$

Démonstration. Premièrement, l'ensemble de couples $H = G \circ F$ est une fonction (1.3.9) parce que les hypothèses $(x, z) \in H$ et $(x, z') \in H$ entraînent d'après (2) que $z = z' = G(F(x))$.

Deuxièmement, la proposition $x \in \mathrm{Dom}(G \circ F)$ équivaut à $\exists z((x, z) \in G \circ F)$ (1.3.2), ce qui équivaut d'après (2) à ($x \in \mathrm{Dom}F$ et $F(x) \in \mathrm{Dom}(G)$).

Troisièmement, si l'on suppose que $x \in \mathrm{Dom}(G \circ F)$ et si l'on pose $z = (G \circ F)(x)$, alors il vient $(x, z) \in G \circ F$ (1.3.10) et l'on en tire $z = G(F(x))$ d'après (2).

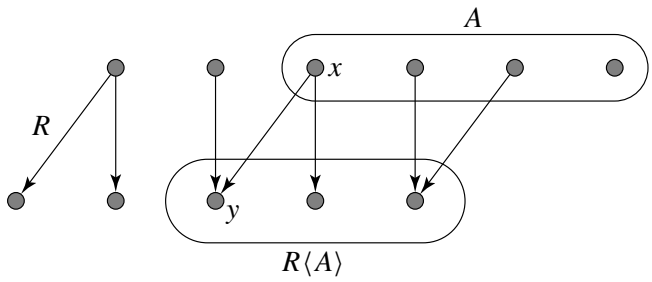
Enfin, si $\mathrm{Prt}F \subset \mathrm{Dom}G$, alors $x \in \mathrm{Dom}F \Rightarrow F(x) \in \mathrm{Dom}G$, de sorte que la deuxième formule du théorème se simplifie en $x \in \mathrm{Dom}(G \circ F) \Leftrightarrow x \in \mathrm{Dom}F$, suivant la règle de logique: si $\mathbf{A} \Rightarrow \mathbf{B}$ est vraie, ($\mathbf{A}$ et $\mathbf{B}$) $\Leftrightarrow \mathbf{A}$ est vraie.

1.3.23. Transformé d'un ensemble par une relation

Si R est une relation et si A est un ensemble, on appelle *transformé* de A par R l'ensemble des objets y qui vérifient la condition: il existe un $x \in A$ tel que $(x, y) \in R$. Nous notons cet ensemble $R\langle A\rangle$. Par définition, on a

(1) $$y \in R\langle A\rangle \Leftrightarrow \exists x(x \in A \text{ et } x\,R\,y).$$

Cette notion est illustrée par la figure ci-dessous. On a toujours $R\langle A\rangle \subset \mathrm{Prt}R$.



Les théorèmes suivants sont démontrés dans [A]. Ils sont d'ailleurs assez évidents, intuitivement.

Théorème 1. Soient R une relation et A, A', B des ensembles. On a

(2) $R\langle\varnothing\rangle = \varnothing$.

(3) $A \subset A' \Rightarrow R\langle A\rangle \subset R\langle A'\rangle$.

(4) $R\langle A\rangle \subset B \Leftrightarrow \forall x \forall y\big((x \in A \text{ et } x\,R\,y) \Rightarrow y \in B\big)$.

(5) $y \in R\langle\{a\}\rangle \Leftrightarrow a\,R\,y$.

(6) $R\langle A\rangle = \bigcup\limits_{x \in A} R\langle\{x\}\rangle$.

Théorème 2. Soient R et S des relations, A un ensemble. On a

$$R \subset S \Leftrightarrow \forall a(\, R\langle\{a\}\rangle \subset S\langle\{a\}\rangle\,).$$

Théorème 3. Soient R et S des relations, A un ensemble. On a

$$(S \circ R)\langle A\rangle = S\langle R\langle A\rangle\rangle.$$

Théorème 4. Soient R une relation et $(A_i)_{i \in I}$ une famille d'ensembles. On a

$$R\langle\bigcup_{i \in I} A_i\rangle = \bigcup_{i \in I} R\langle A_i\rangle.$$

Théorème 5. Soient $(R_i)_{i \in I}$ une famille de relations, A un ensemble. On a

$$\big(\bigcup_{i \in I} R_i\big)\langle A\rangle = \bigcup_{i \in I}\big(R_i\langle A\rangle\big).$$

Théorème 6. Soient E et A des ensembles. On a

$$\mathrm{Id}_E\langle A\rangle = A \cap E;$$
$$A \subset E \Rightarrow \mathrm{Id}_E\langle A\rangle = A.$$

1.3.24. Transformé d'un ensemble par une fonction

Une fonction étant un cas particulier de relation, le transformé $F\langle A\rangle$ d'un ensemble A par une fonction F est défini par la formule 1.3.23(1) (en remplaçant R par F). En tenant compte de la formule de 1.3.10, on obtient, pour une fonction F, la formule :

(1) $$y \in F\langle A\rangle \Leftrightarrow \exists x\big(x \in A \cap \mathrm{Dom}\,F \text{ et } y = F(x)\big).$$

En particulier, comme $A \subset \mathrm{Dom}\,F \Leftrightarrow A \cap \mathrm{Dom}\,F = A$, on peut énoncer :

(2) Si F est une fonction et si $A \subset \mathrm{Dom}\,F$, alors

$$y \in F\langle A\rangle \Leftrightarrow \exists x(x \in A \text{ et } y = F(x))$$

Pour un ensemble à un seul élément, $A = \{a\}$, la formule (1) entraîne :

(3) $$F\langle\{a\}\rangle = \begin{cases} \{F(a)\} & \text{Si } a \in \mathrm{Dom}\,F; \\ \varnothing & \text{Si } a \notin \mathrm{Dom}\,F. \end{cases}$$

Démonstrations : voir [A].

1.4 Ensembles ordonnés

1.4.1. Relations d'ordre

Une relation R dans un ensemble E (1.3.3) est appelée une *relation d'ordre* dans E, si, quels que soient les éléments x, y, z de E, on a

$$(1)\qquad \begin{cases} x\,R\,x; & \text{(réflexivité)} \\ (x\,R\,y \text{ et } y\,R\,z) \Rightarrow x\,R\,z; & \text{(transitivité)} \\ (x\,R\,y \text{ et } y\,R\,x) \Rightarrow x = y. & \text{(antisymétrie)} \end{cases}$$

Ces formules peuvent s'écrire bien entendu de la manière suivante :

$$\begin{array}{l} (x, x) \in R; \\ ((x, y) \in R \text{ et } (y, z) \in R) \Rightarrow (x, z) \in R; \\ ((x, y) \in R \text{ et } (y, x) \in R) \Rightarrow x = y. \end{array}$$

Ces trois propriétés caractéristiques d'une relation d'ordre R peuvent s'exprimer par ailleurs plus succinctement, sans parler d'éléments x, y, z de E, par les formules suivantes :

$$\begin{array}{ll} \mathrm{Id}_E \subset R; & \text{(réflexivité)} \\ R \circ R \subset R; & \text{(transitivité)} \\ R \cap R^{-1} \subset \mathrm{Id}_E. & \text{(antisymétrie)} \end{array}$$

Notations. Si R est une relation d'ordre dans un ensemble E, on utilise souvent la notation

$$x \leq_R y$$

au lieu de $x\,R\,y$, donc de $(x, y) \in R$, et l'on dit que x *est inférieur à* y *suivant* R. On omet souvent l'indice $_R$ de $\leq_R$ (on écrit simplement $x \leq y$ et l'on dit que x est inférieur à y) lorsqu'il n'y a pas de confusion possible quant à la relation d'ordre R dont il est question. Avec cette notation, les propriétés caractéristiques d'une relation d'ordre (réflexivité, transitivité, antisymétrie) prennent la forme bien familière :

$$(2)\qquad \begin{cases} x \leq_R x; & \text{(réflexivité)} \\ (x \leq_R y \text{ et } y \leq_R z) \Rightarrow x \leq_R z; & \text{(transitivité)} \\ (x \leq_R y \text{ et } y \leq_R x) \Rightarrow x = y. & \text{(antisymétrie)} \end{cases}$$

La notation $x <_R y$ se lit x *est strictement inférieur à* y suivant R. Elle est définie par

$$(3)\qquad x <_R y \;\Leftrightarrow\; (x \leq_R y \text{ et } x \neq y).$$

La proposition $x <_R y$ équivaut donc à $(x, y) \in R$ et $x \neq y$. On en déduit la formule

$$(4)\qquad x \leq_R y \;\Leftrightarrow\; (x <_R y \text{ ou } x = y).$$

Cette formule se déduit par logique propositionnelle de la définition (3) et de la formule $x = y \Rightarrow x \leq_R y$ qui découle de la réflexivité de R [A].

Ensembles ordonnés. On appelle *ensemble ordonné* un ensemble E *muni d'une relation d'ordre* R (dans E). Ce que signifie formellement l'expression « muni de », dans cette définition, c'est qu'un ensemble ordonné est en fait un *couple* (E, R) où E est un ensemble et R est une relation d'ordre dans E.

Lorsque, dans un certain ensemble E, c'est toujours la même relation d'ordre R qui est prise en considération, on parle simplement de « l'ensemble ordonné E », au lieu de l'ensemble ordonné (E, R). La relation d'ordre R, connue, est sous-entendue.

1.4.2. Exemple : relation d'ordre totale

Soit R l'ensemble des couples (x, y) d'entiers naturels ($(x, y) \in \mathbb{N} \times \mathbb{N}$) tels que $x \leq y$, où l'inégalité $x \leq y$ a son sens usuel[1]. Autrement dit, soit

$$R = \{(x, y) \mid (x, y) \in \mathbb{N} \times \mathbb{N} \text{ et } x \leq y\}.$$

Il est clair que R est une relation d'ordre dans $\mathbb{N}$, car si x et y sont des éléments quelconques de $\mathbb{N}$, on a

$$(x, y) \in R \Leftrightarrow x \leq y \quad \text{(au sens usuel)},$$

et les propriétés de réflexivité, transitivité et antisymétrie 1.4.1 (1) sont bien connues dans ce cas. Si l'on écrit $x \leq_R y$ pour $(x, y) \in R$, le symbole $\leq_R$ a le sens de l'inégalité usuelle entre nombres entiers. Muni de cette relation d'ordre, l'ensemble $\mathbb{N}$ est *l'ensemble ordonné* $\mathbb{N}$ au sens usuel.

En plus des propriétés de réflexivité, transitivité et antisymétrie, cette relation d'ordre R dans l'ensemble $E = \mathbb{N}$ jouit de la propriété suivante :

$$(1) \qquad \forall x \in E : \forall y \in E : \ (x \leq_R y \text{ ou } y \leq_R x).$$

Définition. Lorsqu'une relation d'ordre R dans un ensemble E vérifie la proposition (1), on dit que R est une relation d'ordre *totale* dans E, et, muni de cette relation d'ordre, l'ensemble E est appelé un ensemble *totalement ordonné*.

L'ensemble ordonné $\mathbb{N}$ est donc un ensemble totalement ordonné. Il en va de même de l'ensemble ordonné $\mathbb{R}$ (au sens usuel), à savoir l'ensemble $\mathbb{R}$ muni de la relation R' qui est l'ensemble des couples $(x, y) \in \mathbb{R} \times \mathbb{R}$ tels que $x \leq y$ au sens usuel.

1.4.3. Exemple : relation d'ordre non totale

Soient $X = \{a, b, c\}$ un ensemble à trois éléments distincts et $E = \mathcal{P}(X)$ l'ensemble des parties (ou sous-ensembles) de X. L'ensemble E possède huit éléments, à savoir les huit ensembles notés dans la figure 1.1 à côté des sommets de ce diagramme. Soit R l'ensemble des couples (x, y) d'éléments de E tels que $x \subset y$. Par exemple, le couple $(x, y) = (\{b\}, \{a, b\})$ appartient à R puisque $\{b\} \subset \{a, b\}$. L'appartenance de ce couple à R est représentée dans la figure 1.1 par la ligne montante reliant le sommet $\{b\}$ au sommet $\{a, b\}$ (il est sous-entendu que toutes les lignes de ce diagramme vont de bas en haut). De façon générale, par définition de R, pour deux éléments x, y quelconques de E, on a

$$x \, R \, y \Leftrightarrow x \subset y.$$

Il en découle immédiatement que R est une relation d'ordre dans E. Elle est réflexive, transitive et antisymétrique en vertu des formules générales de théorie des ensembles :

$$\begin{aligned} &x \subset x; \\ &(x \subset y \text{ et } y \subset z) \Rightarrow x \subset z; \\ &(x \subset y \text{ et } y \subset x) \Rightarrow x = y. \end{aligned}$$

Muni de cette relation d'ordre, l'ensemble $E = \mathcal{P}(X)$ est dit *ordonné par inclusion*.

Cette relation d'ordre R dans E n'est pas totale. Elle ne vérifie pas la proposition 1.4.2 (1). Il existe des éléments x, y de E, par exemple $x = \{a\}$ et $y = \{b, c\}$, pour lesquels on a

$$\text{non}(x \, R \, y \text{ ou } y \, R \, x).$$

---

[1] On peut rappeler ce sens en disant que pour deux entiers naturels x, y on a $x \leq y$ (par définition) si et seulement s'il existe un $n \in \mathbb{N}$ tel que $x + n = y$.

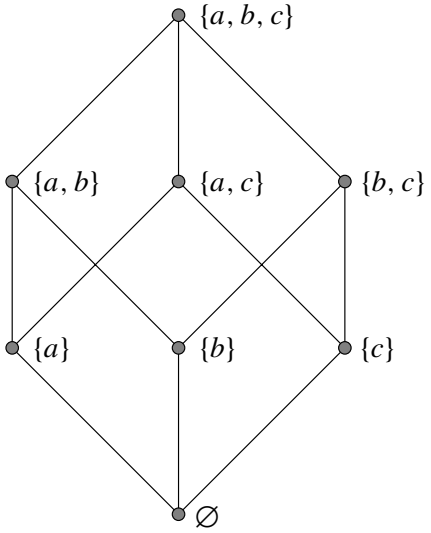


Figure 1.1
L'ensemble $\mathcal{P}(\{a, b, c\})$ ordonné par inclusion.

Dans le diagramme de la figure 1.1 tous les couples (x, y) appartenant à R ne sont pas marqués par un trait allant directement du sommet x au sommet y. Par exemple, bien que l'on ait $\varnothing \leq_R \{a, c\}$, il n'y a pas de trait allant directement du sommet $\varnothing$ au sommet $\{a, c\}$. Mais ce trait est sous-entendu. Il découle par transitivité des traits allant de $\varnothing$ à $\{a\}$ et de $\{a\}$ à $\{a, c\}$, ou de ceux qui vont de $\varnothing$ à $\{c\}$ et de $\{c\}$ à $\{a, c\}$.

Une relation d'ordre semblable peut être définie dans tout ensemble E qui est un ensemble d'ensembles. Autrement dit tout ensemble d'ensembles E peut être ordonné par inclusion, c'est-à-dire muni de la relation d'ordre

$$R = \{(x, y) \mid (x, y) \in E \times E \text{ et } x \subset y\}. \tag{1}$$

Cette relation peut être totale ou non, cela dépend de l'ensemble E considéré. Par exemple si, au lieu de l'ensemble $E = \mathcal{P}(\{a, b, c\})$ de l'exemple ci-dessus, on se limite à l'ensemble

$$E' = \{\varnothing, \{a\}, \{a, b\}, \{a, b, c\}\},$$

alors la relation d'ordre par inclusion dans cet ensemble est totale. Son diagramme (figure 1.2) est « linéaire », tout comme celui de l'ensemble ordonné $\mathbb{N}$ (figure 1.3). Un diagramme linéaire est caractéristique d'une relation d'ordre totale et certains auteurs parlent de relations d'ordre linéaires au lieu de relations d'ordre totales.

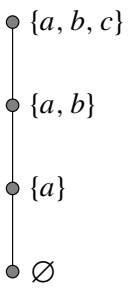


Figure 1.2

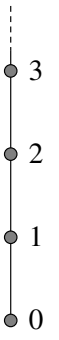


Figure 1.3

Une bizarrerie de la terminologie. Le contraire de l'adjectif « total » étant « partiel », il serait naturel d'appeler *relation d'ordre partielle* une relation d'ordre qui n'est pas totale. Mais, sous l'influence des auteurs anglo-saxons, le terme de *relation d'ordre partielle* est utilisé très souvent dans le sens de relation d'ordre *quelconque*, totale ou non. Donc le terme

générique pour *relation d'ordre* est devenu *relation d'ordre partielle*, avec la conséquence singulière qu'une relation d'ordre totale est un cas particulier de relation d'ordre partielle et devrait donc s'appeler une *relation d'ordre partielle totale* ...

1.4.4. Plus petit élément d'un ensemble ordonné

Si R est une relation d'ordre dans l'ensemble E, il existe au plus un élément x de E tel que l'on ait

(1) $$\forall y \in E : x \leq_R y.$$

En effet, si x et x' sont deux éléments de E qui vérifient (1), ce par quoi nous entendons que x vérifie (1) et que x' vérifie la proposition correspondante $\forall y \in E : x' \leq_R y$, alors, puisque x et x' sont éléments de E, on a $x \leq_R x'$ et $x' \leq_R x$, donc $x = x'$.

Donc s'il existe un élément x de E vérifiant (1), cet élément est unique et il est appelé *le plus petit élément de* E pour la relation d'ordre R.

Exemple 1. L'ensemble ordonné $\mathbb{N}$ possède un plus petit élément x, à savoir $x = 0$.

Exemple 2. L'ensemble ordonné $\mathbb{R}$ n'a pas de plus petit élément.

Exemple 3. L'ensemble ordonné représenté dans la figure 1.1, à savoir l'ensemble $E = \mathcal{P}(\{a, b, c\})$, ordonné par inclusion, possède un plus petit élément x, à savoir l'ensemble $x = \varnothing$.

Exemple 4. L'ensemble des parties *non vides* d'un ensemble $\{a, b, c\}$ à trois éléments distincts, ordonné par inclusion, est représenté dans la figure 1.4. Cet ensemble ne possède pas de plus petit élément.

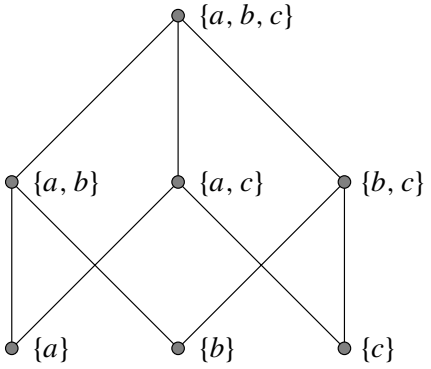


Figure 1.4
L'ensemble des parties non vides d'un ensemble $\{a, b, c\}$, ordonné par inclusion.

1.4.5. L'ensemble S_x des minorants stricts d'un élément x

Lorsqu'il est question d'une relation d'ordre R dans un ensemble E, nous utilisons toujours la notation S_x, ou $S(x)$, pour désigner l'ensemble des éléments de E qui sont strictement inférieurs à un élément x de E suivant la relation R considérée. Nous posons donc, pour tout $x \in E$,

(1) $$S_x = \{z \mid z <_R x\}.$$

Par définition, on a donc pour tout $x \in E$:

(2) $$S_x \subset E;$$

(3) $$z \in S_x \Leftrightarrow z <_R x;$$

(4) $$x \notin S_x.$$

Remarque. Il est clair que la notation S_x, ou $S(x)$, est incomplète car elle ne comporte pas d'indication de la relation d'ordre R considérée. Il faudrait écrire $S_R(x)$ par exemple. Mais en général la relation d'ordre R est connue et la notation S_x suffit.

1.4.6. Éléments minimaux

Soient E un ensemble, R une relation d'ordre dans E et A une partie de E ($A \subset E$). On dit qu'un élément x de A est un élément *minimal* de A suivant R s'il n'existe pas d'élément de A strictement inférieur à x suivant R, autrement dit si l'ensemble $S_x \cap A$ est vide. Par définition, si $A \subset E$, on a les équivalences :

$$\begin{aligned} &x \text{ est un élément minimal de } A \text{ suivant } R \\ &\Leftrightarrow\ x \in A \text{ et } S_x \cap A = \varnothing \\ &\Leftrightarrow\ x \in A \text{ et non}\exists z(z \in A \text{ et } z <_R x). \end{aligned} \tag{1}$$

Exemple 1. La figure 1.5 représente à nouveau l'ensemble ordonné E présenté au §1.4.3 (figure 1.1), en désignant ses éléments par $a_0, \ldots, a_7$ pour la commodité. On considère le sous-ensemble $A = \{a_1, a_4, a_5, a_6, a_7\}$ de E, dont les éléments sont mis en évidence par les sommets noirs de la figure. Cet ensemble A possède deux éléments minimaux (pour la relation d'ordre considérée dans E), à savoir les éléments a_1 et a_6. En effet, pour ces deux éléments a_i de A, et seulement pour ces deux, il n'existe pas d'élément $a_j \in A$ tel que $a_j < a_i$. Il revient au même de dire que

$$S_{a_1} \cap A = \varnothing \text{ et } S_{a_6} \cap A = \varnothing.$$

On a en effet $S_{a_1} = \{a_0\}$ et $S_{a_6} = \{a_0, a_2, a_3\}$, et ces ensembles n'ont pas d'élément appartenant à A. Par contre $S_{a_5} = \{a_0, a_1, a_3\}$ et l'on a $S_{a_5} \cap A = \{a_1\} \neq \varnothing$, donc a_5 n'est pas un élément minimal de A.

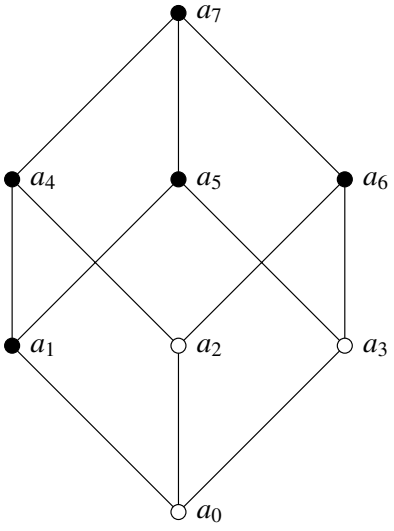


Figure 1.5

Ensemble ordonné $E = \{a_0, \ldots, a_7\}$.
Sous-ensemble $A = \{a_1, a_4, a_5, a_6, a_7\}$.
Éléments minimaux de A : a_1 et a_6.

Exemple 2. La définition d'un élément minimal x d'un sous-ensemble A d'un ensemble ordonné E, et notamment la formule (1), peuvent s'appliquer au cas particulier du sous-ensemble $A = E$. Un élément minimal de A est alors un *élément minimal de* E et la formule (1) se simplifie de la manière suivante, compte tenu de ce que l'on a $S_x \cap E = S_x$ d'après 1.4.5 (2) pour tout $x \in E$:

$$\begin{aligned} &x \text{ est un élément minimal de } E \text{ suivant } R \\ &\Leftrightarrow\ x \in E \text{ et } S_x = \varnothing \\ &\Leftrightarrow\ x \in E \text{ et non}\exists z(z <_R x). \end{aligned} \tag{2}$$

Dans l'exemple de la figure 1.5, l'unique élément minimal de E est a_0. Par ailleurs, a_0 est le plus petit élément de E (1.4.4) et cet exemple illustre une propriété générale qui s'énonce ainsi : Lorsqu'un ensemble ordonné E possède un plus petit élément, alors cet élément x est *l'unique élément minimal* de E. Les exemples des figures 1.2 et 1.3 illustrent aussi cette assertion (démontrée dans [A]).

Exemple 3. L'ensemble ordonné E représenté dans la figure 1.4 possède trois éléments minimaux : $\{a\}$, $\{b\}$, $\{c\}$. (On suppose que a, b, c sont trois objets distincts.)

Ce dernier exemple montre bien qu'il ne faut pas confondre les notions de *plus petit élément* (1.4.4) et d'*élément minimal* d'un ensemble ordonné E. L'ensemble ordonné de la figure 1.4 ne possède pas de plus petit élément. Nous avons vu qu'un ensemble ordonné possède *au plus un* plus petit élément. Le présent exemple montre qu'un ensemble ordonné peut avoir plusieurs éléments minimaux. Il peut en avoir une infinité. Par exemple l'ensemble des parties non vides d'un ensemble X quelconque, ordonné par inclusion, a pour éléments minimaux, comme dans la figure 1.4, tous les ensembles à un seul élément $\{x\}$ tels que $x \in X$. Il y en a une infinité si X est un ensemble infini.

1.4.7. Relations d'ordre noethériennes

L'ensemble ordonné E représenté dans la figure 1.5 possède la propriété évidente suivante : toute partie non vide A de E possède un élément minimal — ce par quoi l'on entend naturellement qu'elle en possède *au moins un*. Les ensembles ordonnés qui possèdent cette propriété ont une grande importance en informatique, car c'est elle qui permet fondamentalement l'emploi de la « récursivité » en informatique, c'est-à-dire qui permet la définition « récursive » de fonctions. Ces mots entre guillemets appartiennent encore au « franglais ». Plus loin, nous parlerons plutôt de « récurrence ».

Définition 1. On dit qu'une relation d'ordre R dans un ensemble E est une relation d'ordre *noethérienne* dans E si toute partie non vide A de E possède un élément minimal (au moins un) pour R. Muni de cette relation d'ordre, l'ensemble E est appelé un ensemble ordonné *noethérien*.

À la place de l'adjectif « noethérien », de nombreux auteurs parlent de relations d'ordre *bien fondées* (en anglais *well-founded*) et d'ensembles ordonnés *bien fondés*. Mais Noether est le nom d'une *mathématicienne*[2] et les femmes mathématiciennes de renom sont assez rares pour qu'on leur fasse une petite révérence.

Théorème 1. Pour qu'une relation d'ordre R dans un ensemble E soit une relation d'ordre noethérienne dans E, il faut et il suffit qu'elle satisfasse à la condition suivante : il n'existe pas de suite $(x_n)_{n\in\mathbb{N}}$ d'éléments de E strictement décroissante suivant R, c'est-à-dire telle que l'on ait $x_{n+1} <_R x_n$ pour tout $n \in \mathbb{N}$.

Ce théorème fournit une manière facile (en général) de reconnaître une relation d'ordre noethérienne R dans un ensemble E. La condition du théorème peut en effet s'interpréter de la manière suivante : dans le diagramme de l'ensemble ordonné E, partant de n'importe quel élément (sommet) x_0, il n'est pas possible de descendre indéfiniment suivant les arcs du diagramme, c'est-à-dire de parcourir une suite infinie de sommets $x_0 > x_1 > x_2 > \dots$

Cette propriété est immédiatement visible dans les exemples des figures 1.1, 1.2, 1.3, 1.4. En particulier, l'ensemble ordonné $\mathbb{N}$ (figure 1.3) est noethérien. Il vérifie même une condition plus forte que celle qui figure dans la définition d'un tel ordre. Toute partie non vide A de $\mathbb{N}$ possède un *plus petit élément*, c'est-à-dire un élément a tel que $a \leq x$ pour tout

[2] Amalie Emmy Noether (1882–1935), mathématicienne allemande.

$x \in A$, et cet élément a est l'unique élément minimal de A. Par contre, l'ensemble ordonné $\mathbb{R}$ n'est pas noethérien, puisqu'il existe évidemment une suite infinie $x_0 > x_1 > x_2 > \ldots$ strictement décroissante d'éléments de $\mathbb{R}$.

On trouvera en annexe [A] la démonstration informelle (facile) du théorème 1. Pour le lecteur intéressé, des indications complémentaires sont données sur la manière de formaliser cette démonstration, car elle fait intervenir des déductions mathématiques fondamentales.

1.5 Récurrence

1.5.1. Introduction

Le mot « récurrence » est utilisé en mathématique à propos de deux choses. Il sert premièrement à désigner certaines formes de démonstrations, dites démonstrations « par récurrence ». On emploie aussi dans ce sens le mot « induction » en parlant de démonstrations « par induction ». Deuxièmement, le mot « récurrence » sert à désigner une certaine manière de définir des fonctions. On parle de la définition de fonctions « par récurrence ». Le mot anglais qui correspond à « récurrence » dans ce sens est « recursion », substantif auquel correspond l'adjectif « recursive ».

En informatique, la définition de fonctions par récurrence est une technique de programmation très importante, que permettent tous les langages de programmation évolués. Elle constitue même l'une des méthodes de programmation *essentielles* des langages de programmation dits « fonctionnels ».

Le concept de récurrence fait donc partie des bases de la programmation et son étude devrait faire partie de la formation mathématique d'un informaticien universitaire. La présente section n'est pas un cours complet sur ce sujet. Le lecteur a sans doute déjà pratiqué la programmation « récursive » (adjectif importé de l'anglais) et rencontré en mathématique des démonstrations par récurrence. Nous ne voulons ici que *formuler de manière précise* les quelques schémas de déduction et théorèmes principaux qui sont la base mathématique de ces deux démarches, que nous utiliserons souvent dans ce cours.

La définition d'une fonction F par récurrence consiste à poser pour F des équations d'une forme particulière (équations de récurrence), dans lesquelles F peut figurer des deux côtés d'une égalité, ce qui fait dire souvent que « F est utilisée dans sa propre définition ». Du point de vue mathématique, ce sont simplement des équations en F qui possèdent une solution F et une seule et peuvent donc être prises comme définissant une fonction F unique et bien déterminée. Par exemple, on peut démontrer qu'il existe une fonction $F : \mathbb{N} \to \mathbb{N}$ et une seule telle que l'on ait

$$(1) \qquad \begin{cases} F(0) = 1; \\ \forall n \in \mathbb{N} : F(n+1) = (n+1) \cdot F(n). \end{cases}$$

Il s'agit de la fonction factorielle $F = (\lambda n \in \mathbb{N} : n!)$ qui peut être définie par (1). C'est l'existence et l'unicité d'une fonction $F : \mathbb{N} \to \mathbb{N}$ vérifiant (1) qui permettent de prendre ces équations comme *définition* de la fonction factorielle. On parle d'une définition de fonction par récurrence — ou d'une définition « récursive » (franglais) de la fonction factorielle. L'existence et l'unicité de F est un *théorème* — généralement admis sans démonstration dans l'enseignement traditionnel de la programmation, et il en va ainsi de toutes les définitions récursives de fonctions.

L'*unicité* d'une fonction définie par récurrence est en général facile à démontrer. Dans l'exemple des équations (1), il s'agit de démontrer qu'il existe *au plus* une fonction F

vérifiant ces équations, c'est-à-dire que si F et F' sont deux fonctions de domaine $\mathbb{N}$ vérifiant ces équations, alors elles sont égales, et pour cela, on démontre par récurrence sur $n \in \mathbb{N}$ (1.5.2) que l'on a $\forall n \in \mathbb{N} : F(n) = F'(n)$. Voir 1.5.3.

L'existence d'une solution est beaucoup moins facile à prouver, même si elle paraît intuitivement évidente. Il en va ainsi de la plupart des définitions de fonctions par récurrence. La preuve d'existence des solutions sort du cadre de ce cours. Cette difficulté est d'ailleurs la raison pour laquelle on ne fait cette démonstration qu'une fois pour toutes, c'est-à-dire pour certaines formes d'équations de récurrence suffisamment générales, auxquelles se ramènent plus ou moins facilement tous les cas particuliers. C'est la principale de ces formes générales que nous présenterons plus loin (1.5.7). Nous allons parler d'abord de démonstrations par récurrence.

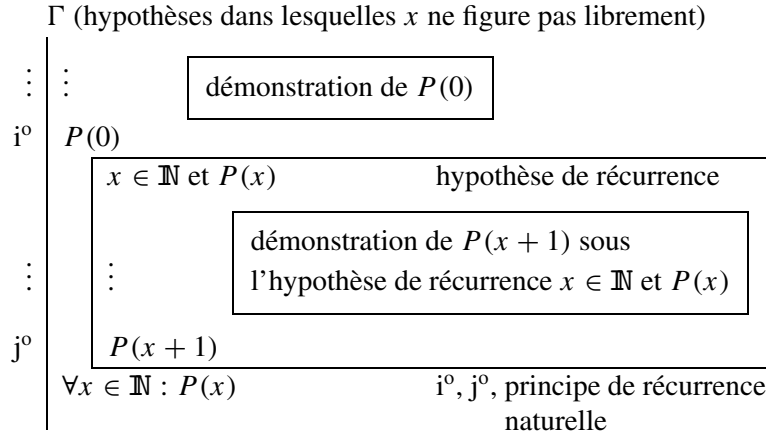


Figure 1.6
Plan d'une démonstration par récurrence naturelle sur x dans $\mathbb{N}$.

1.5.2. Démonstrations par récurrence naturelle dans $\mathbb{N}$

Chacun sait que pour démontrer une proposition de la forme

$$\forall x \in \mathbb{N} : P(x),$$

autrement dit pour démontrer que tout entier naturel x possède une certaine propriété exprimée par une assertion $P(x)$, il suffit de démontrer premièrement que l'élément 0 de $\mathbb{N}$ possède cette propriété, autrement dit que la proposition

(1) $$P(0)$$

est vraie, et deuxièmement, de démontrer que si un entier naturel x quelconque possède la propriété en question, alors il en va de même de son successeur, l'entier $x+1$. Formellement, cela signifie démontrer que la proposition

(2) $$\forall x \in \mathbb{N} : \quad P(x) \Rightarrow P(x+1)$$

est vraie ([3]). Lorsqu'on a démontré (1) et (2), on en conclut que la proposition $\forall x \in \mathbb{N} : P(x)$ est vraie, et cette déduction est appelée un raisonnement par récurrence (naturelle) sur x dans $\mathbb{N}$. Le fait que cette déduction soit permise peut être considéré comme une

[3] On rappelle que la notation $\forall x \in \mathbb{N} : P(x)$ est une abréviation de $\forall x(x \in \mathbb{N} \Rightarrow P(x))$. Nous écrivons $\forall x \in \mathbb{N} : P(x) \Rightarrow P(x+1)$ pour $\forall x \in \mathbb{N} : (P(x) \Rightarrow P(x+1))$, et cela signifie donc $\forall x(x \in \mathbb{N} \Rightarrow (P(x) \Rightarrow P(x+1)))$.

propriété fondamentale des entiers naturels. Nous l'appelons le *principe de démonstration par récurrence naturelle* dans $\mathbb{N}$.

En général, un tel raisonnement s'effectue dans un contexte logique qui ne comporte aucune hypothèse sur x (la variable de récurrence). C'est pourquoi, dans la figure 1.6 qui donne le plan général d'une démonstration par récurrence naturelle sur x dans $\mathbb{N}$, on suppose que la variable x ne figure pas librement dans les hypothèses du contexte (Γ).

1.5.3. Exemple

Comme exemple très simple de démonstration par récurrence naturelle dans $\mathbb{N}$, nous allons démontrer ici *l'unicité* d'une fonction $F : \mathbb{N} \to \mathbb{N}$ telle que

$$(1) \qquad \begin{cases} F(0) = 1; \\ \forall x \in \mathbb{N} : F(x+1) = (x+1) \cdot F(x) \end{cases}$$

(cf 1.5.1), à savoir l'assertion: il existe au plus une fonction $F : \mathbb{N} \to \mathbb{N}$ vérifiant les équations (1). Cela signifie, comme chacun sait, que si deux fonctions $F, F' : \mathbb{N} \to \mathbb{N}$ vérifient (1), alors $F = F'$. Pour démontrer cela, on suppose donc que F et F' sont deux applications de $\mathbb{N}$ dans $\mathbb{N}$ et l'on fait l'hypothèse que F vérifie (1) et que F' vérifie les équations correspondantes, où l'on remplace F par F'. Sous ces hypothèses, on démontre que $F = F'$, c'est-à-dire (1.3.13) que

$$(2) \qquad \forall x \in \mathbb{N} : \underbrace{F(x) = F'(x)}_{P(x)}.$$

La proposition (2) se démontre par récurrence sur $x \in \mathbb{N}$ suivant le plan de la figure 1.6. Les hypothèses Γ sont celles que nous venons de mentionner à propos de F et F'. Dans ces hypothèses, la variable x ne figure pas librement. La proposition $P(0)$ (figure 1.6) est dans notre cas $F(0) = F'(0)$. Elle découle immédiatement des hypothèses $F(0) = 1$, $F'(0) = 1$. Faisons l'hypothèse de récurrence

$$(3) \qquad x \in \mathbb{N} \text{ et } \underbrace{F(x) = F'(x)}_{P(x)}.$$

On peut alors écrire

$$\begin{aligned} F(x+1) &= (x+1) \cdot F(x) && \text{équations (1)} \\ &= (x+1) \cdot F'(x) && \text{hypothèse de récurrence (3)} \\ &= F'(x+1) && \text{équations (1) avec } F'. \end{aligned}$$

On a donc démontré $P(x+1)$, c'est-à-dire $F(x+1) = F'(x+1)$, sous l'hypothèse de récurrence $x \in \mathbb{N}$ et $P(x)$. La démonstration de (2) suivant le plan de la figure 1.6 est achevée.

1.5.4. Théorème

Soient A un ensemble, α une application de A dans lui-même ($\alpha : A \to A$) et a un élément de A. Il existe une suite $(x_n)_{n \in \mathbb{N}}$ d'éléments de A et une seule telle que $x_0 = a$ et telle que

$$\forall n \in \mathbb{N} : \; x_{n+1} = \alpha(x_n).$$

Ce théorème est le premier théorème d'existence et d'unicité d'une fonction satisfaisant à certaines équations de récurrence. Il s'agit ici d'une fonction $x : \mathbb{N} \to A$, appelée « suite d'éléments de A » (1.3.18). C'est non seulement le premier théorème de ce genre que

nous énonçons dans ce cours, mais surtout le premier dans la construction logique de toute la « théorie de la récurrence » dont nous présentons dans cette section quelques éléments importants.

La démonstration sort du cadre de ce cours. Disons seulement qu'elle utilise le principe de récurrence naturelle (1.5.2) et que c'est la raison pour laquelle ce théorème est placé ici. Mais il est déjà utilisé dans [A], dans la démonstration du théorème de 1.4.7 caractérisant les relations d'ordre noethériennes.

1.5.5. Démonstrations par récurrence noethérienne

Nous allons présenter dans ce paragraphe une méthode très importante pour démontrer une proposition de la forme

(1) $$\forall x \in E : P(x)$$

dans un contexte où E est un ensemble muni d'une relation d'ordre noethérienne R. Nous rappelons que dans un tel contexte nous désignons par S_x, pour tout $x \in E$, l'ensemble des éléments z de E tels que $z <_R x$ (1.4.5).

Le principe de récurrence noethérien. Démontrer (1) par « récurrence noethérienne dans E » consiste à démontrer, pour un élément x quelconque de E, que l'on a

(a) $S_x = \varnothing \Rightarrow P(x)$.

Autrement dit : tout élément minimal de E (1.4.6) vérifie P.

(b) $\big[S_x \neq \varnothing \text{ et } \forall z \in S_x : P(z)\big] \Rightarrow P(x)$.

Autrement dit :
si x n'est pas un élément minimal de E et si tous les éléments de E strictement inférieurs à x vérifient P, alors x vérifie P.

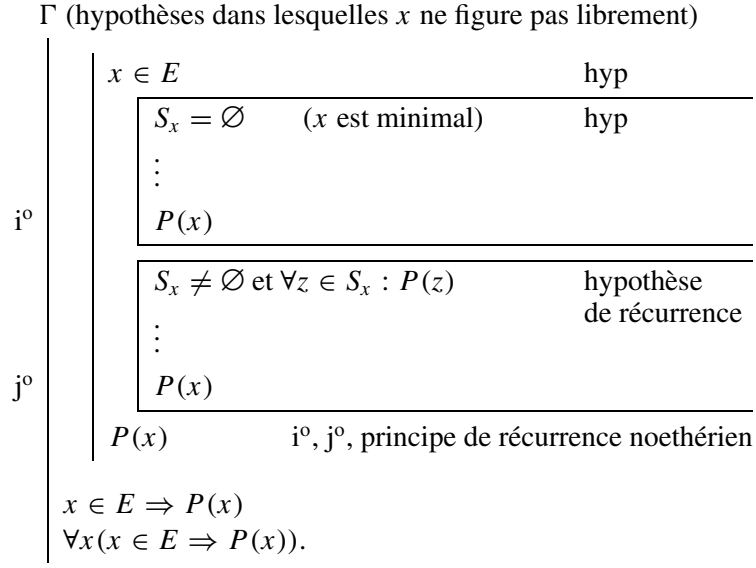


Figure 1.7

Plan d'une démonstration par récurrence noethérienne dans un ensemble ordonné noethérien E. Les deux lignes finales sont omises en général.

Pratiquement, une telle démonstration se déroule suivant le plan donné dans la figure 1.7. Les hypothèses du contexte dans lequel on démontre de cette manière la proposition (1), désignées par Γ dans la figure, ne doivent évidemment pas contenir d'occurrence libre de la variable de récurrence x pour qu'on puisse effectuer la généralisation finale sur x dans la démonstration (dernière ligne).

Le principe de récurrence noethérien, invoqué à la ligne j + 1 de la figure 1.7 est en fait un schéma de déduction, au sens de la logique. Il est démontré dans [A].

1.5.6. Cas particulier : démonstrations par récurrence noethérienne dans $\mathbb{N}$

Comme nous l'avons vu au §1.4.7, la relation d'ordre usuelle dans $\mathbb{N}$ est noethérienne. On peut donc utiliser la méthode de démonstration de la figure 1.7 en l'adaptant à ce cas particulier. Le plan de démonstration par récurrence noethérienne dans $\mathbb{N}$ qui en résulte est donné dans la figure 1.8. Dans ce cas particulier, $S_x = \varnothing$ équivaut à $x = 0$. La proposition $\forall z \in S_x : P(z)$ peut s'écrire $\forall z < x : P(z)$.

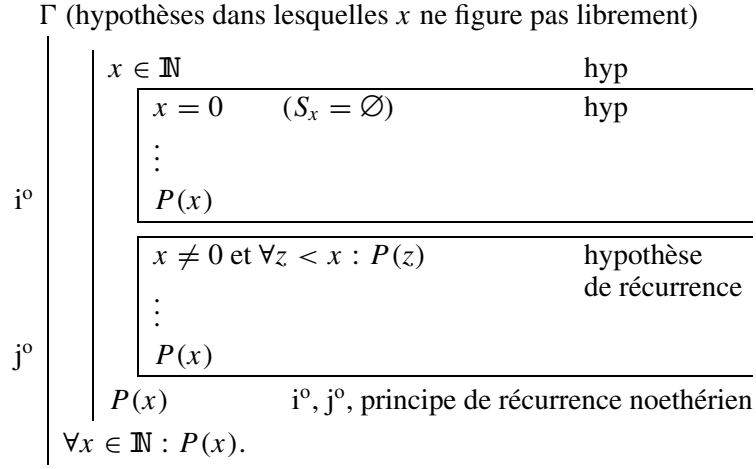


Figure 1.8

Plan d'une démonstration par récurrence noethérienne dans $\mathbb{N}$, conformément au plan général de la figure 1.7.

1.5.7. Définitions de fonctions par récurrence noethérienne

Nous allons énoncer dans ce paragraphe un schéma de déduction très général permettant de justifier toutes les définitions récursives de fonctions auxquelles nous aurons affaire dans la suite, et toutes celles que l'on rencontre plus généralement en informatique. Nous appellerons ce schéma le *principe de définition par récurrence noethérienne*. Il exprime que dans un contexte où E est un ensemble muni d'une relation d'ordre noethérienne R, il existe une fonction F et une seule, de domaine E, satisfaisant à des équations de récurrence d'une certaine forme. Dans la description de cette forme (voir encadré ci-dessous) interviennent deux termes indéterminés $\mathbf{T}_1$ et $\mathbf{T}_2$. La notation $F_{|S_x}$, pour un $x \in E$, désigne la restriction de la fonction F à l'ensemble S_x (1.3.21). La notation $(F_{|S_x}|F)\mathbf{T}_2$ est celle d'une substitution (cours de logique élémentaire). Elle désigne le terme obtenu en remplaçant chaque occurrence libre de la variable F dans $\mathbf{T}_2$ par le terme $F_{|S_x}$.

Principe de définition par récurrence noethérienne. Dans un contexte où E désigne un ensemble et R une relation d'ordre noethérienne dans E et où, pour tout $x \in E$, S_x désigne l'ensemble des éléments z de E tels que $z <_R x$ (§1.4.5), la proposition

(1)
$$\boxed{\begin{array}{l}\text{Il existe une fonction } F \text{ et une seule telle que} \\ \text{Dom}\, F = E \text{ et} \\ \forall x \in E : \quad F(x) = \begin{cases} \mathbf{T}_1 & \text{si } S_x = \varnothing; \\ (F_{|S_x}|F)\mathbf{T}_2 & \text{si } S_x \neq \varnothing \end{cases}\end{array}}$$

est vraie si $\mathbf{T}_1$ et $\mathbf{T}_2$ sont des termes satisfaisant aux conditions suivantes :

- la variable F ne figure pas librement dans $\mathbf{T}_1$
- le terme $F_{|S_x}$ est librement substituable à F dans $\mathbf{T}_2$.

L'interprétation de ce principe est discutée au §1.5.9, après l'exemple qui suit.

1.5.8. Exemple

La suite des *nombres de Fibonacci (non nuls)*

$$1, 2, 3, 5, 8, 13, 21, 34, \ldots$$

est caractérisée par le fait que les deux premiers de ces nombres sont 1 et 2, et que chacun des nombres suivants est la somme des deux nombres qui le précèdent. Formellement, on définit cette suite comme étant *la fonction* $F : \mathbb{N} \to \mathbb{N}$ telle que, pour tout $x \in \mathbb{N}$:

(1)
$$F(x) = \begin{cases} 1 & \text{si } x = 0; \\ 2 & \text{si } x = 1; \\ F(x-2) + F(x-1) & \text{sinon.} \end{cases}$$

En acceptant cette définition, on admet qu'il existe une fonction $F : \mathbb{N} \to \mathbb{N}$ et une seule vérifiant (1) pour tout $x \in \mathbb{N}$. Nous allons montrer que cela découle du principe de définition par récurrence noethérienne. L'application de ce principe à un cas particulier nécessite que l'on choisisse une relation d'ordre noethérienne appropriée au cas particulier. Dans cet exemple, ce n'est pas la relation d'ordre usuelle dans $\mathbb{N}$ qu'il faut considérer (figure 1.3), mais celle qui est dépeinte dans la figure 1.9 sous le nom de $R_{\text{Fibonacci}}$, que nous abrégerons R_F.

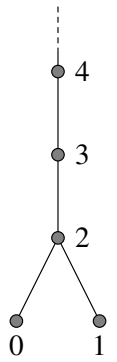


Figure 1.9
Ordre noethérien $R_{\text{Fibonacci}}$ dans $\mathbb{N}$.

Cette relation R_F dans $\mathbb{N}$ est l'ensemble des couples (x, y) d'éléments de $\mathbb{N}$ qui appartiennent à la relation d'ordre usuelle mais qui ne sont pas le couple $(0, 1)$. Autrement dit, pour deux éléments x, y quelconques de $\mathbb{N}$, on a

$$(x, y) \in R_F \Leftrightarrow (x \leq y \text{ et } (x, y) \neq (0, 1)),$$

formule dans laquelle le signe $\leq$ est pris au sens de la relation d'ordre usuelle. Cette formule permet de montrer facilement que R_F est bien une relation d'ordre dans $\mathbb{N}$ et elle est évidemment noethérienne selon le critère (théorème) du §1.4.7. Muni de la relation d'ordre R_F l'ensemble $\mathbb{N}$ ne possède pas de plus petit élément, mais il possède deux éléments minimaux (0 et 1). En désignant par S_x (pour tout $x \in \mathbb{N}$) l'ensemble des éléments z de $\mathbb{N}$ tels que $z <_{R_F} x$, on a $S_0 = S_1 = \varnothing$ et $S_x \neq \varnothing$ pour tout $x \notin \{0, 1\}$. L'équation (1) peut donc s'écrire, de manière équivalente:

$$F(x) = \begin{cases} x + 1 & \text{si } S_x = \varnothing; \\ F(x-2) + F(x-1) & \text{si } S_x \neq \varnothing. \end{cases} \tag{2}$$

Pour tout $x \in \mathbb{N}$ tel que $S_x \neq \varnothing$, les nombres $x-2$ et $x-1$ appartiennent à l'ensemble S_x et, par conséquent, pour toute fonction F de domaine $\mathbb{N}$, on a

$$F(x-2) = F_{|S_x}(x-2) \text{ et } F(x-1) = F_{|S_x}(x-1). \tag{1.3.21}$$

L'équation (2) peut donc s'écrire finalement de manière équivalente:

$$F(x) = \begin{cases} x + 1 & \text{si } S_x = \varnothing; \\ F_{|S_x}(x-2) + F_{|S_x}(x-1) & \text{si } S_x \neq \varnothing. \end{cases} \tag{3}$$

Cette équation est de la forme décrite dans le principe de définition par récurrence noethérienne (1.5.5), notamment:

- le terme $\mathbf{T}_1$ est $x + 1$
- le terme $\mathbf{T}_2$ est $F(x-2) + F(x-1)$.

On a bien dans l'équation (3) le terme $(F_{|S_x}|F)\mathbf{T}_2$. Les conditions du schéma sont satisfaites: la variable F ne figure pas librement dans $\mathbf{T}_1$, le terme $F_{|S_x}$ est librement substituable à F dans $\mathbf{T}_2$. D'après le principe en question il existe une fonction F et une seule de domaine $\mathbb{N}$ vérifiant l'équation (3) (pour tout $x \in \mathbb{N}$). Il en est donc de même de l'équation équivalente (1).

1.5.9. Interprétation du principe de définition par récurrence noethérienne

Si E est un ensemble ordonné noethérien, définir une fonction F de domaine E suivant le principe mentionné, consiste premièrement à définir explicitement les valeurs $F(x)$ pour les éléments minimaux x de E en donnant un terme $\mathbf{T}_1$ dans lequel F ne figure pas (librement). Donc, pour ces éléments de E, il n'y a pas de « récurrence »: F n'est pas « utilisée dans sa propre définition ». Pour les éléments x non minimaux ($S_x \neq \varnothing$), F peut figurer à droite dans l'équation $F(x) = \mathbf{T}_2$, comme dans 1.5.8(1), mais on doit pouvoir remplacer toutes les occurrences (libres) de F dans $\mathbf{T}_2$ par $F_{|S_x}$ sans changer la valeur de ce terme. Pour cela, il suffit que F soit appliquée dans ce terme uniquement à des éléments z de E strictement inférieurs à x, ce qui implique $F(z) = F_{|S_x}(z)$. Dans l'exemple qui précède 1.5.8(1), ce sont les éléments $x-1, x-2$. Le calcul de $F(x)$ au moyen d'une telle équation fait donc appel à la valeur $F(z)$ pour certains éléments z de E strictement inférieurs à x. Le calcul de ces valeurs $F(z)$ peut nécessiter l'usage de la même équation, c'est-à-dire faire appel à des valeurs $F(z')$ correspondant à des éléments z' encore plus petits, etc. Cependant, ce processus ne peut pas se répéter indéfiniment puisqu'il n'existe pas de suite infinie strictement décroissante $x_0 > x_1 > x_2 > \ldots$ d'éléments de E (1.4.7). On aboutit nécessairement à des éléments minimaux de E, pour lesquels F retourne la valeur du terme $\mathbf{T}_1$.

1.6 Monoïdes

1.6.1. Opérations binaires dans un ensemble

On appelle *opération binaire dans un ensemble* E une application φ de l'ensemble $E \times E$ dans lui-même. Autrement dit, par définition,

$$(\varphi \text{ est une opération binaire dans } E) \Leftrightarrow \varphi : E \times E \to E.$$

Étant donné un symbole d'opération binaire $*$ (symbole fonctionnel binaire selon la terminologie des langages du premier ordre), on dit qu'une opération binaire φ dans l'ensemble E est *notée au moyen du signe d'opération* $*$ si l'on pose :

$$\forall x \in E : \forall y \in E : \quad x * y = \varphi(x, y).$$

NB. Lorsqu'on parle ainsi de « noter l'opération φ » au moyen du signe d'opération $*$, cela ne signifie pas que ce symbole $*$ désigne l'opération φ (fonction) elle-même. On peut écrire $\varphi(x, y) = x * y$, mais il est incorrect d'écrire $\varphi = *$. Un signe d'opération tout seul ne constitue pas une expression syntaxiquement correcte; il doit toujours être combiné avec des arguments x, y.

Opérations binaires associatives. On dit qu'une opération binaire φ dans l'ensemble E est *associative* si, quels que soient les éléments x, y, z de E :

$$\varphi(\varphi(x, y), z) = \varphi(x, \varphi(y, z)).$$

Lorsque φ est notée au moyen du signe d'opération $*$, cette égalité s'écrit :

$$(x * y) * z = x * (y * z).$$

Élément neutre pour une opération binaire. Soit φ une opération binaire dans E. On dit qu'un élément e de E est *neutre pour* φ si, pour tout $x \in E$:

$$\varphi(x, e) = x \text{ et } \varphi(e, x) = x.$$

Si φ est notée au moyen du signe d'opération $*$, ces égalités s'écrivent

$$x * e = x \text{ et } e * x = x.$$

Il existe au plus un élément de E neutre pour φ. En effet, si e et e' sont deux éléments neutres de E pour cette opération, alors $\varphi(e, e') = e$ puisque e' est neutre et $\varphi(e, e') = e'$ puisque e est neutre, donc $e = e'$.

1.6.2. Monoïdes

On appelle *monoïde* un triplet (M, φ, e) dans lequel

M est un ensemble;
φ est une opération binaire associative dans M;
e est élément neutre de M pour φ.

Un tel triplet est appelé souvent un « ensemble M muni d'une opération binaire associative φ et d'un élément neutre e ». L'ensemble M est appelé le *support* du monoïde (M, φ, e). Lorsque l'opération φ est notée au moyen d'un symbole d'opération $*$ (1.6.1), on écrit souvent abusivement $(M, *, e)$ pour désigner le monoïde (M, φ, e).

Exemple 1. Le triplet $(\mathbb{N}, \cdot, 1)$, à savoir l'ensemble $\mathbb{N}$ muni de l'opération de multiplication usuelle (associative) et de l'élément neutre 1, est appelé le *monoïde multiplicatif des entiers naturels*.

Exemple 2. Le triplet $(\mathbb{N}, +, 0)$, à savoir l'ensemble $\mathbb{N}$ muni de l'opération usuelle d'addition (associative) et de l'élément neutre 0 est un monoïde appelé le *monoïde additif des entiers naturels*.

Exemple 3. Si E est un ensemble quelconque, le triplet $(\mathcal{P}(E), \cup, \varnothing)$, à savoir l'ensemble des parties de E muni de l'opération (associative) de réunion et de l'élément $\varnothing \in \mathcal{P}(E)$ (neutre pour la réunion), est un monoïde. Le triplet $(\mathcal{P}(E), \cap, E)$ est un autre monoïde de même support $\mathcal{P}(E)$.

Les monoïdes des exemples 1 à 3 sont des monoïdes *commutatifs*, en ce sens que leur opération binaire φ est commutative : quels que soient les éléments x, y du support M, on a

$$\varphi(x, y) = \varphi(y, x).$$

Par contre, le monoïde de l'exemple suivant n'est pas commutatif.

Exemple 4. L'ensemble M des matrices réelles 2×2 muni de l'opération (associative) de produit matriciel et de la matrice diagonale $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ (neutre pour l'opération de produit considérée) est un monoïde. Ce monoïde n'est pas commutatif : le produit AB de deux matrices réelles carrées n'est pas égal au produit BA en général.

Exemple 5. Si X est un ensemble quelconque, l'ensemble $\mathrm{W}(X)$ des séquences sur X (2.2.1) muni de l'opération associative de concaténation (notée avec le signe `cat` ou comme une multiplication) et de l'élément neutre Λ — autrement dit le triplet $(\mathrm{W}(X), \mathsf{cat}, \Lambda)$ — est un monoïde, que l'on appelle le monoïde des séquences sur X.

Ce monoïde n'est pas commutatif si l'ensemble X possède au moins deux éléments distincts. Par exemple, si a et b sont des éléments de X distincts ($a \neq b$), on a $\langle\!\langle\, a \,\rangle\!\rangle \langle\!\langle\, b \,\rangle\!\rangle = \langle\!\langle\, a, b \,\rangle\!\rangle \neq \langle\!\langle\, b, a \,\rangle\!\rangle = \langle\!\langle\, b \,\rangle\!\rangle \langle\!\langle\, a \,\rangle\!\rangle$. Par contre, si $X = \varnothing$ ou si X n'a qu'un seul élément ($X = \{a\}$), le monoïde $\mathrm{W}(X)$ est commutatif.

1.6.3. Le monoïde des relations dans un ensemble E

Nous utiliserons systématiquement la notation $\mathcal{R}(E)$ pour désigner *l'ensemble des relations dans un ensemble E*. En vertu de cette définition et de celle d'une « relation dans E » (1.3.3), nous avons les équivalences

$$\begin{aligned} R \in \mathcal{R}(E) &\Leftrightarrow R \text{ est une relation dans } E \\ &\Leftrightarrow R \subset E \times E \\ &\Leftrightarrow R \in \mathcal{P}(E \times E). \end{aligned}$$

La notation $\mathcal{P}(A)$ désigne l'ensemble des parties (ou sous-ensembles) d'un ensemble A et la troisième équivalence ci-dessus est une application de la formule générale de théorie des ensembles : $X \in \mathcal{P}(A) \Leftrightarrow X \subset A$. L'ensemble $\mathcal{R}(E)$ des relations dans E est donc l'ensemble des parties de $E \times E$.

Nous avons vu au §1.3.4 que la composée $S \circ R$ de deux relations dans E est elle-même une relation dans E. Ceci nous permet de munir l'ensemble $\mathcal{R}(E)$ d'une opération binaire $\varphi : \mathcal{R}(E) \times \mathcal{R}(E) \to \mathcal{R}(E)$ en posant, pour chaque couple $(R, S) \in \mathcal{R}(E) \times \mathcal{R}(E)$,

$$\varphi(R, S) = S \circ R.$$

Dans certains contextes, notamment celui de la théorie des automates, il est commode de noter multiplicativement RS la composée $S \circ R$ de deux relations dans un ensemble E :

$$RS = S \circ R,$$

d'où la formule $(a, b) \in RS \Leftrightarrow \exists x(a\,R\,x$ et $x\,S\,b)$. Nous noterons ainsi multiplicativement l'opération φ dans $\mathcal{R}(E)$ définie ci-dessus et nous l'appellerons le *produit de composition* dans $\mathcal{R}(E)$.

Cette opération binaire dans $\mathcal{R}(E)$ est associative (1.3.4). L'ensemble $\mathcal{R}(E)$ possède en outre un élément neutre pour cette opération, à savoir la relation Id_E, relation identique de E (1.3.8). Ces propriétés du produit de composition se traduisent par les formules suivantes, valables pour des relations R, S, T quelconques dans E :

$$(RS)T = R(ST); \quad R\,\mathrm{Id}_E = R; \quad \mathrm{Id}_E R = R.$$

Le triplet $(\mathcal{R}(E), \varphi, \mathrm{Id}_E)$, où φ est l'opération de produit de composition dans E, est donc un monoïde, appelé le *monoïde des relations dans* E.

1.6.4. Monoïdes additifs et multiplicatifs

Beaucoup d'opérations binaires importantes sont notées soit additivement, c'est-à-dire au moyen du signe +, soit muliplicativement, c'est-à-dire au moyen du signe $\cdot$ (celui-ci pouvant être omis). Un monoïde dont l'opération est notée au moyen du signe + (resp. $\cdot$) est appelé un monoïde *additif* (resp. *multiplicatif*). Mais ces adjectifs se rapportent seulement à la notation utilisée pour l'opération et non à une propriété de cette opération. Par exemple, le monoïde $(\mathrm{W}(X), \varphi, \Lambda)$, dans lequel φ est l'opération de concaténation des séquences, est noté multiplicativement lorsqu'on écrit xy pour la concaténation $\varphi(x, y)$ de deux séquences sur X.

1.6.5. Itérés d'un élément d'un monoïde

Théorème. Soient $(M, *, e)$ un monoïde dont l'opération est notée au moyen du signe $*$, et a un élément de M. Il existe une fonction $F : \mathbb{N} \to M$ et une seule telle que, pour tout $n \in \mathbb{N}$:

$$F(n) = \begin{cases} e & \text{si } n = 0; \\ a * F(n-1) & \text{si } n \neq 0. \end{cases} \tag{1}$$

La démonstration est une application évidente du principe de définition par récurrence noethérienne (1.5.7), que l'on applique à l'ensemble $E = \mathbb{N}$ muni de l'ordre usuel. En considérant cet ordre dans $\mathbb{N}$, on peut écrire (1) de manière équivalente

$$F(n) = \begin{cases} e & \text{si } S_n = \varnothing; \\ a * F_{|S_n}(n-1) & \text{si } S_n \neq \varnothing. \end{cases}$$

On applique 1.5.7 en prenant pour $\mathbf{T}_1$ le terme e et pour $\mathbf{T}_2$ le terme $F(n-1)$.

Définition. Étant donné un monoïde $(M, *, e)$ et un élément a de M, on désigne par it_a l'unique fonction $F : \mathbb{N} \to M$ qui vérifie (1) pour tout $n \in \mathbb{N}$. On a donc, par définition,

$$\mathrm{it}_a(0) = e; \tag{2}$$

$$\forall n \in \mathbb{N} : \quad \mathrm{it}_a(n+1) = a * \mathrm{it}_a(n). \tag{3}$$

On peut donc écrire, successivement :

$$\begin{aligned} &\mathrm{it}_a(0) = e; \\ &\mathrm{it}_a(1) = a; \\ &\mathrm{it}_a(2) = a * a; \\ &\mathrm{it}_a(3) = a * a * a; \\ &\text{etc.} \end{aligned}$$

L'élément $\mathrm{it}_a(n)$ de M est appelé le *n-ième itéré* de a.

Exemple. Supposons que $(M, *, e)$ soit le monoïde $(\mathrm{W}(X), \mathsf{cat}, \Lambda)$ des séquences sur un ensemble X. Soient u, v des éléments de X et $a = \langle\!\langle\, u, v \,\rangle\!\rangle$. On a

$$\begin{aligned}
&\mathrm{it}_{\langle\!\langle u,v \rangle\!\rangle}(0) = \Lambda;\\
&\mathrm{it}_{\langle\!\langle u,v \rangle\!\rangle}(1) = \langle\!\langle\, u, v \,\rangle\!\rangle;\\
&\mathrm{it}_{\langle\!\langle u,v \rangle\!\rangle}(2) = \langle\!\langle\, u, v, u, v \,\rangle\!\rangle;\\
&\mathrm{it}_{\langle\!\langle u,v \rangle\!\rangle}(3) = \langle\!\langle\, u, v, u, v, u, v \,\rangle\!\rangle;\\
&\text{etc.}
\end{aligned}$$

Notation multiplicative. On utilise la notation particulière

$$\mathrm{it}_a(n) = a^n$$

lorsque l'opération du monoïde M est notée multiplicativement. Avec cette notation, les formules (2) et (3) prennent la forme suivante :

$$\begin{aligned}
&a^0 = e;\\
\forall n \in \mathbb{N} :\quad &a^{n+1} = a(a^n).
\end{aligned}$$

Par exemple, on utilise cette notation pour une séquence a sur un ensemble X, c'est-à-dire un élément a du monoïde $M = \mathrm{W}(X)$, lorsque celui-ci est noté multiplicativement. Ainsi dans l'exemple précédent, au lieu de $\mathrm{it}_{\langle\!\langle u,v \rangle\!\rangle}(n)$, on peut écrire

$$\langle\!\langle\, u, v \,\rangle\!\rangle^n.$$

Notation additive. On utilise la notation particulière

$$\mathrm{it}_a(n) = na$$

pour un élément a d'un monoïde M dont l'opération est notée additivement. Avec cette notation, les formules (2) et (3) prennent la forme suivante :

$$\begin{aligned}
&0a = e;\\
\forall n \in \mathbb{N} :\quad &(n+1)a = a + (na).
\end{aligned}$$

1.6.6. Théorème

Soit a un élément d'un monoïde $(M, *, e)$. Quels que soient $m, n \in \mathbb{N}$, on a

$$\mathrm{it}_a(m+n) = \mathrm{it}_a(m) * \mathrm{it}_a(n).$$

Notations particulières. Lorsque l'opération du monoïde est notée multiplicativement, l'élément $\mathrm{it}_a(n)$ est noté aussi a^n, et avec cette notation l'égalité ci-dessus devient

$$a^{m+n} = a^m \cdot a^n.$$

Lorsque l'opération du monoïde est notée additivement, $\mathrm{it}_a(n)$ est noté aussi na, et l'égalité s'écrit

$$(m+n)a = ma + na.$$

Démonstration. Pour un $n \in \mathbb{N}$ fixe, mais quelconque, on démontre la proposition

$$\text{(1)} \qquad \forall m \in \mathbb{N} :\quad \underbrace{\mathrm{it}_a(m) * \mathrm{it}_a(n) = \mathrm{it}_a(m+n)}_{P(m)}$$

par récurrence naturelle sur m (cf. §1.5.2, figure 1.6).

Démonstration de $P(0)$:

$$\begin{aligned} \mathrm{it}_a(0) * \mathrm{it}_a(n) &= e * \mathrm{it}_a(n) && 1.6.5(2) \\ &= \mathrm{it}_a(n) && e \text{ neutre} \\ &= \mathrm{it}_a(0+n). \end{aligned}$$

Démonstration de $P(m+1)$ sous l'hypothèse de récurrence $m \in \mathbb{N}$ et $P(m)$:

$$\begin{aligned} \mathrm{it}_a(m+1) * \mathrm{it}_a(n) &= (a * \mathrm{it}_a(m)) * \mathrm{it}_a(n) && 1.6.5(3) \\ &= a * (\mathrm{it}_a(m) * \mathrm{it}_a(n)) && \text{associativité de } * \\ &= a * \mathrm{it}_a(m+n) && \text{hypothèse de récurrence} \\ &= \mathrm{it}_a(m+n+1) && 1.6.5(3) \\ &= \mathrm{it}_a((m+1)+n). \end{aligned}$$

Commentaire. À première vue, il n'y a pas de raison de démontrer le théorème par récurrence sur m, comme nous l'avons fait, plutôt que de démontrer

$$\forall n \in \mathbb{N} : \quad \mathrm{it}_a(m) * \mathrm{it}_a(n) = \mathrm{it}_a(m+n) \tag{2}$$

par récurrence sur n, pour un m fixe quelconque. Mais on n'y arrive pas, et il faut essayer pour s'en rendre compte. Cette dissymétrie provient de la formule 1.6.5(3). On peut démontrer (2) par récurrence sur n à condition de démontrer au préalable (par récurrence sur n) la formule suivante, symétrique de 1.6.5(3) :

$$\forall n \in \mathbb{N} : \quad \mathrm{it}_a(n+1) = \mathrm{it}_a(n) * a.$$

1.6.7. Exercice

Soient $(M, *, e)$ un monoïde et soient $a, b \in M$. Démontrer les propositions suivantes par récurrence sur n (pour un m fixe quelconque dans le cas de (4)).

(1) $\forall n \in \mathbb{N} : \ \mathrm{it}_e(n) = e.$

(2) Si $a * b = b * a$, alors $\forall n \in \mathbb{N} : \ \mathrm{it}_a(n) * b = b * \mathrm{it}_a(n).$

Conséquence. Lorsque deux éléments a, b de M sont tels que $a * b = b * a$, on dit que ces éléments *commutent*. C'est le cas trivialement si $a = b$. D'après la formule (2), on a donc, pour tout $n \in \mathbb{N}$:

$$\mathrm{it}_a(n) * a = a * \mathrm{it}_a(n).$$

(3) Si $a * b = b * a$, alors $\forall n \in \mathbb{N} : \ \mathrm{it}_{a*b}(n) = \mathrm{it}_a(n) * \mathrm{it}_b(n).$

(4) $\forall m \in \mathbb{N} : \forall n \in \mathbb{N} : \ \mathrm{it}_a(mn) = \mathrm{it}_{\mathrm{it}_a(m)}(n).$

Donner les formules qui correspondent à (3) et (4) dans la notation multiplicative ($a*b = a \cdot b$, $\mathrm{it}_a(n) = a^n$) et dans la notation additive ($a * b = a + b$, $\mathrm{it}_a(n) = na$).

1.6.8. Monoïde opposé d'un monoïde

On appelle *monoïde opposé* d'un monoïde (M, φ, e) le monoïde (M, ψ, e) où l'opération $\psi : M \times M \to M$ est définie par

$$\forall x \in M : \forall y \in M : \quad \psi(x, y) = \varphi(y, x).$$

Il faut s'assurer bien entendu que (M, ψ, e) est un monoïde, à savoir que l'opération ψ est associative, et que e est neutre pour cette opération. En notant φ au moyen du signe $*$ et ψ

au moyen de $*_{\mathrm{op}}$, on a en effet, quels que soient $x, y, z \in M$:

$$(x *_{\mathrm{op}} y) *_{\mathrm{op}} z = z * (y * x) = (z * y) * x = x *_{\mathrm{op}} (y *_{\mathrm{op}} z).$$
$$x *_{\mathrm{op}} e = e * x = e.$$
$$e *_{\mathrm{op}} x = x * e = e.$$

1.6.9. Homomorphismes de monoïdes

Soient $(M, *, e)$ et $(M', *', e')$ deux monoïdes. Un *homomorphisme* de $(M, *, e)$ dans $(M', *', e')$ est une fonction $f : M \to M'$ telle que

(1) $$f(e) = e';$$

(2) $$\forall x \in M : \forall y \in M : \quad f(x * y) = f(x) *' f(y).$$

Exemple 1. Soient X un ensemble et $f : \mathrm{W}(X) \to \mathbb{N}$ la fonction « longueur des séquences sur X », autrement dit la fonction $f = (\lambda x \in \mathrm{W}(X) : \lg(x))$. On a

$$f(\Lambda) = 0;$$
$$\forall x \in \mathrm{W}(X) : \forall y \in \mathrm{W}(X) : \quad f(x \mathbin{\mathsf{cat}} y) = f(x) + f(y).$$

Ces propriétés de f peuvent s'exprimer en disant que f est un homomorphisme du monoïde $(\mathrm{W}(X), \mathsf{cat}, \Lambda)$ dans le monoïde $(\mathbb{N}, +, 0)$.

Exemple 2. Soit $\mathbb{R}_+^*$ l'ensemble des nombres réels strictement positifs (> 0). La fonction logarithme $f = (\lambda x \in \mathbb{R}_+^* : \mathrm{Log}(x))$ est un homomorphisme du monoïde multiplicatif $(\mathbb{R}_+^*, \cdot, 1)$ dans le monoïde additif $(\mathbb{R}, +, 0)$, puisque $f(1) = 0$ et $f(xy) = f(x) + f(y)$.

Exemple 3. La fonction ρ de renversement des séquences sur X (2.2.3) est un homomorphisme du monoïde $(\mathrm{W}(X), \mathsf{cat}, \Lambda)$ dans le monoïde opposé $(\mathrm{W}(X), \mathsf{cat}_{\mathrm{op}}, \Lambda)$, car on a $\rho(\Lambda) = \Lambda$ (2.2.3(1)) et quels que soient $x, y \in \mathrm{W}(X)$:

$$\begin{aligned} \rho(x \mathbin{\mathsf{cat}} y) &= \rho(y) \mathbin{\mathsf{cat}} \rho(x) \qquad 2.2.4(1) \\ &= \rho(x) \mathbin{\mathsf{cat}_{\mathrm{op}}} \rho(y). \end{aligned}$$

Exemple 4. La fonction $\mathrm{it}_a : \mathbb{N} \to M$, pour un élément a d'un monoïde $(M, *, e)$ est un homomorphisme du monoïde $(\mathbb{N}, +, 0)$ dans le monoïde M, d'après 1.6.5(2) et 1.6.6.

1.6.10. Théorèmes

(1) Pour tout monoïde $(M, *, e)$, la fonction identique $\mathrm{Id}_M : M \to M$ est un homomorphisme de $(M, *, e)$ dans lui-même.

(2) Soient $(M, *, e)$, $(M', *', e')$, $(M'', *'', e'')$ trois monoïdes. Si f est un homomorphisme de $(M, *, e)$ dans $(M', *', e')$ et si g est un homomorphisme de $(M', *', e')$ dans $(M'', *'', e'')$, alors la fonction $g \circ f : M \to M''$ est un homorphisme de $(M, *, e)$ dans $(M'', *'', e'')$.

Démonstration : voir [A].

Chapitre 2 Séquences

2.1 Séquences, concaténation

2.1.1. Intervalles de $\mathbb{N}$

Si a et b sont deux entiers naturels ($a, b \in \mathbb{N}$), nous désignons par $[a \,..\, b]$ l'intervalle d'extrémités a, b de $\mathbb{N}$, à savoir l'ensemble des entiers naturels k tels que $a \leq k \leq b$. Cette définition s'écrit, selon (1.2.2),

$$\begin{aligned}[a \,..\, b] &= \{k \mid k \in \mathbb{N} \text{ et } a \leq k \leq b\};\\ &= \{k \in \mathbb{N} \mid a \leq k \leq b\}.\end{aligned}$$

La notation $a \leq k \leq b$ est une abréviation de la conjonction ($a \leq k$ et $k \leq b$). Si $a, b \in \mathbb{N}$, on a donc par définition (1.2.2)

$$k \in [a \,..\, b] \Leftrightarrow (k \in \mathbb{N} \text{ et } a \leq k \text{ et } k \leq b).$$

Notre définition de $[a \,..\, b]$ ne suppose pas que l'on ait $a \leq b$. Elle peut être appliquée au cas où $a > b$ et elle entraîne dans ce cas que l'intervalle $[a \,..\, b]$ est vide ($[a \,..\, b] = \varnothing$) car, si $a > b$, il n'existe pas d'entier naturel k tel que l'on ait $a \leq k$ et $k \leq b$.

Les intervalles de $\mathbb{N}$ la forme $[1 \,..\, n]$ jouent un rôle particulièrement important. Premièrement, ils servent d'ensembles de référence en tant qu'ensembles finis. Un ensemble E est fini, *par définition*, s'il existe un entier naturel n et une bijection de l'ensemble $[1 \,..\, n]$ sur E. Cet entier n est alors unique et il est appelé le nombre d'éléments de E.

Deuxièmement, les intervalles de $\mathbb{N}$ de la forme $[1 \,..\, n]$ interviennent dans la notion de séquence, définie ci-dessous. En prévision de cette définition, nous notons les formules suivantes, que nous admettons sans démonstration :

Si $m \in \mathbb{N}$ et $n \in \mathbb{N}$, alors

(1) $m = 0 \Leftrightarrow [1 \,..\, m] = \varnothing$;
(2) $m = n \Leftrightarrow [1 \,..\, m] = [1 \,..\, n]$.

2.1.2. Séquences : définition mathématique

Dans ce cours, nous appelons *séquence* toute fonction dont le domaine est un intervalle de $\mathbb{N}$ de la forme $[1 \,..\, n]$. Par exemple, le tableau suivant représente une fonction α telle que $\mathrm{Dom}\alpha = \{1, 2, 3, 4\} = [1 \,..\, 4]$. Cette fonction est donc une séquence. On dit qu'il s'agit d'une séquence de *longueur* 4.

(1)

k	$\alpha[k]$
1	0.12
2	2.50
3	0.12
4	3.14

La valeur $\alpha(k)$ qui correspond à un élément k du domaine d'une séquence α sera toujours désignée par $\alpha[k]$, avec des parenthèses carrées [,]. Dans l'exemple (1), toutes les valeurs $\alpha[k]$ sont des nombres réels :

$$\alpha[1] = 0.12, \quad \alpha[2] = 2.50, \quad \alpha[3] = 0.12, \quad \alpha[4] = 3.14.$$

On dit que α est une séquence de nombres réels. Cela peut s'exprimer par l'inclusion $\mathrm{Prt}\,\alpha \subset \mathbb{R}$. La portée d'une séquence est toujours un ensemble fini, mais à part cela, cet ensemble peut être quelconque. C'est seulement le domaine des fonctions qui intervient dans la définition générale des séquences, définition qui se formalise par l'équivalence:

(2) $\quad \alpha$ est une séquence $\Leftrightarrow \big(\alpha$ est une fonction et $\exists n \in \mathbb{N} : \mathrm{Dom}\,\alpha = [1 \,..\, n]\big)$.

Longueur d'une séquence. — Si α est une séquence, non seulement il existe un entier naturel n tel que $\mathrm{Dom}\,\alpha = [1 \,..\, n]$, mais encore cet entier est unique, en vertu de 2.1.1 (2). Cet unique entier est appelé la *longueur* de α et noté $\lg(\alpha)$.

2.1.3. Notation des séquences

Pour donner une séquence α explicitement, on peut utiliser une table de cette fonction, comme dans l'exemple 2.1.2 (1). Il vient immédiatement à l'idée de simplifier cette table en omettant sa première colonne dont on sait qu'elle ne contient qu'une numérotation des lignes de la table. Il reste le tableau suivant:

0.12
2.50
0.12
3.14

Nous utiliserons à la place de ce tableau l'expression horizontale

$$\langle\!\langle\, 0.12,\ 2.50,\ 0.12,\ 3.14 \,\rangle\!\rangle.$$

De façon générale, si $\mathbf{a}_1, \ldots, \mathbf{a}_n$ ($n \geq 1$) sont des expressions quelconques, l'expression

$$\langle\!\langle\, \mathbf{a}_1, \ldots, \mathbf{a}_n \,\rangle\!\rangle$$

représente la séquence α de longueur n telle que $\alpha[1] = \mathbf{a}_1, \ldots, \alpha[n] = \mathbf{a}_n$. Une telle fonction, en tant qu'ensemble de couples, peut être représentée encore par l'expression $\{1 \mapsto \mathbf{a}_1, \ldots, n \mapsto \mathbf{a}_n\}$. En fait, l'égalité

(1) $$\langle\!\langle\, \mathbf{a}_1, \ldots, \mathbf{a}_n \,\rangle\!\rangle = \{1 \mapsto \mathbf{a}_1, \ldots, n \mapsto \mathbf{a}_n\}$$

peut être prise comme définition de la notation $\langle\!\langle\, \mathbf{a}_1, \ldots, \mathbf{a}_n \,\rangle\!\rangle$. Les formules suivantes (où l'on suppose $n \geq 1$) découlent trivialement de cette définition:

(2) $\langle\!\langle\, \mathbf{a}_1, \ldots, \mathbf{a}_n \,\rangle\!\rangle$ est une séquence;

(3) $\lg(\langle\!\langle\, \mathbf{a}_1, \ldots, \mathbf{a}_n \,\rangle\!\rangle) = n$;

(4) Pour tout $k \in [1 \,..\, n]$: $\quad \langle\!\langle\, \mathbf{a}_1, \ldots, \mathbf{a}_n \,\rangle\!\rangle[k] = \mathbf{a}_k$.

(5) (x est une séquence de longueur n) $\Leftrightarrow x = \langle\!\langle\, x[1], \ldots, x[n] \,\rangle\!\rangle$.

Il ne faut pas confondre la notation $\langle\!\langle\, \mathbf{a}_1, \ldots, \mathbf{a}_n \,\rangle\!\rangle$ avec la notation $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$. Dans une expression de la forme $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ on peut changer l'ordre d'écriture des $\mathbf{a}_i$ et supprimer les répétitions sans changer l'ensemble représenté. Par exemple, si a, b, c désignent trois objets distincts, l'ensemble $\{a, b, a, c\}$ est égal aux ensembles $\{a, b, c\}$, $\{b, a, c\}$, etc. Par contre, la séquence $\langle\!\langle\, a, b, a, c \,\rangle\!\rangle$ n'est pas égale à la séquence $\langle\!\langle\, a, b, c \,\rangle\!\rangle$, laquelle n'est pas égale à la séquence $\langle\!\langle\, c, b, a \,\rangle\!\rangle$, etc.

2.1.4. La séquence vide

On sait que l'ensemble $\varnothing$ est une fonction (1.3.15) et qu'il est l'unique fonction F telle que $\mathrm{Dom}\,F = \varnothing$. Comme l'ensemble $\varnothing$ est aussi égal à l'intervalle $[1 \,..\, 0]$ de $\mathbb{N}$ (2.1.1), on

peut dire que la fonction vide a pour domaine un intervalle $[1 \,..\, n]$ de $\mathbb{N}$, donc qu'elle est une séquence, et en outre qu'elle est l'unique séquence de longueur 0.

Nous l'appellerons naturellement la *séquence vide*, mais, au lieu du symbole $\varnothing$, nous utiliserons plutôt le symbole Λ (lettre grecque lambda majuscule) pour désigner cette séquence particulière. Le symbole ε (epsilon) est aussi utilisé dans la littérature pour désigner cette séquence. En résumé, nous avons par définition :

(1) Λ est une séquence.

(2) $\lg(\Lambda) = 0$.

(3) Si x est une séquence, alors

$$\lg(x) = 0 \Leftrightarrow x = \Lambda.$$

2.1.5. Égalité de séquences

L'égalité de deux séquences est une égalité de fonctions. Comme cas particulier de la formule de 1.3.13, on peut énoncer :

Deux séquences $x = \langle\!\langle\, x[1], \ldots, x[m] \,\rangle\!\rangle$, et $y = \langle\!\langle\, y[1], \ldots, y[n] \,\rangle\!\rangle$ sont égales ($x = y$) si et seulement si $m = n$ et, pour tout $k \in [1 \,..\, m]$, $x[k] = y[k]$. Cela peut se formuler aussi : si x et y sont des séquences, alors

(1) $$x = y \;\Leftrightarrow\; \big(\lg(x) = \lg(y) \text{ et } \forall k \in [1 \,..\, \lg(x)] : x[k] = y[k]\big).$$

2.1.6. Séquences élémentaires

Une séquence *élémentaire* est une séquence de longueur 1, autrement dit une séquence de la forme $\langle\!\langle\, a \,\rangle\!\rangle$. Les formules (3) et (4) de 2.1.3, appliquées à ce cas particulier, donnent :

$$\lg\langle\!\langle\, a \,\rangle\!\rangle = 1 \;\text{ et }\; \langle\!\langle\, a \,\rangle\!\rangle[1] = a.$$

D'après 2.1.4(3), on a donc $\langle\!\langle\, a \,\rangle\!\rangle \neq \Lambda$.

NB. Il faut se garder de confondre un objet a avec la séquence élémentaire $\langle\!\langle\, a \,\rangle\!\rangle$. Celle-ci est la fonction (ensemble de couples) $\{1 \mapsto a\}$, ce qui est bien différent de a. Néanmoins, dans certains contextes, c'est un abus de langage commode que d'écrire a au lieu de $\langle\!\langle\, a \,\rangle\!\rangle$.

2.1.7. Concaténation de deux séquences

À partir de deux séquences, $x = \langle\!\langle\, x[1], \ldots, x[m] \,\rangle\!\rangle$ et $y = \langle\!\langle\, y[1], \ldots, y[n] \,\rangle\!\rangle$, on peut former la séquence

$$z = \langle\!\langle\, x[1], \ldots, x[m], y[1], \ldots, y[n] \,\rangle\!\rangle.$$

Cette séquence, de longueur $m+n$, est appelée *concaténation* de x et de y ([4]). Cette opération est notée le plus souvent comme un produit :

$$z = xy.$$

Nous utiliserons aussi le symbole d'opération plus spécifique cat :

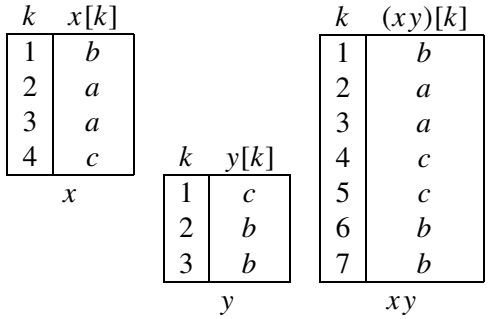
$$z = x \;\mathsf{cat}\; y.$$

Par exemple, la concaténation de $x = \langle\!\langle\, b, a, a, c \,\rangle\!\rangle$ et $y = \langle\!\langle\, c, b, b \,\rangle\!\rangle$ est la séquence $xy = \langle\!\langle\, b, a, a, c, c, b, b \,\rangle\!\rangle$. Ces séquences sont présentées ci-dessous sous la forme tabulaire.

---

[4] Concaténation = enchaînement. Du latin *catena*, la chaîne.

Cela peut faciliter la compréhension de la définition formelle qui suit.



Définition. Si x et y sont des séquences, la séquence désignée par xy est caractérisée par les propriétés suivantes :

(1) $\lg(xy) = \lg(x) + \lg(y)$;

(2) Pour tout $k \in [1 \,..\, \lg(xy)]$:

$$(xy)[k] = \begin{cases} x[k] & \text{si } k \in [1 \,..\, \lg(x)]; \\ y[k - \lg(x)] & \text{si } k \in [\lg(x) + 1 \,..\, \lg(xy)]. \end{cases}$$

2.1.8. Exercice

On peut récrire la formule 2.1.7(2) de la manière suivante :

$$\forall k \in [1 \,..\, \lg(xy)] : \quad (xy)[k] = \begin{cases} x[k] & \text{si } k \in [1 \,..\, \lg(x)]; \\ y[k - \lg(x)] & \text{si } k \notin [1 \,..\, \lg(x)]. \end{cases}$$

Le membre de droite de cette dernière égalité est ce qu'on appelle un *terme conditionnel*. Ce genre de terme est d'un usage fréquent dans les définitions. Par exemple, la valeur absolue $|x|$ d'un nombre réel x est souvent définie par

$$|x| = \begin{cases} x & \text{si } x \geq 0; \\ -x & \text{si } x < 0 \quad (\text{non}(x \geq 0)). \end{cases}$$

On peut l'écrire $|x| = (\textbf{\textit{Si}}\ x \geq 0\ \textbf{\textit{alors}}\ x\ \textbf{\textit{sinon}}\ -x)$, ou plus succinctement :

$$|x| = \textbf{\textit{Si}}(\, x \geq 0,\ x, -x\,).$$

De façon générale, un terme conditionnel est un terme de la forme $\textbf{\textit{Si}}(P, T_1, T_2)$, où P est une proposition et T_1, T_2 sont des termes. Par définition, les implications suivantes sont vraies :

$$P \Rightarrow \big(\textbf{\textit{Si}}(P, T_1, T_2) = T_1\big); \qquad \text{non}P \Rightarrow \big(\textbf{\textit{Si}}(P, T_1, T_2) = T_2\big).$$

On demande de reformuler la définition de la concaténation de deux séquences x, y en remplaçant les deux formules 2.1.7(1), 2.1.7(2) par une seule égalité de la forme $xy = \ldots$ Utiliser à droite la notation lambda (1.3.17) pour définir la séquence xy en tant que fonction de domaine $[1 \,..\, \lg(xy)]$.

2.1.9. Propriétés de la concaténation

Nous donnons ici quelques formules utiles sur la concaténation de séquences. Dans toutes ces formules, on suppose que x, y, u sont des séquences. Les démonstrations sont fournies en annexe [A].

(1) $\quad x \neq \Lambda \Rightarrow (xy)[1] = x[1]$.

(2) $k \in [1 \,..\, \lg(y)] \;\Rightarrow\; y[k] = (xy)[\lg(x) + k]$.

(3) $x\Lambda = x$ et $\Lambda x = x$. (Λ est neutre pour la concaténation)

(4) $xy = \Lambda \Leftrightarrow (x = \Lambda$ et $y = \Lambda)$.
$xy \neq \Lambda \Leftrightarrow (x \neq \Lambda$ ou $y \neq \Lambda)$.

(5) $(xy)z = x(yz)$. (associativité de la concaténation)

(6) $ux = uy \;\Rightarrow\; x = y$. (simplifiabilité à gauche)
$xu = yu \;\Rightarrow\; x = y$. (simplifiabilité à droite)

(7) $xx = x \;\Leftrightarrow\; x = \Lambda$.

2.1.10. Reste d'une séquence non vide

Si $x = \langle\!\langle\, x[1], x[2], \ldots, x[n] \,\rangle\!\rangle$ est une séquence non vide ($n \neq 0$), on appelle *reste* de x la séquence obtenue en enlevant le premier élément de x, à savoir la séquence $\langle\!\langle\, x[2], \ldots, x[n] \,\rangle\!\rangle$. Nous désignons cette séquence par la notation $x{\downarrow}$. Elle est égale à Λ si $n = 1$. Par exemple :

$$x = \langle\!\langle\, a, b, b, a \,\rangle\!\rangle = \{1 \mapsto a, 2 \mapsto b, 3 \mapsto b, 4 \mapsto a\}.$$
$$x{\downarrow} = \langle\!\langle\, b, b, a \,\rangle\!\rangle = \{1 \mapsto b, 2 \mapsto b, 3 \mapsto a\}.$$

La définition formelle de $x{\downarrow}$ est la suivante.

Définition. Si x est une séquence non vide ($x \neq \Lambda$), la séquence $x{\downarrow}$ est caractérisée par les deux formules suivantes :

(1) $\lg(x{\downarrow}) = \lg(x) - 1$.

(2) $\forall k \in [1 \,..\, \lg(x{\downarrow})] : \quad (x{\downarrow})[k] = x[k+1]$.

Théorèmes. Les formules suivantes sont vraies pour une séquence $x \neq \Lambda$ et une séquence y quelconque.

(3) $\forall k \in [2 \,..\, \lg(x)] : \quad (x{\downarrow})[k-1] = x[k]$.

(4) $\lg(x) = 1 \Leftrightarrow x{\downarrow} = \Lambda$.

(5) $\langle\!\langle\, a \,\rangle\!\rangle{\downarrow} = \Lambda$.

(6) Si $x \neq \Lambda$, alors $x = \langle\!\langle\, x[1] \,\rangle\!\rangle\,(\,x{\downarrow}\,) = \langle\!\langle\, x[1] \,\rangle\!\rangle\ \mathsf{cat}\ (\,x{\downarrow}\,)$.

(7) Si $x \neq \Lambda$, alors $(xy){\downarrow} = (x{\downarrow})y$.

(8) $(\,\langle\!\langle\, a \,\rangle\!\rangle\, y\,){\downarrow} = y$.

La démonstration de ces formules, sauf (7) (Exercice 2.1.11), est donnée en annexe [A].

2.1.11. Exercices

(a) Remplacer les deux formules 2.1.10(1) et 2.1.10(2) par une seule égalité de la forme $x{\downarrow} = \ldots$ en utilisant la notation lambda (1.3.17) pour définir la séquence $x{\downarrow}$.

(b) Démontrer l'égalité de 2.1.10(7) en supposant que $x \neq \Lambda$. Cette hypothèse entraîne $xy \neq \Lambda$ (2.1.9(4)). La démonstration consiste à appliquer la formule 2.1.10(6), d'une part à x dans le produit xy et d'autre part à (xy) elle-même. En tenant compte de ce que $(xy)[1] = x[1]$ (2.1.9(1)), on obtient une égalité que l'on peut simplifier d'après 2.1.9(6).

2.2 L'ensemble des séquences sur un ensemble X

2.2.1. Les ensembles $\mathrm{W}(X)$ et $\mathrm{W}_n(X)$

Une séquence x de longueur n est par définition (2.1.2) une fonction dont le domaine est l'intervalle $[1\,..\,n]$ de $\mathbb{N}$. Étant donné une telle fonction x, si X est un ensemble tel que l'on ait

$$\forall k \in [1\,..\,n] : x[k] \in X,$$

alors la séquence x considérée est une application de l'ensemble $[1\,..\,n]$ dans l'ensemble X (1.3.14), ce qu'on peut écrire $x : [1\,..\,n] \to X$. Dans ce cas, on dit que x est une séquence de longueur n *sur l'ensemble* X, ou encore une séquence (de longueur n) *d'éléments de* X.

Définition 1. Étant donné un entier $n \in \mathbb{N}$ et un ensemble X, nous désignons par $\mathrm{W}_n(X)$ *l'ensemble des séquences de longueur n sur* X, autrement dit l'ensemble des applications de $[1\,..\,n]$ dans X. Par définition, nous avons donc

(1) $$\begin{aligned} x \in \mathrm{W}_n(X) &\Leftrightarrow (x : [1\,..\,n] \to X) \\ &\Leftrightarrow x \text{ est une séquence de longueur } n \text{ sur } X. \end{aligned}$$

Les formules suivantes découlent de cette définition.

(2) $\mathrm{W}_0(X) = \{\Lambda\}$.

(3) $\mathrm{W}_1(X) = \{\langle\!\langle\, a \,\rangle\!\rangle \mid a \in X\}$.
$\mathrm{W}_2(X) = \{\langle\!\langle\, a, b \,\rangle\!\rangle \mid a \in X \text{ et } b \in X\}$.
$\mathrm{W}_3(X) = \{\langle\!\langle\, a, b, c \,\rangle\!\rangle \mid a \in X \text{ et } b \in X \text{ et } c \in X\}$.
etc.

(4) $n \geq 1 \Rightarrow W_n(\varnothing) = \varnothing$.

Démonstration. L'ensemble $\mathrm{W}_0(X)$ est l'ensemble des séquences de longueur 0 sur X. La formule (2) se justifie par le fait que la séquence Λ est l'unique séquence de longueur 0. Cela permet d'affirmer que $\mathrm{W}_0(X) \subset \{\Lambda\}$. Pour prouver l'implication réciproque, il s'agit de montrer que pour tout ensemble X, on peut affirmer que Λ est une séquence sur X. Cela est vrai parce que Λ a été définie (2.1.4) comme étant la fonction vide et par suite que, pour tout ensemble X, on a $\Lambda : [1\,..\,0] \to X$. Voir [A] pour plus de précisions.

Une des formules (3) est démontrée formellement dans [A]. Nous rappelons seulement que, d'après la définition 1 de 1.2.3, la proposition $x \in \{\langle\!\langle\, a \,\rangle\!\rangle \mid a \in X\}$ est équivalente à $\exists a\big(x = \langle\!\langle\, a \,\rangle\!\rangle \text{ et } a \in X\big)$. De même, la proposition $x \in \{\langle\!\langle\, a, b \,\rangle\!\rangle \mid a \in X \text{ et } b \in X\}$ est équivalente à $\exists a \exists b\big(x = \langle\!\langle\, a, b \,\rangle\!\rangle \text{ et } a \in X \text{ et } b \in X\big)$, etc.

Pour la formule (4), il suffit de montrer que les hypothèses $n \geq 1$ et $x \in \mathrm{W}_n(\varnothing)$ donnent lieu à une contradiction. Cela provient du fait général qu'il n'existe pas d'application d'un ensemble non vide, par exemple d'un intervalle $[1\,..\,n]$ tel que $n \geq 1$, dans l'ensemble $\varnothing$. Voir [A] pour plus de précisions.

Exemple. Soit $X = \{a, b\}$ un ensemble à deux éléments, que nous supposons distincts ($a \neq b$). On a

$W_0(X) = \{\Lambda\}$;

$\mathrm{W}_1(X) = \{\langle\!\langle\, a \,\rangle\!\rangle, \langle\!\langle\, b \,\rangle\!\rangle\}$;

$\mathrm{W}_2(X) = \{\langle\!\langle\, a, a \,\rangle\!\rangle, \langle\!\langle\, a, b \,\rangle\!\rangle, \langle\!\langle\, b, a \,\rangle\!\rangle, \langle\!\langle\, b, b \,\rangle\!\rangle\}$;

$\mathrm{W}_3(X) = \{\langle\!\langle\, a, a, a \,\rangle\!\rangle, \langle\!\langle\, a, a, b \,\rangle\!\rangle, \langle\!\langle\, a, b, a \,\rangle\!\rangle, \langle\!\langle\, a, b, b \,\rangle\!\rangle, \langle\!\langle\, b, a, a \,\rangle\!\rangle, \ldots\}$;

etc.

Pour tout ensemble X, la famille d'ensembles $(\mathrm{W}_n(X))_{n\in\mathbb{N}}$ est une famille d'ensembles *mutuellement disjoints*. On entend par là que, quels que soient $m, n \in \mathbb{N}$, on a

$$m \neq n \Rightarrow \mathrm{W}_m(X) \cap \mathrm{W}_n(X) = \varnothing. \tag{5}$$

En effet, la contraposée est immédiate : si $\mathrm{W}_m(X) \cap \mathrm{W}_n(X) \neq \varnothing$, alors il existe une séquence x appartenant à ces deux ensembles, donc telle que $\lg(x) = m$ et $\lg(x) = n$, d'où $m = n$.

Définition 2. Pour tout ensemble X, on désigne par $\mathrm{W}(X)$ la réunion de tous les ensembles $\mathrm{W}_n(X)$ ($n \in \mathbb{N}$). Autrement dit, on pose

$$\mathrm{W}(X) = \bigcup_{n\in\mathbb{N}} \mathrm{W}_n(X). \tag{6}$$

Cet ensemble est appelé *l'ensemble des séquences sur X*, ou l'ensemble des séquences *d'éléments de X*. Par définition de la réunion d'une famille d'ensembles (1.3.19), on a d'après (6) et (1) :

$$\begin{aligned} x \in \mathrm{W}(X) &\Leftrightarrow \exists n(n \in \mathbb{N} \text{ et } x \in \mathrm{W}_n(X)) \\ &\Leftrightarrow \exists n(n \in \mathbb{N} \text{ et } x : [1 \,..\, n] \to X). \end{aligned}$$

On en déduit l'équivalence suivante, qui caractérise l'ensemble $\mathrm{W}(X)$:

$$x \in \mathrm{W}(X) \;\Leftrightarrow\; \big(x \text{ est une séquence et } \forall k \in [1 \,..\, \lg(x)] : x[k] \in X\big) \tag{7}$$

Notation. La lettre W des expressions $\mathrm{W}(X)$, $\mathrm{W}_n(X)$ est la première lettre du mot anglais « Word » (ou de l'allemand « Wort » ou du néerlandais « Woord », bien entendu), signifiant « mot ». Une séquence sur un ensemble X est en effet appelée aussi un *mot* sur X, par analogie avec les mots au sens usuel, considérés comme des séquences de lettres.

2.2.2. L'ordre noethérien standard dans $\mathrm{W}(X)$

Pour tout ensemble X, nous allons définir une relation d'ordre noethérienne R dans l'ensemble $\mathrm{W}(X)$ que nous appellerons *l'ordre noethérien standard* dans cet ensemble. Dans la suite, lorsque nous écrirons $x \leq y$ pour deux séquences x, y sur un ensemble X, il s'agira toujours, sauf indication contraire, de la relation d'ordre standard que nous allons définir.

Définition. Étant donné un ensemble X, la relation d'ordre standard R dans $\mathrm{W}(X)$ est l'ensemble des couples (x, y) de séquences sur X tels que

$$x = y \text{ ou } \lg(x) < \lg(y).$$

Par définition, on a donc, quels que soient $x, y \in \mathrm{W}(X)$:

$$\begin{aligned} x\,R\,y &\Leftrightarrow (x, y) \in R \\ &\Leftrightarrow (x = y \text{ ou } \lg(x) < \lg(y)). \end{aligned} \tag{1}$$

C'est un simple exercice de logique propositionnelle [A] que de démontrer la réflexivité, la transitivité et l'antisymétrie de R, à savoir, pour des séquences quelconques x, y, z sur X, les formules :

$$\begin{gathered} x\,R\,x \\ (x\,R\,y \text{ et } y\,R\,z) \Rightarrow x\,R\,z \\ (x\,R\,y \text{ et } y\,R\,x) \Rightarrow x = y. \end{gathered}$$

Puisque R est une relation d'ordre dans $\mathrm{W}(X)$, on peut écrire $x \leq_R y$ au lieu de $x\,R\,y$, ou simplement $x \leq y$ lorsqu'on sait de quelle relation d'ordre R il s'agit. La proposition $x <_R y$

équivaut par définition à ($x \leq_R y$ et $x \neq y$), donc à (($x = y$ ou $\lg(x) < \lg(y)$) et $x \neq y$), d'où, en bonne logique :

$$x <_R y \Leftrightarrow \lg(x) < \lg(y). \tag{2}$$

Exemple. La figure 2.1 représente partiellement la relation d'ordre standard dans l'ensemble $\mathrm{W}(X)$, pour un ensemble $X = \{a, b\}$ composé de deux éléments a, b supposé distincts. Les « étages » de ce diagramme sont les ensembles $\mathrm{W}_0(X)$, $\mathrm{W}_1(X)$, $\mathrm{W}_2(X)$, etc. Chaque élément de l'ensemble $\mathrm{W}_{n+1}(X)$ a pour prédécesseurs immédiats, suivant cet ordre, tous les éléments de l'ensemble $\mathrm{W}_n(X)$. On a illustré ceci par les arcs (ascendants) reliants les séquences $\langle\!\langle a, a, a \rangle\!\rangle$, $\langle\!\langle a, a \rangle\!\rangle$, $\langle\!\langle a \rangle\!\rangle$ à leur prédécesseurs immédiats.

Cet exemple montre bien que l'ordre standard R dans l'ensemble $\mathrm{W}(X)$ n'est pas un ordre total lorsque l'ensemble X possède au moins deux éléments distincts. Deux séquences x, y distinctes mais de même longueur, comme par exemple $x = \langle\!\langle a, a, a \rangle\!\rangle$ et $y = \langle\!\langle a, a, b \rangle\!\rangle$ ($a \neq b$), ne sont pas comparables : on a non($x \leq y$ ou $y \leq x$).

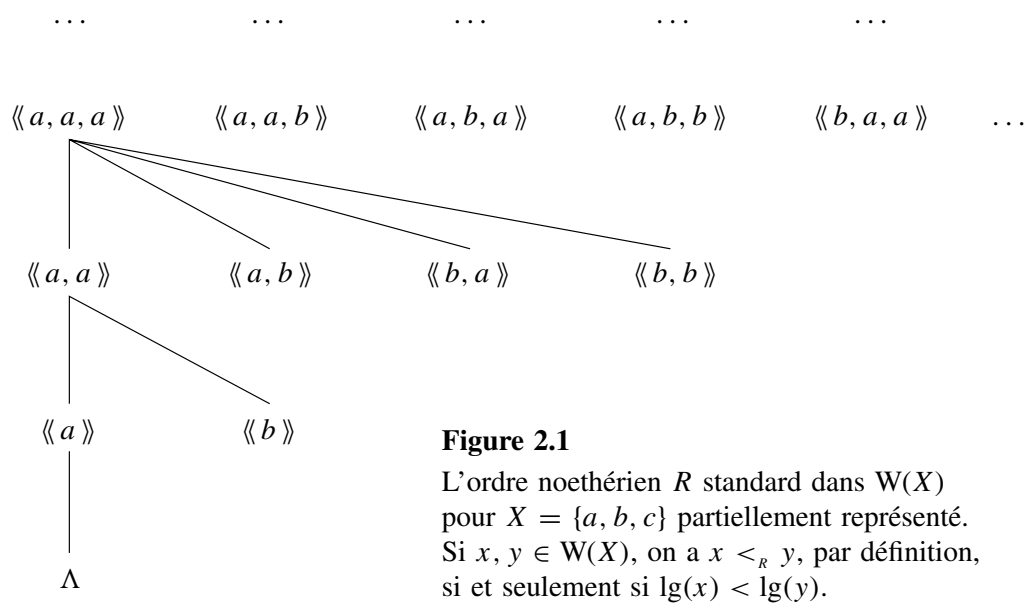


Figure 2.1
L'ordre noethérien R standard dans $\mathrm{W}(X)$ pour $X = \{a, b, c\}$ partiellement représenté. Si $x, y \in \mathrm{W}(X)$, on a $x <_R y$, par définition, si et seulement si $\lg(x) < \lg(y)$.

La formule (2) montre immédiatement que la relation R est une relation d'ordre noethérienne dans $\mathrm{W}(X)$, d'après le critère (théorème) de 1.4.7, à savoir qu'il n'existe pas de suite $(x_n)_{n \in \mathbb{N}}$ de séquences sur X strictement décroissante suivant R, c'est-à-dire telle que l'on ait

$$\forall n \in \mathbb{N} : \; x_{n+1} <_R x_n.$$

La séquence Λ est le plus petit élément de l'ensemble $\mathrm{W}(X)$ suivant la relation d'ordre R (1.4.4), en ce sens que l'on a

$$\forall x \in \mathrm{W}(X) : \quad \Lambda \leq_R x \tag{3}$$

Nous désignons d'habitude par S_x l'ensemble des minorants stricts d'un élément x d'un ensemble ordonné E (1.4.5), c'est-à-dire l'ensemble des éléments y de E tels que $y < x$. Les éléments *minimaux* de E sont par définition les éléments x tels que $S_x = \varnothing$ (1.4.6). Dans le cas de l'ensemble $E = \mathrm{W}(X)$ muni de l'ordre standard R, S_x est l'ensemble des

séquences y sur X telles que $\lg(y) < \lg(x)$. La séquence Λ est l'unique élément minimal de $\mathrm{W}(X)$. Autrement dit l'on a, quel que soit $x \in \mathrm{W}(X)$:

$$S_x = \varnothing \Leftrightarrow x = \Lambda. \tag{4}$$

Applications. L'ordre noethérien standard dans l'ensemble $\mathrm{W}(X)$ (pour un ensemble X quelconque) sera la base de la plupart de nos démonstrations par récurrence de théorèmes de la forme

$$\forall x \in \mathrm{W}(X) : P(x),$$

exprimant que toute séquence x sur X vérifie une certaine proposition $P(x)$. Nous reviendrons sur cette méthode de démonstration à l'occasion du premier exemple (2.2.4).

L'ordre noethérien standard dans $\mathrm{W}(X)$ sera aussi la base de toutes les définitions par récurrence de fonctions de domaine $\mathrm{W}(X)$. Un premier exemple est donné au §2.2.3.

Au lieu de « récurrence suivant l'ordre noethérien standard » dans $\mathrm{W}(X)$, on parle aussi de démonstrations et de définitions de fonctions *par récurrence sur la longueur des séquences*.

2.2.3. La fonction de renversement des séquences sur X

Nous donnons ici un exemple de fonction de domaine $\mathrm{W}(X)$, où X est un ensemble quelconque, définie *par récurrence suivant l'ordre noethérien standard dans* $\mathrm{W}(X)$. Il s'agit d'une fonction ρ qui est une application de l'ensemble $\mathrm{W}(X)$ dans lui-même. Cette fonction $\rho : \mathrm{W}(X) \to \mathrm{W}(X)$ est appelée la fonction de renversement des séquences sur X. Par exemple, si a, b, c sont des éléments de X, on aura :

$$\begin{gathered}\rho(\langle\!\langle\, a, b \,\rangle\!\rangle) = \langle\!\langle\, b, a \,\rangle\!\rangle;\\ \rho(\langle\!\langle\, a, b, c \,\rangle\!\rangle) = \langle\!\langle\, c, b, a \,\rangle\!\rangle;\\ \rho(\langle\!\langle\, a \,\rangle\!\rangle) = \langle\!\langle\, a \,\rangle\!\rangle;\\ \rho(\Lambda) = \Lambda.\end{gathered}$$

Ces exemples suggèrent de prendre la proposition suivante comme *définition* de cette fonction par récurrence sur la longueur des séquences :

$$\forall x \in \mathrm{W}(X) : \quad \rho(x) = \begin{cases} \Lambda & \text{si } x = \Lambda; \\ \rho(x\downarrow)\ \mathsf{cat}\ \langle\!\langle\, x[1] \,\rangle\!\rangle & \text{si } x \neq \Lambda. \end{cases} \tag{1}$$

Application du principe de définition par récurrence noethérienne. Il est facile de montrer, d'après le principe général de définition de fonction par récurrence noethérienne (1.5.7), qu'il existe une fonction ρ de domaine $\mathrm{W}(X)$ et une seule vérifiant (1). C'est ce que nous allons faire.

Tout d'abord, en vertu de 2.2.2(4), on peut récrire (1) sous la forme

$$\forall x \in \mathrm{W}(X) : \quad \rho(x) = \begin{cases} \Lambda & \text{si } S_x = \varnothing; \\ \rho(x\downarrow)\ \mathsf{cat}\ \langle\!\langle\, x[1] \,\rangle\!\rangle & \text{si } S_x \neq \varnothing. \end{cases} \tag{2}$$

Ensuite, au lieu de $\rho(x\downarrow)$, à droite de l'égalité (2), on peut écrire $\rho_{|S_x}(x\downarrow)$, où $\rho_{|S_x}$ est la restriction de la fonction ρ à l'ensemble S_x (1.3.21). En effet, pour toute séquence $x \neq \Lambda$ sur X, on a $x\downarrow \in S_x$ et par suite $F(x\downarrow) = F_{|S_x}(x\downarrow)$ pour toute fonction F de domaine $\mathrm{W}(X)$. Donc la proposition (1) peut se mettre sous la forme équivalente :

$$\forall x \in \mathrm{W}(X) : \quad \rho(x) = \begin{cases} \Lambda & \text{si } S_x = \varnothing; \\ \rho_{|S_x}(x\downarrow)\ \mathsf{cat}\ \langle\!\langle\, x[1] \,\rangle\!\rangle & \text{si } S_x \neq \varnothing. \end{cases} \tag{3}$$

Cette proposition est de la forme décrite dans le principe de définition par récurrence noethérienne (1.5.7). Les variables F et E du §1.5.7 sont remplacées ici par ρ et $\mathrm{W}(X)$. Le

terme $\mathbf{T}_1$ est Λ. Le terme $\mathbf{T}_2$ est $\rho(x{\downarrow})\ \mathsf{cat}\ \langle\!\langle\, x[1]\,\rangle\!\rangle$. On a dans (3) le terme $(\rho_{|S_x}|\rho)\mathbf{T}_2$. La variable ρ ne figure pas librement dans $\mathbf{T}_1$. Le terme $\rho_{|S_x}$ est librement substituable à ρ dans $\mathbf{T}_2$.

2.2.4. Démonstrations par récurrence noethérienne dans $\mathrm{W}(X)$. Exemple

Pour un ensemble X quelconque, nous allons démontrer par récurrence noethérienne dans $\mathrm{W}(X)$ muni de l'ordre standard (2.2.2) — on dit aussi par récurrence sur la longueur des séquences sur X — une propriété de la fonction $\rho : \mathrm{W}(X) \to \mathrm{W}(X)$, dite de renversement des séquences sur X (2.2.3). Cette propriété est la suivante: pour deux séquences x, y quelconques sur X, on a

$$\rho(x\ \mathsf{cat}\ y) = \rho(y)\ \mathsf{cat}\ \rho(x). \tag{1}$$

Cette propriété est intuitivement évidente. Nous voulons montrer qu'elle découle de la définition de la fonction ρ (2.2.3(1)). Dans une démonstration par récurrence dans $\mathrm{W}(X)$, il s'agit toujours de démontrer une proposition de la forme

$$\forall x \in \mathrm{W}(X) : P(x),$$

exprimant que toute séquence x sur X vérifie une certaine proposition $P(x)$. Un tel théorème commence toujours par un quantificateur universel $\forall x \in \mathrm{W}(X)$, et l'on parle plus précisément de le démontrer *par récurrence sur* x. Bien entendu, à la place de x, ce peut être une autre variable. Dans notre exemple, nous allons prendre pour $P(x)$ la proposition (1) et démontrer

$$\forall x \in \mathrm{W}(X) : \underbrace{\rho(x\ \mathsf{cat}\ y) = \rho(y)\ \mathsf{cat}\ \rho(x)}_{P(x)} \tag{2}$$

sous l'hypothèse que y est une séquence quelconque mais fixe sur X. De façon précise, le plan de cette démonstration sera le suivant:

$y \in \mathrm{W}(X)$ hyp
⋮
Démonstration par récurrence sur x
(le contenu de ce cadre est présenté dans la fig. 2.2)
⋮
$\forall x \in \mathrm{W}(X) : P(x)$

$y \in \mathrm{W}(X) \Rightarrow (\forall x \in \mathrm{W}(X) : P(x))$
$\forall y\big[\, y \in \mathrm{W}(X) \Rightarrow (\forall x \in \mathrm{W}(X) : P(x))\,\big]$

ce qui s'abrège:
$\forall y \in \mathrm{W}(X) : (\forall x \in \mathrm{W}(X) : P(x))$.

La proposition finale de cette démonstration, dont on peut permuter si l'on veut les quantificateurs typés $(\forall y \in \mathrm{W}(X))$, $(\forall x \in \mathrm{W}(X))$, est bien le théorème qu'il fallait démontrer, à savoir que (1) est vraie quels que soient x, $y \in \mathrm{W}(X)$.

Les trois dernières lignes de cette démonstration sont triviales et généralement omises. La partie à laquelle nous nous intéressons est le cadre intérieur, contenant la démonstration par récurrence à proprement parler. Le contenu de ce cadre, sauf l'hypothèse initiale $y \in \mathrm{W}(X)$, est présenté dans la figure 2.2.

$x \in \mathrm{W}(X)$ hyp

$x = \Lambda$	hyp
$\rho(xy) = \rho(\Lambda y)$	
$= \rho(y)$	Λ neutre pour la concaténation
$= \rho(y)\Lambda$	
$= \rho(y)\rho(\Lambda)$	définition de ρ (2.2.3(1))
$= \rho(y)\rho(x)$	
8^{o} $\underbrace{\rho(xy) = \rho(y)\rho(x)}_{P(x)}$	

$x \neq \Lambda$ et $\forall z \in S_x : \underbrace{\rho(zy) = \rho(y)\rho(z)}_{P(z)}$	hypothèse de récurrence
$\rho(xy) = \rho((xy)\downarrow)\langle\!\langle\, (xy)[1]\,\rangle\!\rangle$	définition de ρ (2.2.3(1)), $xy \neq \Lambda$
$= \rho((x\downarrow)y)\langle\!\langle\, x[1]\,\rangle\!\rangle$	2.1.10(7)
$= \rho(y)\rho(x\downarrow)\langle\!\langle\, x[1]\,\rangle\!\rangle$	hypothèse de récurrence, $x\downarrow \in S_x$
$= \rho(y)\rho(x)$	définition de ρ
14^{o} $\underbrace{\rho(xy) = \rho(y)\rho(x)}_{P(x)}$	

$\underbrace{\rho(xy) = \rho(y)\rho(x)}_{P(x)}$ 8^{o}, 14^{o}, principe de récurrence.

$\forall x \in \mathrm{W}(X) : \underbrace{\rho(xy) = \rho(y)\rho(x)}_{P(x)}$.

Figure 2.2

Démonstration de (2) par récurrence sur la longueur des séquences ou récurrence noethérienne dans $\mathrm{W}(X)$ (cf. figure 1.7, §1.5.5).

Plan de démonstration général. Le plan général d'une démonstration par récurrence sur la longueur des séquences (ou récurrence noethérienne) dans $\mathrm{W}(X)$ peut se résumer par les *directives suivantes :*

Pour démontrer une proposition de la forme $\forall x \in \mathrm{W}(X) : P(x)$ sous des hypothèses Γ dans lesquelles x ne figure pas librement, on démontre $P(x)$, pour une séquence $x \in \mathrm{W}(X)$:

(a) sous l'hypothèse $x = \Lambda$;
(b) sous l'hypothèse (dite *de récurrence*) $x \neq \Lambda$ et $\forall z \in S_x : P(z)$,

où S_x désigne l'ensemble des séquences $z \in \mathrm{W}(X)$ telles que $\lg(z) < \lg(x)$.

2.2.5. Exercice

Soient X un ensemble et a un élément de X.

(a) Définir par récurrence sur la longueur des séquences la fonction $F : \mathrm{W}(X) \to \mathrm{W}(X)$ qui « supprime les occurrences de a » dans toute séquence x sur X. Par exemple, si b et c sont des éléments de X distincts de a, on doit avoir $F(\langle\!\langle\, b, a, a, b, c, a\,\rangle\!\rangle) = \langle\!\langle\, b, b, c\,\rangle\!\rangle$. Pour

une séquence x dans laquelle il n'y a pas d'occcurrence de a, on doit avoir $F(x) = x$. Au moyen du principe de définition par récurrence noethérienne (1.5.7), prouver l'existence et l'unicité d'une fonction F de domaine $\mathrm{W}(X)$ vérifiant l'équation de récurrence prise comme définition.

(b) Montrer par récurrence sur la longueur des séquences que, quelles que soient les séquences $x, y \in \mathrm{W}(X)$, on a

$$F(x \mathbin{\mathsf{cat}} y) = F(x) \mathbin{\mathsf{cat}} F(y).$$

Plus précisément, démontrer $\forall x \in \mathrm{W}(X) : F(x \mathbin{\mathsf{cat}} y) = F(x) \mathbin{\mathsf{cat}} F(y)$ par récurrence sur x, pour une séquence y fixe quelconque (cf. 2.2.4).

(c) Définir par récurrence sur la longueur des séquences la fonction $N_a : \mathrm{W}(X) \to \mathbb{N}$ qui retourne, pour chaque séquence $x \in \mathrm{W}(X)$, le nombre d'occurrences de a dans X.

(d) Montrer par récurrence sur la longueur des séquences que la fonction F de « suppression de a » satisfait à la condition suivante :

$$\forall x \in \mathrm{W}(X) : N_a(F(x)) = 0.$$

2.2.6. Exercice

Soit X un ensemble.

(a) Définir par récurrence sur la longueur des séquences sur X la fonction $F : \mathrm{W}(X) \to \mathrm{W}(X)$ qui *supprime toutes les répétitions consécutives* d'éléments dans les séquences. Par exemple, si a, b, c sont des éléments distincts de X, on doit avoir :

$$F(\langle\!\langle\, b, b, b, a, c, c, b, a, a, a \,\rangle\!\rangle) = \langle\!\langle\, b, a, c, b, a \,\rangle\!\rangle.$$

(b) Au moyen du principe de définition par récurrence noethérienne (1.5.7), prouver l'existence et l'unicité d'une fonction F de domaine $\mathrm{W}(X)$ vérifiant l'équation de récurrence prise comme définition.

(c) Démontrer, par récurrence sur la longueur des séquences, que pour toute séquence non vide $x \in \mathrm{W}(X)$, on a $(F(x))[1] = x[1]$.

2.2.7. Démonstrations par induction structurelle dans $\mathrm{W}(X)$

Pour démontrer que toute séquence x sur un ensemble X vérifie une certaine proposition $P(x)$, on peut procéder par récurrence noethérienne (ou récurrence sur la longueur des séquences) dans $\mathrm{W}(X)$ (figure 2.2). Nous présentons ici une autre méthode dite *par induction structurelle* dans $\mathrm{W}(X)$. Nous dirons plus loin ce que ces mots signifient. La méthode repose sur le *schéma de déduction* suivant.

Schéma de déduction. Pour qu'une proposition de la forme

$$\forall x \in \mathrm{W}(X) : P(x)$$

soit vraie dans un contexte, il suffit que les trois propositions suivantes soient vraies :

(1) $$P(\Lambda);$$
(2) $$\forall a \in X : P(\langle\!\langle\, a \,\rangle\!\rangle);$$
(3) $$\forall x \in \mathrm{W}(X) : \forall y \in \mathrm{W}(X) : (P(x) \text{ et } P(y)) \Rightarrow P(x \mathbin{\mathsf{cat}} y).$$

Démonstration. Pour justifier ce schéma, nous allons montrer que si les trois propositions en question sont vraies, alors on peut démontrer $\forall x \in \mathrm{W}(X) : P(x)$ par récurrence sur la longueur des séquences. Prenons en effet ces trois propositions comme base. Nous nous

référons au plan de démonstration de la figure 2.2. Sous l'hypothèse $x = \Lambda$, on a $P(x)$ en vertu de (1). Sous l'hypothèse de récurrence

$$x \neq \Lambda \text{ et } \forall z \in S_x : P(z),$$

on démontre $P(x)$ par disjonction des cas $\lg(x) = 1$, $\lg(x) \neq 1$. Dans le premier cas on a $P(x)$ en vertu de (2). Dans le deuxième cas, on a $\lg(x) \geq 2$ donc

$$\langle\!\langle\, x[1] \,\rangle\!\rangle \in S_x \text{ et } x{\downarrow} \in S_x.$$

En vertu de l'hypothèse de récurrence, on a donc $P(\langle\!\langle\, x[1] \,\rangle\!\rangle)$ et $P(x{\downarrow})$. Comme $x = \langle\!\langle\, x[1] \,\rangle\!\rangle$ cat $x{\downarrow}$, on a $P(x)$ en vertu de (3).

***Ce que signifie* « *induction structurelle* ».** Dans une démonstration par récurrence noethérienne dans $\mathrm{W}(X)$, on considère cet ensemble comme étant muni d'une certaine relation d'ordre. Dans le schéma de déduction énoncé ci-dessus, ce n'est pas la relation d'ordre en question qui intervient, mais ce sont :

(a) certains *éléments particuliers* de l'ensemble $W(X)$, à savoir toutes les séquences sur X de longueur 0 ou 1;

(b) l'opération de concaténation dans $\mathrm{W}(X)$, une « loi » qui, à chaque couple (x, y) d'éléments de cet ensemble, fait correspondre un élément (xy) du même ensemble.

Lorsqu'on munit un ensemble E soit d'une relation d'ordre, soit d'une opération comme dans (b), soit d'éléments particuliers comme dans (a), soit d'autres choses encore, on dit que l'on munit l'ensemble E d'une certaine *structure*. Muni d'une structure, E n'est pas seulement un « amas » d'objets, mais un ensemble d'objets dans lequel on considère des relations entre éléments, des opérations sur ces éléments, des éléments distingués, des sous-ensembles particuliers, etc.

On peut munir un ensemble E de différentes espèces de structure. Une relation d'ordre en est une. Une donnée telle que (a) et (b) en est une autre, que l'on qualifie d'algébrique. Le terme *induction structurelle*, dans le titre de ce paragraphe, est en fait incomplet, car il sous-entend que la structure considérée n'est pas celle d'ordre mais celle qui est constituée par (a) et (b). Le terme complet serait *induction structurelle dans le monoïde* $\mathrm{W}(X)$ (2.3).

Le genre de structure que l'on considère dans un ensemble E lorsqu'on parle d'induction structurelle dans E se compose toujours de données du type (a), c'est-à-dire d'éléments particuliers de E, et du type (b), c'est-à-dire d'opérations dans E. Ces données doivent avoir la propriété essentielle suivante : on doit pouvoir engendrer tout élément de E à partir d'éléments du sous-ensemble (a), en effectuant un nombre fini d'opérations (b). Dans le cas de l'ensemble $\mathrm{W}(X)$, toute séquence $x \in \mathrm{W}(X)$ peut s'obtenir en prenant des séquences de longueur 0 ou 1 et en effectuant un nombre fini de produits (concaténations).

Les propositions (1) et (2) expriment que tous les objets du sous-ensemble (a) ont la propriété P. La proposition (3) exprime que l'opération (b) *conserve* la propriété P, en ce sens que si on l'applique à des objets qui vérifient P, on obtient un objet qui vérifie P. Comme tout élément de l'ensemble $E = \mathrm{W}(X)$ peut être obtenu à partir d'objets (a) par des opérations (b), tout élément de E vérifie P. C'est cette déduction qu'exprime le schéma de ce paragraphe. C'est aussi le même genre de déduction qu'exprimeront d'autres « schémas d'induction structurelle », dans d'autres espèces d'ensembles que nous verrons plus loin.

2.2.8. Exemple

Nous considérons la fonction $F : \mathrm{W}(X) \to \mathrm{W}(X)$ de l'exercice 2.2.5, qui « supprime les occurrences de a » dans les séquences sur X, X étant un ensemble quelconque et a un

élément de X. Nous nous référons à la solution de cet exercice 7, dans laquelle F est définie par récurrence sur la longueur des séquences en posant, pour tout $x \in \mathrm{W}(X)$:

$$F(x) = \begin{cases} \Lambda & \text{si } x = \Lambda; \\ F(x\downarrow) & \text{si } x \neq \Lambda \text{ et } x[1] = a; \\ \langle\!\langle\, x[1] \,\rangle\!\rangle \ \mathsf{cat}\ F(x\downarrow) & \text{si } x \neq \Lambda \text{ et } x[1] \neq a. \end{cases}$$

Nous basant sur cette définition, nous allons démontrer

$$\forall x \in \mathrm{W}(X) : \quad \underbrace{F(F(x)) = F(x)}_{P(x)}.$$

par induction structurelle dans $\mathrm{W}(X)$ (2.2.7).

(1) Démonstration de $P(\Lambda)$.
Par définition de F, on a $F(\Lambda) = \Lambda$, donc $F(F(\Lambda)) = F(\Lambda)$.

(2) Démonstration de $\forall b \in X : P(\langle\!\langle\, b \,\rangle\!\rangle)$.
Soit b un élément quelconque de X. Nous démontrons $P(\langle\!\langle\, b \,\rangle\!\rangle)$ par disjonction des cas $b = a$, $b \neq a$.
Dans le premier cas, on a $F(\langle\!\langle\, b \,\rangle\!\rangle) = F(\langle\!\langle\, b \,\rangle\!\rangle\downarrow)$ puisque $\langle\!\langle\, b \,\rangle\!\rangle[1] = a$, donc $F(\langle\!\langle\, b \,\rangle\!\rangle) = F(\Lambda) = \Lambda$ et par suite $F(F(\langle\!\langle\, b \,\rangle\!\rangle)) = F(\Lambda) = F(\langle\!\langle\, b \,\rangle\!\rangle)$.
Dans le deuxième cas, on a $F(\langle\!\langle\, b \,\rangle\!\rangle) = \langle\!\langle\, b \,\rangle\!\rangle \ \mathsf{cat}\ F(\langle\!\langle\, b \,\rangle\!\rangle\downarrow)$ puisque $\langle\!\langle\, b \,\rangle\!\rangle[1] \neq a$, donc $F(\langle\!\langle\, b \,\rangle\!\rangle) = \langle\!\langle\, b \,\rangle\!\rangle \ \mathsf{cat}\ \Lambda = \langle\!\langle\, b \,\rangle\!\rangle$ et par suite $F(F(\langle\!\langle\, b \,\rangle\!\rangle)) = F(\langle\!\langle\, b \,\rangle\!\rangle)$.

(3) Démonstration de $(P(x)$ et $P(y)) \Rightarrow P(x \ \mathsf{cat}\ y)$
pour des séquences $x, y \in \mathrm{W}(X)$ quelconques.
Supposons que $F(F(x)) = F(x)$ et $F(F(y)) = F(y)$. Il s'agit d'en déduire

$$F(F(x \ \mathsf{cat}\ y)) = F(x \ \mathsf{cat}\ y).$$

Pour cela nous utilisons la formule $F(x \ \mathsf{cat}\ y) = F(x) \ \mathsf{cat}\ F(y)$, démontrée dans l'exercice 2.2.5, vraie quelles que soient $x, y \in \mathrm{W}(X)$. D'après cette formule et en vertu de notre hypothèse, nous avons :

$$\begin{aligned} F\big(F(x \ \mathsf{cat}\ y)\big) &= F\big(F(x) \ \mathsf{cat}\ F(y)\big) \\ &= F\big(F(x)\big) \ \mathsf{cat}\ F\big(F(y)\big) \\ &= F(x) \ \mathsf{cat}\ F(y) \\ &= F(x \ \mathsf{cat}\ y). \end{aligned}$$

2.3 Le monoïde $\mathrm{W}(X)$

2.3.1. Homomorphismes du monoïde $\mathrm{W}(X)$ dans un monoïde M

Soit X un ensemble quelconque. L'ensemble $\mathrm{W}(X)$ des séquences sur X, muni de l'opération de concaténation (associative) et de la séquence Λ (neutre pour cette opération), autrement dit le triplet $(\mathrm{W}(X), \mathsf{cat}, \Lambda)$, est un monoïde (1.6), le monoïde des séquences sur X. Considérons ce monoïde et un deuxième monoïde $(M, *, e)$ quelconque, qui peut être en particulier le monoïde $(\mathrm{W}(X), \mathsf{cat}, \Lambda)$ lui-même. Une fonction $F : \mathrm{W}(X) \to M$ est par définition (1.6.9) un homorphisme du monoïde $(\mathrm{W}(X), \mathsf{cat}, \Lambda)$ dans le monoïde $(M, *, e)$ si l'on a

(1) $F(\Lambda) = e$;
(2) Quelles que soient les séquences $x, y \in \mathrm{W}(X)$: $F(x \ \mathsf{cat}\ y) = F(x) * F(y)$.

Nous avons vu au §1.6.9 l'exemple de la fonction « longueur » $F : \mathrm{W}(X) \to \mathbb{N}$ ($F(x) = \mathrm{lg}(x)$) qui est un homomorphisme du monoïde $(\mathrm{W}(X), \mathsf{cat}, \Lambda)$ dans le monoïde $(\mathbb{N}, +, 0)$.

Le théorème suivant fournit une caractérisation des homomorphismes du monoïde $(\mathrm{W}(X), \mathsf{cat}, \Lambda)$ dans un monoïde $(M, *, e)$ quelconque, dans laquelle la condition (2) est en quelque sorte simplifiée. Ce critère permet souvent de reconnaître immédiatement qu'une fonction donnée $F : \mathrm{W}(X) \to M$, définie par récurrence, est un homomorphisme, donc qu'elle vérifie la condition (2).

Théorème. Soient X un ensemble et $(M, *, e)$ un monoïde. Pour qu'une fonction $F : \mathrm{W}(X) \to M$ soit un homomorphisme du monoïde $(\mathrm{W}(X), \mathsf{cat}, \Lambda)$ dans $(M, *, e)$, il faut et il suffit que, pour tout tout $x \in \mathrm{W}(X)$, on ait

$$F(x) = \begin{cases} e & \text{si } x = \Lambda; \\ F(\langle\!\langle\, x[1] \,\rangle\!\rangle) * F(x\downarrow) & \text{si } x \neq \Lambda. \end{cases} \tag{3}$$

Démonstration : voir [A].

2.3.2. Exercice

On considère la fonction $F : \mathrm{W}(X) \to \mathrm{W}(X)$ de l'exercice 2.2.6, qui « supprime toutes les répétitions consécutives d'éléments » dans les séquences. On suppose $X \neq \varnothing$. Montrer que cette fonction n'est pas un homomorphisme du monoïde $(\mathrm{W}(X), \mathsf{cat}, \Lambda)$ dans lui-même. Définir une autre opération de concaténation cat' dans $\mathrm{W}(X)$, telle que $(\mathrm{W}(X), \mathsf{cat}', \Lambda)$ soit un monoïde et F soit un homomorphisme de $(\mathrm{W}(X), \mathsf{cat}, \Lambda)$ dans $(\mathrm{W}(X), \mathsf{cat}', \Lambda)$. Démontrer cette propriété de F en utilisant la définition de l'opération cat', le théorème du §2.3.1 appliqué au monoïde $(M, *, e) = (\mathrm{W}(X), \mathsf{cat}', \Lambda)$ et la propriété de F démontrée dans l'exercice 2.2.6 : $(F(x))[1] = x[1]$ pour toute séquence $x \neq \Lambda$.

2.3.3. L'homomorphisme $\mathrm{w}_f : \mathrm{W}(X) \to \mathrm{W}(Y)$ associé à une fonction $f : X \to Y$

Soit f une application d'un ensemble X dans un ensemble Y. Si $x = \langle\!\langle\, x[1], \ldots, x[n] \,\rangle\!\rangle$ est une séquence sur X, chacun des $x[i]$ appartient à X et son image $f(x[i])$ appartient à Y. On peut donc associer à toute séquence $x = \langle\!\langle\, x[1], \ldots, x[n] \,\rangle\!\rangle \in \mathrm{W}(X)$ la séquence $y = \langle\!\langle\, f(x[1]), \ldots, f(x[n]) \,\rangle\!\rangle \in \mathrm{W}(Y)$ que l'on peut considérer comme étant la « traduction » dans $\mathrm{W}(Y)$ de la séquence x au moyen de la fonction f. En fait, la séquence y est l'image de la séquence x par une nouvelle fonction

$$\mathrm{w}_f : \mathrm{W}(X) \to \mathrm{W}(Y),$$

associée à f. Cette fonction w_f peut être définie en posant

$$\forall x \in \mathrm{W}(X) : \quad \mathrm{w}_f(x) = \begin{cases} \Lambda & \text{si } x = \Lambda; \\ \langle\!\langle\, f(x[1]) \,\rangle\!\rangle \ \mathsf{cat}\ \mathrm{w}_f(x\downarrow) & \text{si } x \neq \Lambda. \end{cases} \tag{1}$$

Cette définition entraîne que $\mathrm{w}_f(\langle\!\langle\, a \,\rangle\!\rangle) = \langle\!\langle\, f(a) \,\rangle\!\rangle$ pour tout $a \in X$. Par suite, pour toute séquence $x \neq \Lambda$ sur X, on a $\mathrm{w}_f(x) = \mathrm{w}_f(\langle\!\langle\, x[1] \,\rangle\!\rangle)\ \mathsf{cat}\ \mathrm{w}_f(x\downarrow)$. On en déduit, d'après le théorème de 2.3.1, que w_f est un homomorphisme du monoïde $(\mathrm{W}(X), \mathsf{cat}, \Lambda)$ dans le monoïde $(\mathrm{W}(Y), \mathsf{cat}, \Lambda)$.

2.3.4. L'homomorphisme $eval_M : \mathrm{W}(M) \to M$ d'un monoïde M

Exemple. Considérons le monoïde multiplicatif $(\mathbb{N}, *, 1)$, où nous utilisons le signe $*$ comme signe d'opération pour la multiplication usuelle dans $\mathbb{N}$. Nous considérons d'autre part le monoïde $(\mathrm{W}(\mathbb{N}), \mathsf{cat}, \Lambda)$ des séquences d'entiers naturels. À toute *séquence de nombres*

$$x = \langle\!\langle\, x[1], \ldots, x[n] \,\rangle\!\rangle \in \mathrm{W}(\mathbb{N}),$$

on peut associer le *nombre*

$$x[1] * x[2] * \cdots * x[n] \ \in \mathbb{N}$$

qui est le produit (multiplication usuelle) des nombres $x[i]$. On définit de cette manière une application de l'ensemble $\mathrm{W}(\mathbb{N})$ dans l'ensemble $\mathbb{N}$. Pour préciser cette application, nous convenons que l'image de la séquence vide $\Lambda \in \mathrm{W}(\mathbb{N})$ sera le nombre $1 \in \mathbb{N}$ ([5]). Cette application de $\mathrm{W}(\mathbb{N})$ dans $\mathbb{N}$ est appelée la *fonction d'évaluation* des séquences d'éléments du monoïde multiplicatif $\mathbb{N}$. On « évalue » une séquence d'éléments de $\mathbb{N}$ en faisant le produit de ses éléments. Cette fonction est notée

$$eval_{\mathbb{N}} : \mathrm{W}(\mathbb{N}) \to \mathbb{N}.$$

Elle possède la propriété suivante, qui peut être prise comme étant sa *définition par récurrence*. Pour tout séquence $x \in \mathrm{W}(\mathbb{N})$:

$$eval_{\mathbb{N}}(x) = \begin{cases} 1 & \text{si } x = \Lambda; \\ x[1] * eval_{\mathbb{N}}(x\downarrow) & \text{si } x \neq \Lambda. \end{cases} \tag{1}$$

La notation $eval_{\mathbb{N}}$ pour cette fonction est en fait ambiguë, car au lieu du monoïde multiplicatif $(\mathbb{N}, *, 1)$ nous aurions pu considérer le monoïde additif $(\mathbb{N}, +, 0)$ et alors l'évaluation d'une séquence d'entiers $\langle\!\langle\, x[1], \ldots, x[n] \,\rangle\!\rangle$ serait la somme $x[1] + \cdots + x[n]$ et l'évaluation de la séquence vide serait le nombre 0. Pour lever toute ambiguïté, on devrait indicer le mot *eval* avec le triplet $(\mathbb{N}, *, 1)$ ou le triplet $(\mathbb{N}, +, 0)$, afin de distinguer les deux fonctions

$$eval_{(\mathbb{N},*,1)} : \mathrm{W}(\mathbb{N}) \to \mathbb{N},$$
$$eval_{(\mathbb{N},+,0)} : \mathrm{W}(\mathbb{N}) \to \mathbb{N}.$$

Nous allons maintenant généraliser cette notion pour un monoïde $(M, *, e)$ quelconque et le monoïde $\mathrm{W}(M)$ des séquences d'éléments de M, c'est-à-dire définir la fonction $eval_{(M,*,e)} : \mathrm{W}(M) \to M$. Pour alléger l'écriture, nous la noterons simplement $eval_M$.

Définition. Soit $(M, *, e)$ un monoïde. On définit la fonction

$$eval_M : \mathrm{W}(M) \to M$$

en posant, pour toute séquence $x \in \mathrm{W}(M)$:

$$eval_M(x) = \begin{cases} e & \text{si } x = \Lambda; \\ x[1] * eval_M(x\downarrow) & \text{si } x \neq \Lambda. \end{cases} \tag{2}$$

[Le principe de définition par récurrence noethérienne (§1.5.7), appliqué dans l'ensemble $\mathrm{W}(M)$ muni de l'ordre noethérien standard (2.2.2), permet de reconnaître facilement l'existence et l'unicité d'une fonction $eval_M$ de domaine $\mathrm{W}(M)$ vérifiant (2).]

Les formules suivantes découlent de (2).

(3) $\forall a \in M : \quad eval_M(\langle\!\langle\, a \,\rangle\!\rangle) = a.$

(4) $\forall x, y \in \mathrm{W}(M) : \quad eval_M(x \ \mathtt{cat} \ y) = eval_M(x) * eval_M(y).$

En vertu de l'égalité $eval_M(\Lambda) = e$ tirée de (2) et de la formule (4), la fonction $eval_M$ est un homorphisme du monoïde $(\mathrm{W}(M), \mathtt{cat}, \Lambda)$ dans le monoïde $(M, *, e)$.

---

[5] Si le lecteur est surpris par cette convention, nous lui rappelons que pour tout nombre x, on a $x^0 = 1$ et que x^n (pour $n \in \mathbb{N}$) peut être vu comme le produit de n facteurs égaux à x. Selon cette vision, un produit de zéro facteurs est égal à 1.

Démonstration. Pour (3) il suffit d'appliquer la formule (2) à la séquence $\langle\!\langle\, a \,\rangle\!\rangle$ pour un $a \in M$:

$$\begin{aligned} eval_M(\langle\!\langle\, a \,\rangle\!\rangle) &= \langle\!\langle\, a \,\rangle\!\rangle[1] * eval_M(\langle\!\langle\, a \,\rangle\!\rangle\downarrow) \\ &= a * eval_M(\Lambda) \qquad\qquad 2.1.6(1),\ 2.1.10(5) \\ &= a * e = a. \end{aligned}$$

En vertu de (3), on déduit de (2) que pour toute séquence $x \in \mathrm{W}(M)$, on a

$$eval_M(x) = \begin{cases} e & \text{si } x = \Lambda; \\ eval_M(\langle\!\langle\, x[1] \,\rangle\!\rangle) * eval_M(x\downarrow) & \text{si } x \neq \Lambda. \end{cases}$$

Le théorème du §2.3.1 permet d'en déduire que la fonction $eval_M$ est un homomorphisme du monoïde $(\mathrm{W}(M), \mathsf{cat}, \Lambda)$ dans $(M, *, e)$. D'où (4).

2.3.5. Exemple. Evaluation de séquences de séquences

Ce qui vient d'être dit (2.3.4) sur la fonction $eval_M : \mathrm{W}(M) \to M$ associée à un monoïde quelconque $(M, *, e)$ peut s'appliquer naturellement en prenant pour $(M, *, e)$ le monoïde $(\mathrm{W}(X), \mathsf{cat}, \Lambda)$ des séquences sur un ensemble X. La fonction

$$eval_{\mathrm{W}(X)} : \mathrm{W}(\mathrm{W}(X)) \to \mathrm{W}(X)$$

a pour domaine l'ensemble $\mathrm{W}(\mathrm{W}(X))$, à savoir l'ensemble des *séquences de séquences sur* X. Par exemple, si a, b, c sont des éléments de X, la séquence

$$x = \langle\!\langle\, \langle\!\langle\, a, b \,\rangle\!\rangle, \Lambda, \langle\!\langle\, b, c, c \,\rangle\!\rangle, \langle\!\langle\, a \,\rangle\!\rangle \,\rangle\!\rangle. \tag{1}$$

est une séquence de séquences sur X. Elle appartient à l'ensemble $\mathrm{W}(\mathrm{W}(X))$ et plus précisément à $\mathrm{W}_4(\mathrm{W}(X))$, avec

$$x[1] = \langle\!\langle\, a, b \,\rangle\!\rangle, \quad x[2] = \Lambda, \quad x[3] = \langle\!\langle\, b, c, c \,\rangle\!\rangle, \quad x[4] = \langle\!\langle\, a \,\rangle\!\rangle.$$

On peut écrire, en particulier : $(x[1])[1] = a$, $(x[1])[2] = b$. De façon générale, dans une séquence

$$x = \langle\!\langle\, x[1], \ldots, x[n] \,\rangle\!\rangle \in \mathrm{W}(\mathrm{W}(X)),$$

chaque $x[i]$ est lui-même une séquence sur X, $x[i] \in \mathrm{W}(X)$, et l'on peut former la concaténation des $x[i]$:

$$eval_{\mathrm{W}(X)}(x) = x[1] \mathsf{\ cat\ } \cdots \mathsf{\ cat\ } x[n].$$

Par exemple, pour la séquence (1), on a

$$\begin{aligned} eval_{\mathrm{W}(X)}(x) &= x[1] \mathsf{\ cat\ } x[2] \mathsf{\ cat\ } x[3] \mathsf{\ cat\ } x[4] \\ &= \langle\!\langle\, a, b \,\rangle\!\rangle \mathsf{\ cat\ } \Lambda \mathsf{\ cat\ } \langle\!\langle\, b, c, c \,\rangle\!\rangle \mathsf{\ cat\ } \langle\!\langle\, a \,\rangle\!\rangle \\ &= \langle\!\langle\, a, b, b, c, c, a \,\rangle\!\rangle. \end{aligned}$$

Les formules (2), (3), (4) du §2.3.4 prennent dans ce cas particulier la forme suivante :

(2) Pour toute séquence $x \in \mathrm{W}(\mathrm{W}(X))$:

$$eval_{\mathrm{W}(X)}(x) = \begin{cases} \Lambda & \text{si } x = \Lambda; \\ x[1] \mathsf{\ cat\ } eval_{\mathrm{W}(X)}(x\downarrow) & \text{si } x \neq \Lambda. \end{cases}$$

(3) $\forall u \in \mathrm{W}(X) : \quad eval_{\mathrm{W}(X)}(\langle\!\langle\, u \,\rangle\!\rangle) = u.$

(4) $\forall x, y \in \mathrm{W}(\mathrm{W}(X)) : \quad eval_{\mathrm{W}(X)}(x \mathsf{\ cat\ } y) = eval_{\mathrm{W}(X)}(x) \mathsf{\ cat\ } eval_{\mathrm{W}(X)}(y).$

2.3.6. Notation

Si $x = \langle\!\langle\, x[1], \ldots, x[n] \,\rangle\!\rangle$ est une séquence d'éléments d'un monoïde $(M, *, e)$, la séquence $eval_M(x) = x[1] * \cdots * x[n]$ est notée souvent

$$eval_M(x) \;=\; \boxed{*}_{i \in [1\,..\,n]} x[i]$$

ou encore

$$eval_M(x) \;=\; \overset{n}{\boxed{*}}_{i=1} x[i].$$

Avec cette notation, la formule 2.3.4(2) s'écrit

$$\overset{\lg(x)}{\underset{i=1}{\boxed{*}}} x[i] = \begin{cases} e & \text{si } \lg(x) = 0; \\ x[1] * \overset{\lg(x)-1}{\underset{i=1}{\boxed{*}}} x[i+1] & \text{si } \lg(x) \neq 0. \end{cases}$$

À la place du signe $\boxed{*}$ on emploie dans chaque cas particulier de monoïde M un signe spécial correspondant à celui qui est utilisé pour l'opération binaire de M. Pour un monoïde additif $(+)$ c'est naturellement le signe $\sum$. Pour un monoïde multiplicatif $(\cdot)$, le signe $\prod$. Dans le cas où $(M, *, e)$ est le monoïde $(\mathrm{W}(X), \mathsf{cat}, \Lambda)$ (2.3.5), on peut écrire, pour une séquence $x \in \mathrm{W}(\mathrm{W}(X))$ de longueur n :

$$eval_{\mathrm{W}(X)}(x) = \overset{n}{\underset{i=1}{\mathsf{CAT}}} x[i].$$

2.3.7. Évaluation d'une séquence constante

Une séquence x de longueur n étant par définition une fonction de domaine $[1\,..\,n]$ (2.1.2), elle est dite *constante* si elle est constante en tant que fonction (1.3.16). Cela signifie que x est de la forme

$$x = \langle\!\langle \underbrace{a, \ldots, a}_{n} \rangle\!\rangle,$$

avec $x = \Lambda$ si $n = 0$. Si a est élément d'un monoïde $(M, *, e)$, on a

(1) $$eval_M(x) = \underbrace{a * \cdots * a}_{n} = \mathrm{it}_a(n) \qquad (1.6.5).$$

En utilisant la notation lambda des fonctions (1.3.17), la séquence constante x peut être représentée par l'expression $(\lambda k \in [1\,..\,n] : a)$. L'énoncé précis et complet de la propriété (1) est donc le théorème suivant.

Théorème. Soient $(M, *, e)$ un monoïde et $a \in M$. Pour tout entier $n \in \mathbb{N}$, on a

(2) $$\mathrm{it}_a(n) = eval_M\big((\lambda k \in [1\,..\,n] : a)\big).$$

2.3.8. Exercice

Démontrer la formule 2.3.7(2) par récurrence naturelle sur n, en utilisant les définitions récursives des fonctions it_a (1.6.5) et $eval_M$ (2.3.4). Pour abréger, on posera

$$s_n(a) = (\lambda k \in [1\,..\,n] : a)$$

et l'on utilisera les propriétés suivantes : $s_0(a) = \Lambda$ et si $n \neq 0$, alors $(s_n(a))[1] = a$ et $(s_n(a)){\downarrow} = s_{n-1}(a)$.

2.3.9. Produit de composition d'une séquence de relations

Nous considérons dans ce paragraphe le monoïde $\mathcal{R}(E)$ des relations dans un ensemble E (1.6.3). L'élément neutre de ce monoïde est la relation identique Id_E. L'opération du monoïde est le produit de composition

$$RS = S \circ R = \{(a, b) \mid \exists x (a\, R\, x \text{ et } x\, S\, b)\}. \tag{1}$$

L'image d'une séquence de relations dans E, c'est-à-dire d'une séquence

$$R = \langle\!\langle\, R[1], \ldots, R[n] \,\rangle\!\rangle \in \mathrm{W}(\mathcal{R}(E)),$$

où chaque $R[i]$ est une relation dans E, par la fonction $eval_{\mathcal{R}(E)} : \mathrm{W}(\mathcal{R}(E)) \to \mathcal{R}(E)$ est le produit de composition des relations $R[1], \ldots, R[n]$ dans cet ordre :

$$eval_{\mathcal{R}(E)}(R) = R[1] \cdots R[n] = \prod_{i=1}^{n} R[i],$$

avec $eval_{\mathcal{R}(E)}(R) = \mathrm{Id}_E$ (élément neutre de $\mathcal{R}(E)$) si $n = 0$.

Plus précisément, la fonction $eval_{\mathcal{R}(E)}$ est définie par récurrence, conformément à 2.3.4(2), par la formule

$$\forall R \in \mathrm{W}(\mathcal{R}(E)) : \quad eval_{\mathcal{R}(E)}(R) = \begin{cases} \mathrm{Id}_E & \text{si } R = \Lambda; \\ R[1] \cdot eval_{\mathcal{R}(E)}(R{\downarrow}) & \text{si } R \neq \Lambda. \end{cases} \tag{2}$$

Le produit $R[1] \cdot eval_{\mathcal{R}(E)}(R{\downarrow})$ est un produit de composition de relations.

Le but principal de ce paragraphe est de présenter le théorème qui caractérise la relation $eval_{\mathcal{R}(E)}(R)$ en tant qu'ensemble de couples. Ce théorème sera donné sans démonstration. Pour faciliter sa compréhension et le rendre intuitivement évident, nous l'introduisons par un exemple.

Exemple. Considérons une séquence de relations dans E, de longueur 3 pour fixer les idées, $R = \langle\!\langle\, R[1], R[2], R[3] \,\rangle\!\rangle$. La définition du produit de composition (1) permet de montrer facilement qu'un couple (a, b) appartient à la relation composée $R[1]R[2]R[3]$, autrement dit à la relation $eval_{\mathcal{R}(E)}(R)$, si et seulement s'il existe des éléments u et u' de E tels que l'on ait

$$(a, u) \in R[1]; \quad (u, u') \in R[2]; \quad (u', b) \in R[3].$$

Pour de tels éléments u, u' de E, la séquence

$$x = \langle\!\langle\, a, u, u', b \,\rangle\!\rangle$$

a les propriétés suivantes :

$\lg(x) = 4 = \lg(R) + 1$;
$x[1] = a$;
$x[4] = x[\lg(x)] = b$;
$(x[1], x[2]) \in R[1]$ et $(x[2], x[3]) \in R[2]$ et $(x[3], x[4]) \in R[3]$.

La dernière de ces propriétés peut s'exprimer aussi

$$\forall k \in [1 \,..\, \lg(R)] : (x[k], x[k + 1]) \in R[k].$$

Inversement, s'il existe une séquence $x \in \mathrm{W}(E)$ avec ces quatre propriétés, on peut en déduire que le couple (a, b) appartient à la relation $R[1]R[2]R[3]$. Le théorème qui suit est la généralisation de cet exemple pour une séquence R de longueur quelconque.

2.3.10. Théorème
Soient E un ensemble, $n \in \mathbb{N}$ et $R = \langle\!\langle\, R[1], \ldots, R[n] \,\rangle\!\rangle$ une séquence de relations dans E. Pour que

$$(a, b) \in \prod_{i=1}^{n} R[i] = eval_{\mathcal{R}(E)}(R),$$

il faut et il suffit qu'il existe une séquence d'éléments de E de longueur $n + 1$

$$x = \langle\!\langle\, x[1], \ldots, x[n+1] \,\rangle\!\rangle$$

telle que :

$$(1) \qquad \begin{cases} x[1] = a; \\ x[n+1] = b; \\ \forall k \in [1 \,..\, n] : \ \big(x[k], x[k+1]\big) \in R[k]. \end{cases}$$

Chapitre 3 Langages

3.1 Langages, produits, réunions

3.1.1. Alphabets

Un alphabet est un ensemble de symboles. Ceci n'est pas une définition mathématique, parce que le concept de « symbole » n'est pas un concept mathématique. Théoriquement, n'importe quel ensemble X peut être appelé un « alphabet », et l'on peut convenir d'appeler « symboles » les éléments de X. On utilise cette terminologie lorsqu'on s'intéresse aux séquences d'éléments de X sans s'intéresser à la nature particulière de ces éléments, ni à leur structure interne s'ils en ont une, ni aux propriétés qui découlent de cette structure. On ne fait que se servir des éléments de X pour former des séquences, que l'on appelle alors souvent des *mots* sur X. L'ensemble X est donc pris simplement comme répertoire de « lettres » pour composer des mots. On utilise souvent, dans ce cas, la notation

$$x[1]x[2]\cdots x[n]$$

pour une séquence $x = \langle\!\langle\, x[1], x[2], \ldots, x[n] \,\rangle\!\rangle$ de longueur n sur X. Par exemple, si `a`, `b`, `c` sont des éléments de X, on écrit `baabbc` au lieu de $\langle\!\langle$ `b`, `a`, `a`, `b`, `b`, `c` $\rangle\!\rangle$. La séquence élémentaire $\langle\!\langle$ `a` $\rangle\!\rangle$ est notée alors `a`. Cette notation abusive est couramment utilisée.

3.1.2. Langages

On appelle *langage sur un alphabet* X (ou simplement sur un *ensemble* X) toute partie (ou sous-ensemble) de l'ensemble $\mathrm{W}(X)$, celui-ci étant l'ensemble des séquences (ou mots) sur X (2.2.1). Autrement dit, par définition :

$$\begin{aligned} L \text{ est un langage sur } X &\Leftrightarrow L \subset \mathrm{W}(X) \\ &\Leftrightarrow L \in \mathcal{P}(\mathrm{W}(X)). \end{aligned}$$

On rappelle que $\mathcal{P}(E)$ désigne l'ensemble des parties d'un ensemble E, à savoir que $A \in \mathcal{P}(E)$ équivaut à $A \subset E$.

Exemple 1. L'ensemble $\varnothing$ est un langage sur X, puisque $\varnothing \subset \mathrm{W}(X)$.

Exemple 2. L'ensemble $\mathrm{W}(X)$ lui-même est un langage sur X, puisque $\mathrm{W}(X) \subset \mathrm{W}(X)$. Les langages $\varnothing$ et $\mathrm{W}(X)$ sont respectivement le plus petit et le plus grand langage sur X, en ce sens que, pour tout langage L sur X, on a $\varnothing \subset L \subset \mathrm{W}(X)$.

Exemple 3. L'ensemble $\{\Lambda\}$ est un langage sur X, car on a $\{\Lambda\} = \mathrm{W}_0(X) \subset \mathrm{W}(X)$ (2.2.1). Nous appellerons l'ensemble $\{\Lambda\}$ le *langage neutre* sur X, car nous définirons plus loin une opération de produit de langages pour laquelle ce langage est effectivement neutre. Il ne faut pas confondre ce langage $\{\Lambda\}$ avec le langage $\varnothing$ (Exemple 1). On a $\{\Lambda\} \neq \varnothing$, puisque $\Lambda \in \{\Lambda\}$ (théorie des ensembles).

Exemple 4. Si a est un élément quelconque de X, l'ensemble $\{\langle\!\langle\, a \,\rangle\!\rangle\}$ est un langage sur X. Nous dirons qu'un tel langage est un *langage élémentaire* sur X. Il se compose d'un seul mot, celui-ci étant une séquence élémentaire (2.1.6) sur X. Il faut bien distinguer les trois objets

$$a \in X, \qquad \langle\!\langle\, a \,\rangle\!\rangle \in \mathrm{W}(X), \qquad \{\langle\!\langle\, a \,\rangle\!\rangle\} \subset \mathrm{W}(X).$$

C'est néanmoins un abus de notation fréquent que de désigner les trois objets par a. Le contexte indique lequel des trois est désigné ainsi.

Exemple 5. Pour tout $n \in \mathbb{N}$, l'ensemble des mots de longueur n sur X, noté $\mathrm{W}_n(X)$ (2.2.1), est évidemment un langage sur X. Nous avons vu (2.2.1(6)) que l'ensemble $\mathrm{W}(X)$ est la réunion des langages de cette forme sur X.

Exemple 6. L'ensemble

$$L = \bigcup_{n \in \mathbb{N}} \mathrm{W}_{2n}(X)$$

est une réunion de langages sur X, donc un langage sur X. Ce langage L est l'ensemble des mots de longueur paire sur X. En effet, par définition de la réunion d'une famille d'ensembles, on a pour ce langage L :

$$\begin{aligned} x \in L &\Leftrightarrow \exists n (n \in \mathbb{N} \text{ et } x \in \mathrm{W}_{2n}(X)) \\ &\Leftrightarrow \exists n (n \in \mathbb{N} \text{ et } x \in \mathrm{W}(X) \text{ et } \lg(x) = 2n) \\ &\Leftrightarrow x \in \mathrm{W}(X) \text{ et } \exists n (n \in \mathbb{N} \text{ et } \lg(x) = 2n) \\ &\Leftrightarrow x \text{ est une séquence de longueur paire sur } X. \end{aligned}$$

Exemple 7. Pour toute propriété $P(x)$ que l'on peut formuler à propos d'une séquence x, l'expression $\{x \mid x \in \mathrm{W}(X) \text{ et } P(x)\}$, couramment abrégée

$$\{x \in \mathrm{W}(X) \mid P(x)\},$$

désigne l'ensemble des séquences x sur X qui vérifient la condition $P(x)$. C'est un sous-ensemble de $\mathrm{W}(X)$, donc un langage sur X. Par exemple, si ρ est la fonction de renversement des séquences sur X (2.2.3), l'ensemble

$$\{x \in \mathrm{W}(X) \mid x = \rho(x)\}$$

est l'ensemble des séquences sur X qui sont identiques à leur renversée : $x = \rho(x)$. Une telle séquence est appelée un *palindrome*. Dans la langue naturelle, un palindrome est un mot qui peut se lire indifféremment de gauche à droite ou de droite à gauche, comme les mots *non*, *elle*, *radar*.

Langages finis et infinis. Un langage *fini* sur X est un ensemble de séquences sur X qui ne comporte qu'un nombre fini de séquences. C'est le cas du langage $\varnothing$ (zéro séquences), des langages $\{\Lambda\}$ et $\{\langle\!\langle\, a \,\rangle\!\rangle\}$ ($a \in X$) (une seule séquence). Si X est un ensemble fini, le langage $\mathrm{W}_n(X)$ pour un $n \in \mathbb{N}$ (Exemple 5) est un langage fini (3.1.3). Si X n'est pas vide, ce qui est le cas en général lorsqu'on parle d'un « alphabet » X, les langages des exemples 2, 6, 7 sont des langages *infinis* sur X. Ils comprennent chacun une infinité de séquences.

3.1.3. Exercice

(a) Soit p le nombre d'éléments d'un ensemble fini X. Exprimer en fonction de p, pour tout $n \in \mathbb{N}$, le nombre d'éléments de l'ensemble $\mathrm{W}_n(X)$. Démontrer la formule proposée par récurrence sur n.

(b) Exprimer de même, pour tout $n \in \mathbb{N}$, le nombre de palindromes de longueur n sur X, c'est-à-dire le nombre d'éléments de l'ensemble $\{x \in \mathrm{W}_n(X) \mid x = \rho(x)\}$, où ρ est la fonction de renversement des séquences sur X. Ce nombre s'exprime différemment selon que n est pair ou impair. On donnera donc deux formules, valables pour tout $n \in \mathbb{N}$: l'une pour le nombre de palindromes de longueur $2n$ sur X ; l'autre pour le nombre de palindromes de longueur $2n + 1$ sur X.

3.1.4. Produit de deux langages

Si A et B sont deux langages sur X, c'est-à-dire si $A \subset \mathrm{W}(X)$ et $B \subset \mathrm{W}(X)$, on pose

$$(1) \qquad \begin{aligned} AB &= \{xy \mid x \in A \text{ et } y \in B\} \\ &= \{x \operatorname{cat} y \mid x \in A \text{ et } y \in B\}. \end{aligned}$$

Autrement dit, AB est l'ensemble des mots sur X formés par concaténation d'un mot du langage A et d'un mot du langage B. Cet ensemble est un langage sur X, appelé le *produit* des langages A, B (dans cet ordre). Par définition, on a la formule

$$(2) \qquad z \in AB \Leftrightarrow \exists x \exists y (z = xy \text{ et } x \in A \text{ et } y \in B).$$

Cette formule se déduit de (1) d'après la Définition 1 de 1.2.3. La définition du produit AB de deux ensembles de séquences est de la même forme générale que celle de la somme $A + B$ de deux ensembles de nombres donnée comme exemple au §1.2.3 (Exemple 1).

Exemple. Soit $X = \{\mathtt{a}, \mathtt{b}\}$ un alphabet à deux éléments distincts. Le produit des langages finis

$$\begin{aligned} A &= \{\mathtt{ab}, \mathtt{baa}\} = \{\langle\!\langle\, \mathtt{a}, \mathtt{b} \,\rangle\!\rangle, \langle\!\langle\, \mathtt{b}, \mathtt{a}, \mathtt{a} \,\rangle\!\rangle\} \\ B &= \{\mathtt{ab}, \mathtt{b}\} = \{\langle\!\langle\, \mathtt{a}, \mathtt{b} \,\rangle\!\rangle, \langle\!\langle\, \mathtt{b} \,\rangle\!\rangle\} \end{aligned}$$

est le langage $AB = \{\mathtt{abab}, \mathtt{abb}, \mathtt{baaab}, \mathtt{baab}\}$.

Tournures informelles. La formule (2), pour deux langages A, B sur un alphabet X, s'exprime souvent de l'une ou l'autre des manières moins formelles suivantes :

$z \in AB$
$\Leftrightarrow$ z admet une décomposition $z = xy$ avec $x \in A$ et $y \in B$
$\Leftrightarrow$ z peut s'écrire sous la forme $z = xy$ avec $x \in A$ et $y \in B$
$\Leftrightarrow$ z est la concaténation d'une séquence de A et d'une séquence de B
$\Leftrightarrow$ z se décompose en une séquence de A et une séquence de B
$\Leftrightarrow$ $z = xy$ avec $x \in A$ et $y \in B$
etc.

Toutes ces tournures *sous-entendent* les quantificateurs $\exists x$, $\exists y$ de la formule exacte (2). Lorsqu'on doit montrer qu'une séquence z appartient à un langage produit AB, il faut montrer qu'il *existe* une séquence $x \in A$ et une séquence $y \in B$ telles que $z = xy$. Ces séquences ne sont d'ailleurs pas uniques en général (Exercice 3.1.6).

3.1.5. Théorème

Si A, B, C sont des langages sur un ensemble X, on a

(1) $(x \in A \text{ et } y \in B) \Rightarrow xy \in AB$.

(2) $AB \subset C \;\Leftrightarrow\; \big[\forall x \forall y : (x \in A \text{ et } y \in B) \Rightarrow xy \in C\,\big]$.

(3) Pour toute famille d'ensembles $(S_z)_{z \in AB}$ indicés par les éléments de AB :

$$\bigcup_{z \in AB} S_z = \bigcup_{x \in A} \Big(\bigcup_{y \in B} S_{xy} \Big).$$

Démonstration. Les formules (1) et (2) sont un cas particulier des théorèmes 1 et 2 de 1.2.3 (voir les exemples 2 et 3 de 1.2.3, qui illustrent ces mêmes théorèmes). Pour la démonstration de (3), voir [A].

3.1.6. Exercice

Considérons, pour deux langages A, B sur un alphabet X, le produit *cartésien* $A \times B$, à savoir l'ensemble des couples de séquences (x, y) tels que $x \in A$ et $y \in B$. Considérons

d'autre par la fonction F de domaine $A \times B$ qui, à chaque couple de séquences $(x, y) \in A \times B$ fait correspondre la séquence $xy = x \,\mathtt{cat}\, y$. Cette fonction est l'ensemble des couples $(x, y) \mapsto xy$ tels que $(x, y) \in A \times B$:

$$F = \{(x, y) \mapsto xy \mid (x, y) \in A \times B\}.$$

Le produit AB des deux langages considérés n'est autre que la portée (1.3.11) de cette fonction :

$$\begin{aligned} \mathrm{Prt}F &= \{F(x, y) \mid (x, y) \in A \times B\} \\ &= \{xy \mid x \in A \text{ et } y \in B\} = AB \qquad (3.1.4). \end{aligned}$$

Dans l'exemple du §3.1.4, cette fonction F est injective. Autrement dit, il n'y a pas deux couples de mots $(x, y) \in A \times B$, $(x', y') \in A \times B$ distincts tels que $xy = x'y'$. Donner un exemple de deux langages A, B sur un alphabet à deux éléments $X = \{\mathtt{a}, \mathtt{b}\}$ pour lesquels cette fonction n'est pas injective.

3.1.7. Exercice

Soient $X = \{\mathtt{a}, \mathtt{b}\}$ un alphabet avec $\mathtt{a} \neq \mathtt{b}$, et

$$C = \{\mathtt{abbba}, \mathtt{baa}, \mathtt{aba}, \mathtt{babba}\}.$$

Donner deux couples distincts (A, B) et (A', B') de langages sur X tels que $C = AB$ et $C = A'B'$ et la séquence Λ n'appartient à aucun des langages A, B, A', B'.

3.1.8. Exercice

Pour quels langages A, pris dans les exemples du §3.1.2, a-t-on $AA = A$?

3.1.9. Exercice

Démontrer en utilisant la formule 2.1.9(4) que, pour deux langages A, B quelconques sur X, on a

$$\Lambda \in AB \Leftrightarrow (\Lambda \in A \text{ et } \Lambda \in B).$$

3.1.10. Exercice

Soient A et B des langages sur X, x et y des séquences sur X. Montrer que

$$xy \in AB \Leftrightarrow \left(\begin{array}{l} \exists u \in \mathrm{W}(X) : \exists a \in A : \ x = au \text{ et } uy \in B \\ \text{ ou} \\ \exists u \in \mathrm{W}(X) : \exists b \in B : \ xu \in A \text{ et } y = ub \end{array} \right).$$

3.1.11. Théorèmes

Soient A, B, C des langages sur un ensemble X et $(D_i)_{i \in I}$ une famille de langages sur X. On a :

(1) $(AB)C = A(BC)$ — Associativité du produit de langages.

(2) $A\{\Lambda\} = A$ et $\{\Lambda\}A = A$ — Langage neutre $\{\Lambda\}$.

(3) $A\Big(\bigcup_{i \in I} D_i\Big) = \bigcup_{i \in I}(AD_i)$ — Distributivité du produit de langages par rapport à la réunion.

$\Big(\bigcup_{i \in I} D_i\Big)A = \bigcup_{i \in I}(D_iA)$

Cas particulier : $A(D_1 \cup D_2) = AD_1 \cup AD_2$

$(D_1 \cup D_2)A = D_1A \cup D_2A.$

Démonstration de ces formules : voir [A].

3.2 Algèbres de Kleene

3.2.1. Exemple. L'algèbre de Kleene des langages sur un ensemble X

Nous désignerons désormais par $\mathcal{L}(X)$ l'ensemble des langages sur un ensemble X. Un langage sur X étant par définition (3.1.2) une partie quelconque de l'ensemble $\mathrm{W}(X)$, l'ensemble $\mathcal{L}(X)$ n'est autre que l'ensemble des parties de $\mathrm{W}(X)$:

$$\mathcal{L}(X) = \mathcal{P}(\mathrm{W}(X)). \tag{1}$$

En vertu des formules 3.1.11 (1) et 3.1.11 (2), le triplet

$$\big(\mathcal{L}(X),\ \text{opération produit de langages},\ \{\Lambda\}\big)$$

est un monoïde (1.6.2), le *monoïde des langages sur* X. Le langage à un seul élément $\{\Lambda\}$ est l'élément neutre de ce monoïde. Il est appelé le *langage neutre*. Il faut se garder évidemment de confondre ce langage avec le langage $\varnothing$. Le langage $\{\Lambda\}$ est souvent noté abusivement Λ. Par exemple, au lieu de 3.1.11 (2), on rencontre souvent la notation abusive $A\Lambda = A$ et $\Lambda A = A$.

En vertu des formules de distributivité 3.1.11 (3), l'ensemble $\mathcal{L}(X)$ des langages sur un alphabet X, muni de l'opération produit de langages et du langage neutre $\{\Lambda\}$, est non seulement un monoïde, mais encore ce que nous appelons une *algèbre de Kleene*. Nous donnerons plus loin la définition générale de ce concept. Comme introduction, nous discutons ici le cas particulier de l'algèbre de Kleene des langages sur un ensemble X.

Pour alléger la notation, posons

$$\mathcal{K} = \mathcal{L}(X),$$

X étant un ensemble quelconque. Les éléments de l'ensemble $\mathcal{K}$ ainsi défini sont les langages sur X. Les éléments de $\mathcal{K}$ sont donc des ensembles, puisqu'un langage est un ensemble de séquences. On peut donc appliquer aux éléments de $\mathcal{K}$ l'opération ensembliste de réunion. Une famille $(A_i)_{i\in I}$ d'éléments de $\mathcal{K}$ est une famille de langages sur X et sa réunion

$$\bigcup_{i\in I} A_i$$

est encore un langage sur X, donc un élément de $\mathcal{K}$. Ceci est l'une des propriétés caractéristiques de ce que nous appelons une algèbre de Kleene. Il s'agit d'un monoïde $\mathcal{K}$ dont les éléments sont des ensembles et qui satisfait la condition suivante : pour toute famille $(A_i)_{i\in I}$ d'ensembles $A_i \in \mathcal{K}$, l'ensemble $B = \bigcup_{i\in I} A_i$ appartient à $\mathcal{K}$. En outre, l'opération du monoïde (dans notre cas le produit de langages sur X) est *distributive par rapport à la réunion*, c'est-à-dire vérifie les formules 3.1.11 (3). L'élément neutre d'un tel monoïde sera noté $I_{\mathcal{K}}$. Dans le cas de l'algèbre de Kleene des langages sur un ensemble X, $\mathcal{K} = \mathcal{L}(X)$, l'élément neutre $I_{\mathcal{K}}$ est le langage neutre $\{\Lambda\}$.

3.2.2. Algèbres de Kleene. Définition

Une *algèbre de Kleene* est un monoïde $(\mathcal{K}, \varphi, I_{\mathcal{K}})$ dans lequel $\mathcal{K}$ est un ensemble d'ensembles, et qui satisfait aux conditions supplémentaires suivantes :

(1) Pour toute famille $(A_i)_{i\in I}$ d'ensembles $A_i \in \mathcal{K}$, on a $\bigcup_{i\in I} A_i \in \mathcal{K}$.

(2) L'opération φ est distributive par rapport à la réunion, en ce sens que, pour toute famille $(A_i)_{i\in I}$ d'ensembles $A_i \in \mathcal{K}$ et pour tout ensemble $B \in \mathcal{K}$, on a, en notant φ

multiplicativement :

$$\Big(\bigcup_{i\in I} A_i\Big)B = \bigcup_{i\in I}(A_i B); \qquad B\Big(\bigcup_{i\in I} A_i\Big) = \bigcup_{i\in I}(B A_i).$$

Comme cas particulier ($I = \{1, 2\}$), on a, quels que soient A_1, A_2, $B \in \mathcal{K}$:

$$(A_1 \cup A_2)B = A_1 B \cup A_2 B \text{ et } B(A_1 \cup A_2) = B A_1 \cup B A_2.$$

Exemples. Nous aurons affaire dans ce cours à deux types d'exemples d'algèbres de Kleene : celle des langages sur un ensemble X, considérée dans le présent chapitre, et celle des relations dans un ensemble E, étudiée au chapitre 5.

Dans la présente section, nous allons établir un certain nombre de théorèmes, constituant un répertoire de « formules générales » valables dans une algèbre de Kleene quelconque, et que nous pourrons donc appliquer aussi bien aux langages sur un ensemble X qu'aux relations dans un ensemble E.

En lisant cette section, pour « fixer les idées », le lecteur peut penser systématiquement à l'algèbre de Kleene des langages sur X, c'est-à-dire interpréter : les variables A, B, C, ... des formules comme désignant des langages sur X, les produits comme étant des produits de langages, le symbole $l_{\mathcal{K}}$ comme le langage neutre $\{\Lambda\}$. L'essentiel est que les formules présentées découlent uniquement de l'associativité du produit, de la neutralité de $l_{\mathcal{K}}$ pour le produit et de la distributivité du produit par rapport à la réunion, qui sont les propriétés caractéristiques d'une algèbre de Kleene.

Par exemple, on établit au paragraphe 3.2.3 la formule $\varnothing B = \varnothing$. Interprétée dans l'algèbre des langages sur X, elle signifie : le produit (au sens de 3.1.4) du langage $\varnothing$ et d'un langage B quelconque sur X est le langage $\varnothing$. Cela se démontre aisément au moyen de la définition du produit (3.1.4). En effet, raisonnons par réduction à l'absurde. Supposons que $\varnothing B \neq \varnothing$, c'est-à-dire qu'il existe une séquence $z \in \varnothing B$. Par définition du produit de langages, une telle séquence admet une décomposition $z = xy$ avec $x \in \varnothing$ et $y \in B$. Cela est impossible car il n'existe pas de $x \in \varnothing$. Démontrée de cette manière, la formule $\varnothing B = \varnothing$ n'est valable que pour l'algèbre des langages sur un alphabet X. Pour qu'elle soit valable dans une algèbre de Kleene quelconque, elle sera démontrée sans aucune interprétation du produit d'ensembles considéré, mais en utilisant uniquement sa propriété de distributivité par rapport à la réunion.

3.2.3. Théorèmes

Soient $\mathcal{K}$ une algèbre de Kleene et A, B, C, A', B' des éléments de $\mathcal{K}$. On a :

(1) $(AB)C = A(BC)$.

(2) $l_{\mathcal{K}} A = A l_{\mathcal{K}} = A$.

(3) $\varnothing \in \mathcal{K}$.

(4) $\varnothing B = \varnothing$ et $B\varnothing = \varnothing$.

(5) $A \subset A' \Rightarrow AB \subset A'B$.
$B \subset B' \Rightarrow AB \subset AB'$.
$(A \subset A' \text{ et } B \subset B') \Rightarrow AB \subset A'B'$.

(6) $l_{\mathcal{K}} \subset B \Rightarrow (A \subset AB \text{ et } A \subset BA)$.

Les deux premières de ces formules ne font que rappeler le fait qu'une algèbre de Kleene est par définition (3.2.2) un monoïde (multiplicatif) et que $l_{\mathcal{K}}$ en est l'élément neutre. La démonstration des autres formules est donnée dans [A].

3.2.4. Puissances d'un ensemble A dans une algèbre de Kleene

Soit $\mathcal{K}$ une algèbre de Kleene. On définit les puissances A^n ($n \in \mathbb{N}$) d'un ensemble $A \in \mathcal{K}$ en posant

(1) $$A^0 = I_{\mathcal{K}}.$$

(2) $$\forall n \in \mathbb{N} : \quad A^{n+1} = A(A^n).$$

Cela revient à dire que A^n est le n-ième itéré de A (1.6.5) en tant qu'élément du monoïde multiplicatif $(\mathcal{K}, \cdot, I_{\mathcal{K}})$. A^n est la notation multiplicative pour $\mathrm{it}_A(n)$ (1.6.5). On a $A^1 = A$, $A^2 = AA$, etc. Les formules de 1.6.6 et de 1.6.7, en notation multiplicative, s'appliquent à ce cas particulier. Ce sont notamment les formules d'exponentiation

$$A^{m+n} = A^m A^n.$$
$$A^{mn} = (A^m)^n. \qquad 1.6.7(4)$$

3.2.5. L'opération de fermeture dans une algèbre de Kleene

Soit $\mathcal{K}$ une algèbre de Kleene. Pour tout ensemble $A \in \mathcal{K}$, on pose :

(1) $$A^* = \bigcup_{n \in \mathbb{N}} A^n.$$

En vertu de la propriété 3.2.2(1) d'une algèbre de Kleene $\mathcal{K}$, si $A \in \mathcal{K}$ alors $A^* \in \mathcal{K}$, puisque $A^n \in \mathcal{K}$ pour tout $n \in \mathbb{N}$. L'ensemble A^* (lire : A étoile) est appelé la *fermeture de Kleene* de A ou aussi *l'étoile* de A. Par définition de la réunion d'une famille d'ensembles (1.3.19), on a

(2) $$z \in A^* \Leftrightarrow \exists n \in \mathbb{N} : z \in A^n.$$

La formule suivante est une conséquence immédiate de la définition de A^* et des propriétés générales de la réunion et de l'inclusion : la réunion d'une famille d'ensembles $(A_i)_{i \in I}$ est contenue dans un ensemble B si et seulement si chacun des ensembles A_i est contenu dans B. Donc, si $A, B \in \mathcal{K}$, on a

(3) $$A^* \subset B \;\Leftrightarrow\; \forall n \in \mathbb{N} : A^n \subset B.$$

D'après cette formule, pour démontrer une inclusion $A^* \subset B$, pour un ensemble $A \in \mathcal{K}$, il suffit de démontrer $\forall n \in \mathbb{N} : A^n \subset B$, ce qui peut se faire par récurrence sur n.

3.2.6. Les quatre propriétés caractéristiques de l'opération *

Soient $\mathcal{K}$ une algèbre de Kleene et $A \in \mathcal{K}$. L'ensemble A^* est défini (3.2.5) comme la réunion des ensembles A^n ($n \in \mathbb{N}$). Nous allons énoncer dans ce paragraphe quatre propriétés de cet ensemble qui le *caractérisent complètement*, en ce sens que l'égalité 3.2.5(1), par laquelle nous avons défini A^*, est *équivalente* à la conjonction de ces quatre propriétés. Celles-ci s'expriment par les formules suivantes, démontrées dans [A] :

(1) $I_{\mathcal{K}} \subset A^*$.

(2) $A \subset A^*$.

(3) $A^*A^* \subset A^*$.

(4) Pour tout ensemble $M \in \mathcal{K}$, on a

$$(I_{\mathcal{K}} \subset M \text{ et } A \subset M \text{ et } MM \subset M) \Rightarrow A^* \subset M.$$

L'utilisation principale de la formule (4) est la suivante : pour démontrer une inclusion de la forme $A^* \subset M$, il suffit de démontrer les trois inclusions $I_{\mathcal{K}} \subset M$, $A \subset M$, $MM \subset M$. Cette méthode est appliquée plusieurs fois dans la démonstration des formules du §3.2.7.

3.2.7. Théorèmes

Soit $\mathcal{K}$ une algèbre de Kleene. Quels que soient $A, B, C \in \mathcal{K}$, on a

(1) $\varnothing^* = 1_{\mathcal{K}}$.

(2) $A^*A^* = A^*$.

(3) $(A^*)^* = A^*$.

(4) $A \subset B \Rightarrow A^* \subset B^*$.

(5) $A \subset B^* \Rightarrow A^* \subset B^*$.

(6) $A = A^* \Leftrightarrow (1_{\mathcal{K}} \subset A \text{ et } AA \subset A)$.

(7) $(A \subset C^* \text{ et } B \subset C^*) \Rightarrow AB \subset C^*$.

(8) $A \subset C^* \Rightarrow (AC^* \subset C^* \text{ et } C^*A \subset C^*)$.

(9) $AA^* \subset A^* \text{ et } A^*A \subset A^*$.

(10) $A \subset AB^* \text{ et } A \subset B^*A$.

(11) $A \subset C \Rightarrow (A^*C^* = C^* \text{ et } C^*A^* = C^*)$.

(12) $A^* = 1_{\mathcal{K}} \cup AA^*$.
$A^* = 1_{\mathcal{K}} \cup A^*A$.

(13) $B \subset A^* \Rightarrow A^* = (A \cup B)^*$.

(14) $(A \cup B)^* = (A^*B^*)^*$.

Les démonstrations de ces formules sont données dans [A].

3.2.8. Définition

Si $\mathcal{K}$ est une algèbre de Kleene, on pose, pour tout ensemble $A \in \mathcal{K}$:

(1) $$A^+ = \bigcup_{n \in \mathbb{N}} A^{n+1}.$$

Une définition équivalente est

$$A^+ = \bigcup_{n \in \mathbb{N}-\{0\}} A^n.$$

En vertu de la distributivité du produit par rapport à la réunion (3.2.2(2)), on tire de (1):

$$A^+ = A\Big(\bigcup_{n \in \mathbb{N}} A^n\Big) = \Big(\bigcup_{n \in \mathbb{N}} A^n\Big)A.$$

Donc, d'après 3.2.5(1) et 3.2.7(12), on a, pour tout ensemble $A \in \mathcal{K}$:

(2) $A^+ = AA^* = A^*A$.

(3) $A^* = 1_{\mathcal{K}} \cup A^+$.

3.3 L'algèbre de Kleene des langages sur X

3.3.1. Puissances d'un langage

Dans ce paragraphe et dans les suivants, nous nous plaçons dans le cas particulier de l'algèbre de Kleene des langages sur un alphabet X, $\mathcal{K} = \mathcal{L}(X)$. Il s'agit de décrire les

puissances A^n d'un langage A ainsi que le langage A^*, c'est-à-dire de caractériser les séquences qui appartiennent à ces langages.

Les puissances A^n d'un langage A, dans l'algèbre de Kleene $\mathcal{K} = \mathcal{L}(X)$, sont définies par les formules 3.2.4(1) et 3.2.4(2). La première s'écrit, dans ce cas particulier:

$$A^0 = \{\Lambda\}. \tag{1}$$

Nous voulons discuter la signification de A^n pour n quelconque. Nous rappelons que le produit AB de deux langages est l'ensemble des séquences de la forme xy (x cat y) avec $x \in A$ et $y \in B$. Une séquence z appartient donc au langage $A^2 = AA$ si et seulement si elle admet une décomposition $z = x_1$ cat x_2 avec $x_1 \in A$ et $x_2 \in A$. Elle appartient au langage $A^3 = A(A^2)$ si et seulement si elle admet une décomposition $z = x_1$ cat y avec $x_1 \in A$ et $y \in A^2$, donc une décomposition de la forme $z = x_1$ cat x_2 cat x_3 où chacune des séquences x_1, x_2, x_3 appartient à A. De façon générale, il est clair (intuitivement) qu'une séquence z appartient au langage A^n ($n > 0$) si et seulement si elle admet une décomposition en n séquences

$$z = x_1 \text{ cat } x_2 \text{ cat } \cdots \text{ cat } x_n$$

avec $x_i \in A$ pour $i = 1, \ldots, n$.

En d'autres termes, une séquence z appartient au langage A^n ($n > 0$) si et seulement s'il existe une *séquence de séquences* $x = \langle\!\langle\, x_1, \ldots, x_n \,\rangle\!\rangle = \langle\!\langle\, x[1], \ldots, x[n] \,\rangle\!\rangle$, appartenant à l'ensemble $\mathrm{W}_n(\mathrm{W}(X))$ (2.3.5), telle que l'on ait $x[i] \in A$ pour tout $i \in [1 \,..\, n]$ et

$$z = \mathop{\mathsf{CAT}}_{i=1}^{n} x[i] = \mathit{eval}_{\mathrm{W}(X)}(x).$$

En vertu de (1) et de 2.3.5(2), cet énoncé est aussi vrai pour $n = 0$.

Formellement, à partir de la définition des puissances de A (3.2.4), de celle du produit de langages (3.1.4), et de 2.3.5(2), on démontre par récurrence sur n (voir [A]) la formule suivante, pour un langage quelconque A sur X:

(2) Pour tout $n \in \mathbb{N}$:

$$z \in A^n \;\Leftrightarrow\; \exists x \in \mathrm{W}_n(\mathrm{W}(X)) : \begin{pmatrix} \forall i \in [1 \,..\, n] : x[i] \in A \\ \text{et} \quad z = \mathit{eval}_{\mathrm{W}(X)}(x) \end{pmatrix}.$$

3.3.2. Fermeture de Kleene d'un langage

Soit A un langage sur X. D'après 3.2.5(2), une séquence z appartient au langage A^* si et seulement s'il existe un $n \in \mathbb{N}$ tel que $z \in A^n$, autrement dit si et seulement si z appartient à une puissance de A. D'après ce que nous venons de voir, cela signifie que z se décompose en un certain nombre (quelconque) de séquences appartenant à A. De façon plus précise, $z \in A^*$ équivaut à: z est la concaténation d'une séquence de séquences $x = \langle\!\langle\, x[1], \ldots, x[\lg(x)] \,\rangle\!\rangle$ telle que $x[i] \in A$ pour tout $i \in [1 \,..\, \lg(x)]$. Autrement dit, pour un langage quelconque A sur X, on a:

$$z \in A^* \;\Leftrightarrow\; \exists x \in \mathrm{W}(\mathrm{W}(X)) : \begin{pmatrix} z = \mathop{\mathsf{CAT}}_{i=1}^{\lg(x)} x[i] \quad \text{et} \\ \forall i \in [1 \,..\, \lg(x)] : x[i] \in A. \end{pmatrix} \tag{1}$$

3.3.3. Formules sur la séquence vide

Si $\mathcal{K}$ est l'algèbre de Kleene des langages sur un alphabet X, l'élément neutre $l_{\mathcal{K}}$ de $\mathcal{K}$ est le langage neutre $\{\Lambda\}$. On peut donc récrire dans ce cas toutes les formules des

paragraphes 3.2.3, 3.2.4, 3.2.6, 3.2.7 dans lesquelles intervient $l_{\mathcal{K}}$ en tenant compte de l'égalité $l_{\mathcal{K}} = \{\Lambda\}$. On obtient ainsi la liste de formules ci-dessous, où A, B, M représentent des langages quelconques sur X. Chaque formule est munie de la référence à la formule générale correspondante.

(1) $\{\Lambda\}A = A\{\Lambda\} = A$. 3.2.3(2)

(2) $\Lambda \in B \Rightarrow (A \subset AB$ et $B \subset BA)$. 3.2.3(6)

(3) $A^0 = \{\Lambda\}$. 3.2.4(1)

(4) $\Lambda \in A^*$. 3.2.6(1)

(5) $(\Lambda \in M$ et $A \subset M$ et $MM \subset M) \Rightarrow A^* \subset M$. 3.2.6(4)

(6) $\varnothing^* = \{\Lambda\}$. 3.2.7(1)

(7) $A = A^* \Leftrightarrow (\Lambda \in A$ et $AA \subset A)$. 3.2.7(6)

(8) $A^* = \{\Lambda\} \cup AA^* = \{\Lambda\} \cup A^*A$. 3.2.7(12)

(9) $A^* = \{\Lambda\} \cup A^+$. 3.2.8(3)

3.3.4. Exemples de vérification de formules

En principe, il est totalement superflu de redémontrer, pour l'algèbre de Kleene des langages sur X, les formules déjà démontrées pour une algèbre de Kleene quelconque. Néanmoins, pour se familiariser avec le produit, les puissances et la fermeture de Kleene de langages, la redémonstration de quelques formules en utilisant la définition du produit (3.1.4) et la caractérisation des puissances et de la fermeture de Kleene d'un langage (3.3.1, 3.3.2) est un exercice très utile.

Comme exemple, nous redémontrons ainsi dans ce paragraphe deux formules. Nous le faisons de manière très informelle. Une version plus formelle est donnée dans [A].

Nous vérifions d'abord la formule

$$A^* = \{\Lambda\} \cup AA^*$$

(3.3.3(8)) pour un langage A quelconque sur X. Nous montrons que $z \in A^*$ implique $z \in \{\Lambda\} \cup AA^*$ et réciproquement. Une séquence $z \in A^*$ admet une décomposition $z = x_1 \,\mathsf{cat}\, \cdots \,\mathsf{cat}\, x_n$, en n séquences appartenant chacune au langage A (avec $n \in \mathbb{N}$ quelconque). Si $n = 0$, cela signifie que $z = \Lambda$, donc que $z \in \{\Lambda\}$. Si $n \neq 0$, alors z est la concaténation de la séquence $x_1 \in A$ avec la séquence $x_2 \,\mathsf{cat}\, \cdots \,\mathsf{cat}\, x_n$, qui appartient à A^*, donc z appartient au produit de langages AA^*. On voit ainsi que $A^* \subset \{\Lambda\} \cup AA^*$. Inversement, une séquence z qui appartient au langage $\{\Lambda\} \cup AA^*$ est soit la séquence Λ, qui appartient au langage A^* (3.3.3(4)), soit la concaténation d'une séquence de A avec une séquence de A^*, et dans ce cas se décompose en un certain nombre de séquences de A, donc appartient au langage A^*.

Comme deuxième exemple, nous considérons la formule 3.2.7(14),

$$(A \cup B)^* = (A^*B^*)^*,$$

en supposant que A et B sont des langages sur X. Nous montrons que $z \in (A \cup B)^* \Rightarrow z \in (A^*B^*)^*$ et réciproquement. Supposons premièrement que $z \in (A \cup B)^*$. Un telle séquence admet une décomposition $z = x_1 \,\mathsf{cat}\, \cdots \,\mathsf{cat}\, x_n$ en n séquences x_i appartenant chacune au langage $(A \cup B)$, donc appartenant chacune au langage A ou au langage B. Pour montrer que $z \in (A^*B^*)^*$, il suffit de montrer que chacune des séquences x_i de cette décomposition de z appartient au langage A^*B^*. On raisonne par disjonction des cas $x_i \in A$, $x_i \in B$. Dans le premier cas, comme $A \subset A^*$ (3.2.6(2)), on a $x_i \in A^*B^*$ car x_i admet la décomposition

$x_i = x_i\Lambda$ avec $x_i \in A^*$ et $\Lambda \in B^*$ (3.3.3(4)). Dans le deuxième cas, comme $B \subset B^*$, on a $x_i \in A^*B^*$ car x_i admet la décomposition $x_i = \Lambda x_i$ avec $\Lambda \in A^*$ et $x_i \in B^*$.

Inversement, si $z \in (A^*B^*)^*$, z admet une décomposition $z = y_1 \,\mathsf{cat}\, \cdots \,\mathsf{cat}\, y_m$ en m séquences ($m \geq 0$) appartenant chacune au langage A^*B^*. Chacune des séquences y_i se décompose à son tour en un certain nombre de séquences appartenant à A suivies d'un certain nombre de séquences appartenant à B. On voit donc que z admet finalement une décomposition en des séquences qui appartiennent toutes à A ou à B, donc à $A \cup B$. Par conséquent, $z \in (A \cup B)^*$.

3.3.5. Exercice

Démontrer la formule suivante dans une algèbre de Kleene $\mathcal{K}$ quelconque et pour des ensembles A, $B \in \mathcal{K}$ quelconques. En donner aussi une démonstration informelle (cf. 3.3.4) dans le cas particulier où $\mathcal{K}$ est l'algèbre de Kleene des langages sur un alphabet X.

$$(A \cup B)^* = A^* \cup A^*B(A \cup B)^*.$$

3.3.6. Exercice

Montrer que l'opération $*$ dans une algèbre de Kleene $\mathcal{K}$ est entièrement déterminée par les quatre formules 3.2.6(1) à 3.2.6(4). Pour cela, montrer que si une autre opération $A^\otimes$ dans $\mathcal{K}$ vérifie quatre formules semblables, à savoir, pour tout $A \in \mathcal{K}$:

(1′) $I_\mathcal{K} \subset A^\otimes$;
(2′) $A \subset A^\otimes$;
(3′) $A^\otimes A^\otimes \subset A^\otimes$;
(4′) $\forall M \in \mathcal{K}: \quad (I_\mathcal{K} \subset M \text{ et } A \subset M \text{ et } MM \subset M) \Rightarrow A^\otimes \subset M$;

alors on a $A^\otimes \subset A^*$ et $A^* \subset A^\otimes$ (quel que soit $A \in \mathcal{K}$), de sorte que les deux opérations coïncident.

3.3.7. Exercice

La formule suivante est valide dans l'algèbre de Kleene $\mathcal{K} = \mathcal{L}(X)$ des langages sur un alphabet X, mais n'est pas valide dans toute algèbre de Kleene:

$$I_\mathcal{K} \subset AB \Leftrightarrow (I_\mathcal{K} \subset A \text{ et } I_\mathcal{K} \subset B).$$

Pour l'algèbre $\mathcal{K} = \mathcal{L}(X)$, elle découle immédiatement de 3.1.9 et du fait que $I_\mathcal{K} = \{\Lambda\}$.

En se servant de cette formule particulière et des formules générales de 3.2.8, démontrer les propositions suivantes pour un langage A quelconque sur X:

(1) $\Lambda \in A \Leftrightarrow \Lambda \in A^+$.
(2) $\Lambda \in A \Leftrightarrow A^+ = A^*$.

3.4 Langages réguliers

3.4.1. Constructions génératrices régulières

Dans toute cette section, nous supposons que $\mathcal{K}$ est une algèbre de Kleene et que $\mathfrak{S}$ ([6]) est une partie de $\mathcal{K}$ ($\mathfrak{S} \subset \mathcal{K}$). Pour fixer les idées, le lecteur peut penser à $\mathcal{K}$ comme étant l'algèbre de Kleene $\mathcal{L}(X)$ des langages sur un ensemble X et à $\mathfrak{S}$ comme étant un ensemble

[6] $\mathfrak{S}$ est la lettre S de l'écriture « gothique » ou « Fraktur ».

de langages particuliers $A, B, C, \ldots$ sur X, d'où $\mathfrak{S} \subset \mathcal{K}$ puisque $\mathcal{L}(X)$ est l'ensemble de tous les langages sur X. Mais la définition qui suit est indépendante de cette interprétation.

Définition. On appelle *construction génératrice régulière de base* $\mathfrak{S}$ *dans* $\mathcal{K}$, ou simplement *construction régulière de base* $\mathfrak{S}$, toute séquence

$$R = \langle\!\langle\, R_1, R_2, \ldots, R_n \,\rangle\!\rangle$$

d'ensembles $R_i \in \mathcal{K}$ telle que, pour chaque $k \in [1 \,..\, n]$, l'une des conditions (a) à (e) suivantes soit vérifiée:

(a) $R_k \in \mathfrak{S}$.

(b) $R_k = \varnothing$.

(c) $R_k = R_i \cup R_j$ pour deux ensembles R_i, R_j tels que $i < k$ et $j < k$.
En d'autres termes, il existe deux entiers $i, j \in [1 \,..\, k-1]$ tels que $R_k = R_i \cup R_j$.

(d) $R_k = R_i R_j$ pour deux ensembles R_i, R_j tels que $i < k$ et $j < k$.
En d'autres termes, il existe deux entiers $i, j \in [1 \,..\, k-1]$ tels que $R_k = R_i R_j$.

(e) $R_k = R_i^*$ pour un ensemble R_i tel que $i < k$.
En d'autres termes, il existe un entier $i \in [1 \,..\, k-1]$ tel que $R_k = R_i^*$.

3.4.2. Exemple

Supposons que A, B, C soient des ensembles appartenant à $\mathfrak{S}$. La séquence $R = \langle\!\langle\, R_1, \ldots, R_9 \,\rangle\!\rangle$ définie par les égalités suivantes est une construction génératrice régulière de base $\mathfrak{S}$:

$$\begin{array}{ll}
R_1 = A & \in \mathfrak{S} \\
R_2 = B & \in \mathfrak{S} \\
R_3 = A \cup B & = R_1 \cup R_2 \\
R_4 = (A \cup B)^* & = R_3^* \\
R_5 = C & \in \mathfrak{S} \\
R_6 = (A \cup B)^* C & = R_4 R_5 \\
R_7 = \varnothing & = \varnothing \\
R_8 = \varnothing^* & = R_7^* \\
R_9 = (A \cup B)^* C \cup \varnothing^* & = R_6 \cup R_8.
\end{array}$$

Lorsqu'on écrit une construction génératrice régulière $\langle\!\langle\, R_1, \ldots, R_n \,\rangle\!\rangle$ de base $\mathfrak{S}$ en notant chacun des ensembles R_i de la séquence sur une ligne, comme dans l'exemple ci-dessus, on voit que la définition d'une telle construction revient à dire qu'à chaque ligne on peut écrire: un ensemble appartenant à $\mathfrak{S}$, ou l'ensemble vide, ou la réunion de deux ensembles pris dans des lignes précédentes (éventuellement deux fois la même ligne), ou le produit de deux ensembles pris dans des lignes précédentes (éventuellement deux fois la même ligne), ou la fermeture de Kleene d'un ensemble d'une ligne précédente.

3.4.3. Théorèmes

Soient $\mathcal{K}$ une algèbre de Kleene et $\mathfrak{S} \subset \mathcal{K}$.

(1) La séquence vide $\langle\!\langle\, \rangle\!\rangle = \Lambda$ est une construction génératrice régulière de base $\mathfrak{S}$.

(2) Si $\langle\!\langle\, R_1, \ldots, R_n \,\rangle\!\rangle$ est une construction génératrice régulière de base $\mathfrak{S}$ de longueur $n \neq 0$, alors $R_1 = \varnothing$ ou $R_1 \in \mathfrak{S}$.

(3) Si $\langle\!\langle\, R_1, \ldots, R_n \,\rangle\!\rangle$ est une construction génératrice régulière de base $\mathfrak{S}$, chacune des séquences partielles $\langle\!\langle\, R_1, \ldots, R_m \,\rangle\!\rangle$ ($m \in [1 \,..\, n]$) est elle-même une construction génératrice régulière de base $\mathfrak{S}$.

(4) Si $R = \langle\!\langle\, R_1, \ldots, R_m \,\rangle\!\rangle$ et $R' = \langle\!\langle\, R'_1, \ldots, R'_n \,\rangle\!\rangle$ sont deux constructions génératrices régulières de base $\mathfrak{S}$, la concaténation de ces deux séquences

$$R'' = R \;\mathsf{cat}\; R' = \langle\!\langle\, R_1, \ldots, R_m, R'_1, \ldots, R'_n \,\rangle\!\rangle$$

est aussi une construction génératrice régulière de base $\mathfrak{S}$.

Démonstration. Ces propriétés découlent immédiatement de la définition d'une construction génératrice régulière de base $\mathfrak{S}$ (3.4.1).

(1) Désignons par $P(k)$ la proposition qui est la disjonction (OU) des propositions (a), (b), (c), (d), (e) de 3.4.1. Une construction génératrice régulière de base $\mathfrak{S}$ est par définition une séquence R qui vérifie la condition $\forall k \in [1 \,..\, n] : P(k)$, où n est la longueur de R. Cette condition est trivialement satisfaite si $n = 0$.

(2) Le premier ensemble (R_1) d'une construction génératrice régulière R de base $\mathfrak{S}$ ne peut vérifier que la condition (a) ou la condition (b) de 3.4.1.

(3) Si $\langle\!\langle\, R_1, \ldots, R_n \,\rangle\!\rangle$ est une construction génératrice régulière de base $\mathfrak{S}$, alors, pour chaque $k \in [1 \,..\, n]$, l'une des conditions (a) – (e) de 3.4.1 est vérifiée. Cela est vrai en particulier pour chaque $k \in [1 \,..\, m]$, si $m \leq n$, donc la séquence partielle $\langle\!\langle\, R_1, \ldots, R_m \,\rangle\!\rangle$ est aussi une construction génératrice de base $\mathfrak{S}$.

(4) Si R'' est la concaténation de deux constructions génératrices régulières R, R' de base $\mathfrak{S}$, de longueurs respectives m et n, il est clair que pour chaque $k \in [1 \,..\, m + n]$, l'ensemble $R''_k = R''[k]$ est soit un ensemble appartenant à $\mathfrak{S}$, soit le l'ensemble $\emptyset$, soit la réunion ou le produit de deux ensembles qui figurent avant lui dans la séquence R'', soit la fermeture de Kleene d'un ensemble qui figure avant lui dans la séquence R''. Donc la séquence R'' est une construction génératrice de base $\mathfrak{S}$.

3.4.4. Ensembles régulièrement engendrés par $\mathfrak{S}$ dans $\mathcal{K}$

Définition. Soient $\mathcal{K}$ une algèbre de Kleene et $\mathfrak{S} \subset \mathcal{K}$. On dit qu'un ensemble $A \in \mathcal{K}$ est *régulièrement engendré* par $\mathfrak{S}$ s'il existe une construction génératrice régulière $\langle\!\langle\, R_1, \ldots, R_n \,\rangle\!\rangle$ de base $\mathfrak{S}$ dans $\mathcal{K}$ telle que $A = R_n$.

De manière moins formelle, on peut dire qu'un ensemble $A \in \mathcal{K}$ est régulièrement engendré par $\mathfrak{S}$ s'il peut être construit à partir d'ensembles appartenant à $\mathfrak{S} \cup \{\emptyset\}$ en effectuant un nombre fini d'opérations de réunion (de deux ensembles) ou de produit ou de fermeture de Kleene.

Remarque. En vertu de 3.4.3(3), si $\langle\!\langle\, R_1, \ldots, R_n \,\rangle\!\rangle$ est une construction génératrice régulière de base $\mathfrak{S}$, *chacun des ensembles* R_m ($m = 1, \ldots, n$) de la séquence, et pas seulement le dernier d'entre eux, est régulièrement engendré par $\mathfrak{S}$.

Exemple. Si A, B, C sont trois ensembles appartenant à $\mathfrak{S}$, alors chacun des ensembles $R_1, \ldots, R_9$ de l'exemple 3.4.2 est régulièrement engendré par $\mathfrak{S}$.

Notation. Nous désignons par $\mathsf{Reg}(\mathfrak{S})$ l'ensemble des ensembles $A \in \mathcal{K}$ qui sont régulièrement engendrés par une partie $\mathfrak{S}$ de $\mathcal{K}$. Par définition, on a donc

$$\begin{aligned} A \in \mathsf{Reg}(\mathfrak{S}) \;&\Leftrightarrow\; A \in \mathcal{K} \text{ et } A \text{ est régulièrement engendré par } \mathfrak{S} \\ &\Leftrightarrow\; \text{il existe une construction génératrice régulière} \\ &\qquad \langle\!\langle\, R_1, \ldots, R_n \,\rangle\!\rangle \text{ de base } \mathfrak{S} \text{ dans } \mathcal{K} \text{ telle que } A = R_n. \end{aligned}$$

3.4.5. Théorèmes

Soient $\mathcal{K}$ une algèbre de Kleene et $\mathfrak{S} \subset \mathcal{K}$. On a :

(1) $\varnothing \in \mathsf{Reg}(\mathfrak{S})$.

(2) $I_{\mathcal{K}} \in \mathsf{Reg}(\mathfrak{S})$.

(3) $A \in \mathfrak{S} \Rightarrow A \in \mathsf{Reg}(\mathfrak{S})$.

(4) $(A \in \mathsf{Reg}(\mathfrak{S})$ et $B \in \mathsf{Reg}(\mathfrak{S})) \Rightarrow (A \cup B) \in \mathsf{Reg}(\mathfrak{S})$.

(5) $(A \in \mathsf{Reg}(\mathfrak{S})$ et $B \in \mathsf{Reg}(\mathfrak{S})) \Rightarrow AB \in \mathsf{Reg}(\mathfrak{S})$.

(6) $A \in \mathsf{Reg}(\mathfrak{S}) \Rightarrow A^* \in \mathsf{Reg}(\mathfrak{S})$.

(7) Pour toute séquence $\langle\!\langle\, A_1, \ldots, A_n \,\rangle\!\rangle$ d'ensembles $A_i \in \mathsf{Reg}(\mathfrak{S})$, on a

$$\Big(\bigcup_{i=1}^{n} A_i\Big) \in \mathsf{Reg}(\mathfrak{S}); \qquad \Big(\prod_{i=1}^{n} A_i\Big) \in \mathsf{Reg}(\mathfrak{S}).$$

Démonstration. Ces propriétés découlent immédiatement de la définition de l'ensemble $\mathsf{Reg}(\mathfrak{S})$ (3.4.4) et de celle d'une construction génératrice de base $\mathfrak{S}$ dans $\mathcal{K}$ (3.4.1).

(1) La séquence $R = \langle\!\langle\, R_1 \,\rangle\!\rangle = \langle\!\langle\, \varnothing \,\rangle\!\rangle$ est une construction génératrice régulière de base $\mathfrak{S}$, donc l'ensemble $\varnothing$ est régulièrement engendré par $\mathfrak{S}$.

(2) La séquence $R = \langle\!\langle\, R_1, R_2 \,\rangle\!\rangle = \langle\!\langle\, \varnothing, \varnothing^* \,\rangle\!\rangle$ est une construction génératrice régulière de base $\mathfrak{S}$, donc l'ensemble $\varnothing^*$ est régulièrement engendré par $\mathfrak{S}$. Or $\varnothing^* = I_{\mathcal{K}}$ (3.2.7(1)).

(3) Si $A \in \mathfrak{S}$, la séquence $R = \langle\!\langle\, R_1 \,\rangle\!\rangle = \langle\!\langle\, A \,\rangle\!\rangle$ est une construction génératrice régulière de base $\mathfrak{S}$, donc A est régulièrement engendré par $\mathfrak{S}$.

(4) Supposons que A et B soient des ensembles régulièrement engendrés par $\mathfrak{S}$. Soit $R = \langle\!\langle\, R_1, \ldots, R_m \,\rangle\!\rangle$ une construction génératrice régulière de base $\mathfrak{S}$ telle que $A = R_m$ (il en existe une par hypothèse sur A), et soit $R' = \langle\!\langle\, R'_1, \ldots, R'_n \,\rangle\!\rangle$ une construction génératrice régulière de base $\mathfrak{S}$ telle que $B = R'_n$. La séquence

$$R'' = \langle\!\langle\, R_1, \ldots, R_m, R'_1, \ldots, R'_n, A \cup B \,\rangle\!\rangle = R \;\mathsf{cat}\; R' \;\mathsf{cat}\; \langle\!\langle\, A \cup B \,\rangle\!\rangle$$

est une construction génératrice régulière de base $\mathfrak{S}$, d'après 3.4.3(4) et du fait que $R''_{m+n+1} = A \cup B = R''_m \cup R''_{m+n}$. Donc $A \cup B$ est régulièrement engendré par $\mathfrak{S}$.

(5) On raisonne de la même manière pour l'ensemble AB.

(6) Supposons que A soit un ensemble régulièrement engendré par $\mathfrak{S}$ et soit $R = \langle\!\langle\, R_1, \ldots, R_n \,\rangle\!\rangle$ une construction génératrice régulière de base $\mathfrak{S}$ telle que $A = R_n$. La séquence $R' = \langle\!\langle\, R_1, \ldots, R_n, A^* \,\rangle\!\rangle$ est aussi une construction génératrice régulière de base $\mathfrak{S}$ puisque $R'_{n+1} = A^* = (R_n)^* = (R'_n)^*$. Donc A^* est régulièrement engendré par $\mathfrak{S}$.

(7) Voir [A].

3.4.6. Langages réguliers sur un alphabet X

Nous rappelons que les *langages élémentaires* sur un alphabet X (3.1.2, Exemple 4) sont les langages de la forme $\{\langle\!\langle\, a \,\rangle\!\rangle\}$ tels que $a \in X$. Ce sont les langages qui ne comportent qu'un seul mot, celui-ci étant de longueur 1.

Définition. On appelle *langages réguliers* sur X les langages qui sont régulièrement engendrés, dans l'algèbre de Kleene des langages sur X, par l'ensemble des langages élémentaires sur X.

Cette définition peut se traduire par la formule :

$$(A \text{ est un langage régulier sur } X) \Leftrightarrow \left(\begin{array}{l} A \in \mathsf{Reg}(\mathfrak{S}), \\ \text{où } \mathfrak{S} \text{ est l'ensemble des langages} \\ \text{élémentaires sur } X. \end{array}\right)$$

En explicitant encore cette définition d'après la définition de l'ensemble $\mathrm{Reg}(\mathfrak{S})$ (3.4.4) et celle d'une construction génératrice régulière de base $\mathfrak{S}$ (3.4.1), on peut préciser ceci :

Un langage A sur X est (par définition) un langage régulier sur X s'il existe une séquence $\langle\!\langle\, R_1, \ldots, R_n \,\rangle\!\rangle$ de langages sur X telle que $A = R_n$ et telle que, pour chaque $k \in [1 \,..\, n]$, l'une des conditions suivantes soit vérifiée :

(a) R_k est un langage élémentaire sur X;

(b) $R_k = \varnothing$;

(c) Il existe deux entiers $i, j \in [1 \,..\, k-1]$ tels que $R_k = R_i \cup R_j$;

(d) Il existe deux entiers $i, j \in [1 \,..\, k-1]$ tels que $R_k = R_i R_j$;

(e) Il existe un entier $i \in [1 \,..\, k-1]$ tel que $R_k = R_i^*$.

Une telle séquence de langages est une construction génératrice régulière de base $\mathfrak{S}$, où $\mathfrak{S}$ est l'ensemble des langages élémentaires sur X (3.4.1). On l'appelle plus simplement une *construction génératrice de langages réguliers* sur X, et l'on dit aussi qu'elle est une construction génératrice du langage régulier $A = R_n$.

Moins formellement, on peut dire qu'un langage régulier sur X est un langage qui peut être construit à partir de langages élémentaires sur X et/ou du langage $\varnothing$, en effectuant un nombre fini d'opérations de réunion ou de produit de deux langages, ou de fermeture de Kleene d'un langage.

3.4.7. Théorèmes

En appliquant les théorèmes 3.4.5 dans l'algèbre de Kleene des langages sur X et à l'ensemble $\mathfrak{S}$ des langages élémentaires sur X, on obtient les propositions suivantes sur les langages réguliers sur X :

(1) Le langage $\varnothing$ est un langage régulier sur X.

(2) Le langage neutre $\{\Lambda\}$ est un langage régulier sur X.

(3) Pour tout $a \in X$, le langage élémentaire $\{\langle\!\langle\, a \,\rangle\!\rangle\}$ est un langage régulier sur X.

(4) Si A et B sont des langages réguliers sur X, $A \cup B$ est un langage régulier sur X.

(5) Si A et B sont des langages réguliers sur X, AB est un langage régulier sur X.

(6) Si A est un langage régulier sur X, A^* est un langage régulier sur X.

(7) Pour toute séquence $\langle\!\langle\, A_1, \ldots, A_n \,\rangle\!\rangle$ de langages réguliers sur X, les langages

$$\Big(\bigcup_{i=1}^{n} A_i\Big), \qquad \Big(\prod_{i=1}^{n} A_i\Big)$$

sont des langages réguliers sur X.

Corollaires :

(8) Pour toute séquence $x \in \mathrm{W}(X)$, le langage $\{x\}$ est un langage régulier sur X.

(9) Tout langage fini sur X est un langage régulier sur X.

Démonstration. Le corollaire (8) peut se démontrer par récurrence sur la longueur des séquences, mais la démonstration par *induction structurelle* dans $\mathrm{W}(X)$ (2.2.7), en utilisant (2), (3), (5). est immédiate. En désignant par $P(x)$ la proposition « $\{x\}$ *est un langage régulier sur* X », on peut affirmer :

1° la proposition $P(\Lambda)$ est vraie d'après (2);

2° la proposition $P(\langle\!\langle\, a \,\rangle\!\rangle)$ est vraie pour tout $a \in X$ d'après (3);

3° si les propositions $P(x)$ et $P(y)$ sont vraies pour deux séquences x, y sur X, alors la proposition $P(xy)$ est vraie d'après (5), parce que le langage $\{xy\}$ est égal au produit de langages $\{x\}\{y\}$.

Pour le corollaire (9), on raisonne comme suit. Un langage fini L sur X est un langage qui peut s'écrire $L = \{x_1, \ldots, x_n\}$, où $x_1, \ldots, x_n$ sont n séquences sur X ($n \in \mathbb{N}$), avec $L = \varnothing$ comme cas particulier ($n = 0$). Un tel langage peut s'écrire aussi

$$L = \bigcup_{i=1}^{n} \{x_i\}.$$

Il est donc régulier d'après (8) et (7).

3.4.8. Expressions régulières

En raison de la définition 3.4.6, les langages élémentaires sur un alphabet X interviennent souvent dans les expressions qui désignent des langages réguliers sur X. La notation rigoureuse $\{\langle\!\langle\, \mathtt{a} \,\rangle\!\rangle\}$ pour un tel langage (avec $\mathtt{a} \in X$) est évidemment encombrante lorsqu'elle se répète souvent. C'est donc un abus de langage courant que d'écrire simplement $\mathtt{a}$ au lieu de $\{\langle\!\langle\, \mathtt{a} \,\rangle\!\rangle\}$.

On écrit d'ailleurs souvent aussi $\mathtt{a}$ au lieu de $\langle\!\langle\, \mathtt{a} \,\rangle\!\rangle$, de sorte que $\mathtt{a}$ peut désigner tantôt un élément de X, tantôt la séquence élémentaire correspondante $\langle\!\langle\, \mathtt{a} \,\rangle\!\rangle$, tantôt le langage élémentaire correspondant $\{\langle\!\langle\, \mathtt{a} \,\rangle\!\rangle\}$. Nous nous intéressons ici surtout au cas où le terme $\mathtt{a}$ est utilisé comme désignation du langage élémentaire $\{\langle\!\langle\, \mathtt{a} \,\rangle\!\rangle\}$. Dans ce cas, on parle du terme $\mathtt{a}$ comme étant *l'expression régulière* $\mathtt{a}$. Si $\mathtt{b}$ et $\mathtt{c}$ sont aussi des éléments de X, l'expression $\mathtt{abc}$, en tant qu'expression régulière, représente le produit de langages élémentaires

$$\{\langle\!\langle\, \mathtt{a} \,\rangle\!\rangle\}\{\langle\!\langle\, \mathtt{b} \,\rangle\!\rangle\}\{\langle\!\langle\, \mathtt{c} \,\rangle\!\rangle\},$$

à savoir le langage $\{\langle\!\langle\, \mathtt{a}, \mathtt{b}, \mathtt{c} \,\rangle\!\rangle\}$.

De façon générale, si les symboles $\mathtt{a}, \mathtt{b}, \mathtt{c}$, etc. désignent des éléments d'un alphabet X, on appelle *expressions régulières* construites avec ces symboles les expressions qui représentent des langages réguliers sur X au moyen de ces symboles — interprétés comme représentant les langages élémentaires correspondants — et au moyen d'opérateurs de réunion, de produit, de fermeture de Kleene, et éventuellement du symbole $\varnothing$.

Exemple. L'expression

$$\mathtt{a}(\mathtt{b} \cup \mathtt{c})^* \tag{1}$$

est une expression régulière construite avec les symboles $\mathtt{a}$, $\mathtt{b}$, $\mathtt{c}$. Si ces symboles désignent des éléments d'un alphabet X et si on les interprète aussi comme les langages élémentaires $\{\langle\!\langle\, \mathtt{a} \,\rangle\!\rangle\}$, $\{\langle\!\langle\, \mathtt{b} \,\rangle\!\rangle\}$, $\{\langle\!\langle\, \mathtt{c} \,\rangle\!\rangle\}$ correspondants, l'expression représente le langage

$$L = \{\langle\!\langle\, \mathtt{a} \,\rangle\!\rangle\}\big(\{\langle\!\langle\, \mathtt{b} \,\rangle\!\rangle\} \cup \{\langle\!\langle\, \mathtt{c} \,\rangle\!\rangle\}\big)^*.$$

Un mot z appartient à ce langage si et seulement s'il admet une décomposition

$$z = xy \quad \text{avec} \quad x \in \{\langle\!\langle\, \mathtt{a} \,\rangle\!\rangle\} \ \text{ et } \ y \in \big(\{\langle\!\langle\, \mathtt{b} \,\rangle\!\rangle\} \cup \{\langle\!\langle\, \mathtt{c} \,\rangle\!\rangle\}\big)^*.$$

Or $x \in \{\langle\!\langle\, \mathtt{a} \,\rangle\!\rangle\}$ équivaut à $x = \langle\!\langle\, a \,\rangle\!\rangle$, et $y \in \big(\{\langle\!\langle\, \mathtt{b} \,\rangle\!\rangle\} \cup \{\langle\!\langle\, \mathtt{c} \,\rangle\!\rangle\}\big)^*$ signifie (3.3.2(1)) que y admet une décomposition en n séquences

$$y = w_1 \cdots w_n$$

avec $\quad n \geq 0$ et $\forall i \in [1 \,..\, n] : w_i \in \{\langle\!\langle\, \mathtt{b} \,\rangle\!\rangle\} \cup \{\langle\!\langle\, \mathtt{c} \,\rangle\!\rangle\}$.

Donc une séquence z appartient à L si et seulement si elle est de la forme

$$z = \langle\!\langle\, a \,\rangle\!\rangle w_1 \cdots w_n$$

avec $n \geq 0$ et $w_i = \langle\!\langle\, b \,\rangle\!\rangle$ ou $w_i = \langle\!\langle\, c \,\rangle\!\rangle$ pour tout $i \in [1 \,..\, n]$. On peut donc, finalement, caractériser le langage L représenté par l'expression régulière (1) comme suit :

$$z \in L \Leftrightarrow \begin{pmatrix} \lg(z) \geq 1 \text{ et } z[1] = a \text{ et} \\ \forall k \in [2 \,..\, \lg(z)] : z[k] = b \text{ ou } z[k] = c \end{pmatrix}.$$

Expressions régulières et constructions génératrices régulières. Une expression régulière bien formée doit montrer que le langage représenté est régulier en fournissant immédiatement, par sa syntaxe, une construction génératrice régulière du langage qu'elle représente. Par exemple, si a, b, c sont des éléments d'un alphabet X, l'expression (1) est une expression régulière bien formée, car on a, pour le langage représenté par cette expression, la construction génératrice régulière $\langle\!\langle\, R_1, \ldots, R_6 \,\rangle\!\rangle$ suivante :

$$\begin{array}{ll} R_1 = \mathtt{a} & \text{langage élémentaire } \{\langle\!\langle\, \mathtt{a} \,\rangle\!\rangle\} \\ R_2 = \mathtt{b} & \text{langage élémentaire } \{\langle\!\langle\, \mathtt{b} \,\rangle\!\rangle\} \\ R_3 = \mathtt{c} & \text{langage élémentaire } \{\langle\!\langle\, \mathtt{c} \,\rangle\!\rangle\} \\ R_4 = \mathtt{b} \cup \mathtt{c} & = R_2 \cup R_3 \\ R_5 = (\mathtt{b} \cup \mathtt{c})^* & = R_4^* \\ R_6 = \mathtt{a}(\mathtt{b} \cup \mathtt{c})^* & = R_1 L_5. \end{array}$$

3.4.9. Exercice

Donner une expression régulière construite avec les symboles 0, 1, 2 pour chacun des langages L_1 à L_{10} sur l'alphabet $X = \{0, 1, 2\}$ décrits ci-dessous.

- L_1 : ensemble des mots qui se terminent par 100, y compris le mot 100 lui-même.
- L_2 : ensemble des mots qui commencent par 100.
- L_3 : ensemble des mots qui comportent au moins un 2.
- L_4 : ensemble des mots qui comportent un 2 et un seul.
- L_5 : ensemble des mots qui ne comportent aucun 2.
- L_6 : ensemble des mots de longueur ≥ 3.
- L_7 : ensemble des mots qui comportent un nombre pair de 1 (y compris zéro).
- L_8 : ensemble des mots qui comportent un nombre impair de 1.
- L_9 : ensemble des mots qui ne comportent aucune syllabe 01, c'est-à-dire dans lesquels un 0 n'est jamais suivi d'un 1.
- L_{10} : ensemble des mots qui comportent une syllabe 01 et une seule.

3.4.10. Démonstrations par induction structurelle dans $\mathsf{Reg}(\mathfrak{S})$

Nous considérons à nouveau une algèbre de Kleene $\mathcal{K}$ quelconque et un ensemble $\mathfrak{S} \subset \mathcal{K}$. Comme précédemment, le lecteur peut imaginer, pour fixer les idées, que $\mathcal{K}$ est l'algèbre de Kleene $\mathcal{L}(X)$ des langages sur un ensemble X, et que $\mathfrak{S}$ est un ensemble de langages particuliers sur X, par exemple l'ensemble des langages élémentaires sur X. Mais ce qui suit est indépendant de cette interprétation.

Soit $P(A)$ une assertion à propos d'un ensemble quelconque $A \in \mathcal{K}$, et supposons que l'on veuille montrer que cette assertion est vraie pour tout ensemble A régulièrement engendré par $\mathfrak{S}$ (3.4.4). Il s'agit donc de démontrer une proposition de la forme

$$(1) \qquad \forall A \in \mathsf{Reg}(\mathfrak{S}) : \quad P(A).$$

Schéma de déduction. Pour démontrer (1), il suffit de montrer que, quels que soient les ensembles A, $B \in \mathcal{K}$, les cinq propositions suivantes sont vraies :

(a) $A \in \mathfrak{S} \Rightarrow P(A)$;

(b) $P(\varnothing)$;

(c) $(P(A) \text{ et } P(B)) \Rightarrow P(A \cup B)$;

(d) $(P(A) \text{ et } P(B)) \Rightarrow P(AB)$;

(e) $P(A) \Rightarrow P(A^*)$.

Autrement dit, la conjonction de ces cinq propositions (dûment munies de quantificateurs $\forall A \in \mathcal{K}$, $\forall B \in \mathcal{K}$) implique (1).

La validité de ce schéma de déduction devrait être intuitivement claire. Les ensembles $A \in \mathcal{K}$ qui appartiennent au sous-ensemble $\mathrm{Reg}(\mathfrak{S}) \subset \mathcal{K}$ sont tous les ensembles $A \in \mathcal{K}$ qui peuvent être construits au moyen de constructions génératrices régulières de base $\mathfrak{S}$, c'est-à-dire construits à partir d'ensembles appartenant à $\mathfrak{S}$ et/ou de l'ensemble $\varnothing$ par un nombre fini d'opérations de construction qui sont les opérations de réunion, de produit et de fermeture de Kleene. Les propositions (c), (d), (e) signifient que ces opérations *conservent la propriété* P, en ce sens que si on les applique à des ensembles A, B qui vérifient P, on obtient des ensembles $A \cup B$, AB, A^* qui vérifient P. Donc, si les ensembles qui appartiennent à $\mathfrak{S}$ et l'ensemble $\varnothing$ vérifient P, ce qu'expriment (a) et (b), tous les ensembles régulièrement engendrés par $\mathfrak{S}$ vérifient P.

Ce raisonnement est développé de manière plus formelle dans la démonstration du schéma donnée dans [A].

Commentaire. Lorsqu'on démontre une proposition de la forme (1) suivant le schéma de déduction ci-dessus, c'est-à-dire en démontrant les cinq propositions (a) – (e), on dit que l'on démontre (1) par *induction structurelle* dans $\mathrm{Reg}(\mathfrak{S})$. La signification générale de ces mots a été décrite au §2.2.7. L'ensemble $\mathrm{Reg}(\mathfrak{S})$ possède d'une part des éléments de base, à savoir les ensembles $A \in \mathfrak{S}$ et l'ensemble $\varnothing$. Il est muni d'autre part de certaines opérations : réunion, produit, fermeture de Kleene. Tous ses éléments peuvent être engendrés à partir des éléments de base en appliquant ces opérations un nombre fini de fois. Si les éléments de base de $\mathrm{Reg}(\mathfrak{S})$ possèdent une certaine propriété et si les opérations en question conservent cette propriété, alors tous les éléments de l'ensemble $\mathrm{Reg}(\mathfrak{S})$ possèdent cette propriété.

3.4.11. Exemple

Nous nous plaçons dans l'algèbre de Kleene $\mathcal{L}(X)$ des langages sur un ensemble X quelconque et nous considérons la fonction $\rho : \mathrm{W}(X) \to \mathrm{W}(X)$ dite de renversement des séquences sur X (2.2.3). Comme application du schéma de déduction 3.4.10, nous allons démontrer le théorème suivant : pour tout langage régulier A sur X, le transformé $\rho\langle A\rangle$ de l'ensemble A par la fonction ρ est un langage régulier sur X.

Nous précisons d'abord que l'ensemble $\rho\langle A\rangle$, pour un langage A sur X, c'est-à-dire un ensemble $A \subset \mathrm{W}(X) = \mathrm{Dom}\,\rho$, est caractérisé d'après 1.3.24(2) par la formule

$$(1) \qquad \forall y : \quad y \in \rho\langle A\rangle \Leftrightarrow \exists x(x \in A \text{ et } y = \rho(x)).$$

Il revient au même d'écrire

$$\rho\langle A\rangle = \{\rho(x) \mid x \in A\} \qquad (1.2.3, \text{Déf. } 1).$$

L'ensemble des langages réguliers sur X est l'ensemble $\mathrm{Reg}(\mathfrak{S})$ où $\mathfrak{S}$ est l'ensemble des langages élémentaires sur X (3.4.6). Il s'agit donc de démontrer, pour cet ensemble $\mathfrak{S}$

particulier, que

$$\forall A \in \mathsf{Reg}(\mathfrak{S}) : \quad \underbrace{\rho\langle A\rangle \in \mathsf{Reg}(\mathfrak{S})}_{P(A)}.$$

D'après 3.4.10, il nous suffit pour cela de démontrer les propositions (a) – (e) suivantes :

(a) $\forall A \in \mathfrak{S} : P(A)$.
En français : si A est un langage élémentaire quelconque sur X, alors $\rho\langle A\rangle$ est un langage régulier sur X. Preuve : si A est un langage élémentaire sur X, disons $A = \{\langle\!\langle\, a \,\rangle\!\rangle\}$, alors $\rho\langle A\rangle = A$, donc $\rho\langle A\rangle$ est un langage élémentaire sur X, donc un langage régulier sur X (3.4.7(3)).

(b) $P(\varnothing)$.
En français : le langage $\rho\langle\varnothing\rangle$ est un langage régulier sur X. Preuve : $\rho\langle\varnothing\rangle = \varnothing$ d'après 1.3.23(2), et l'on conclut d'après 3.4.7(1).

(c) $(P(A)$ et $P(B)) \Rightarrow P(A \cup B)$.
En français : si, pour deux langages A, B sur X, les langages $\rho\langle A\rangle$ et $\rho\langle B\rangle$ sont des langages réguliers sur X, alors le langage $\rho\langle A \cup B\rangle$ est un langage régulier sur X. Cela découle, d'après 3.4.7(4), de la formule

$$\rho\langle A \cup B\rangle = \rho\langle A\rangle \cup \rho\langle B\rangle,$$

qui est une propriété générale du transformé d'un ensemble par une relation (théorème 4 de 1.3.23).

(d) $(P(A)$ et $P(B)) \Rightarrow P(AB)$.
En français : si, pour deux langages A, B sur X, les langages $\rho\langle A\rangle$ et $\rho\langle B\rangle$ sont des langages réguliers sur X, alors le langage $\rho\langle AB\rangle$ est un langage régulier sur X. Cela découle, d'après 3.4.7(5), de la formule plus ou moins évidente

$$\rho\langle AB\rangle = \rho\langle B\rangle\rho\langle A\rangle,$$

qui sera démontrée dans l'exercice 3.4.12.

(e) $P(A) \Rightarrow P(A^*)$.
En français : si, pour un langage A sur X, le langage $\rho\langle A\rangle$ est un langage régulier sur X, alors le langage $\rho\langle A^*\rangle$ est un langage régulier sur X. Cela découle, d'après 3.4.7(6), de la formule plus ou moins évidente

$$\rho\langle A^*\rangle = (\rho\langle A\rangle)^*,$$

qui sera démontrée dans l'exercice 3.4.12.

3.4.12. Exercice

Démontrer les formules

(1) $$\rho\langle AB\rangle = \rho\langle B\rangle\rho\langle A\rangle$$

(2) $$\rho\langle A^*\rangle = (\rho\langle A\rangle)^*$$

utilisées dans 3.4.11, pour des langages A, B sur X et la fonction de renversement des séquences sur X.

Indications. La formule 3.4.11(1) est la formule principale caractérisant $\rho\langle A\rangle$, pour un langage A sur X. Pour démontrer (1), on utilisera 2.2.4(1) et les formules (3), (4), (5)

suivantes, qu'on admettra sans démonstration, pour des séquences x, $y \in \mathrm{W}(X)$ quelconques et pour un langage $A \subset \mathrm{W}(X)$ quelconque :

(3) $\rho(\rho(x)) = x$.

(4) $y \in \rho\langle A\rangle \Leftrightarrow \rho(y) \in A$.

(5) $x \in A \Leftrightarrow \rho(x) \in \rho\langle A\rangle$.

Pour (2), il faut démontrer au préalable, par récurrence sur n, que $\rho\langle A^n\rangle = (\rho\langle A\rangle)^n$ pour tout $n \in \mathbb{N}$, en utilisant (1). Ensuite on utilise la définition de A^* (3.2.5), et le théorème 4 de 1.3.23.

Chapitre 4 Automates finis

4.1 Relations de transition

4.1.1. Définition d'un automate fini

Un *automate fini A sur un alphabet X* est un objet (mathématique) qui se compose des données suivantes :

(a) un ensemble fini Q^A, dont les éléments sont appelés les *états* de A;

(b) un ensemble fini T^A, tel que $T^A \subset Q^A \times X \times Q^A$. Les éléments de cet ensemble sont appelés les *transitions* de A. Les transitions de A sont donc des triplets (p, a, q) tels que $p \in Q^A$, $a \in X$, $q \in Q^A$.

Un automate fini A sur X est *caractérisé* par ces deux ensembles. On le considère donc comme le couple formé par ces deux ensembles, ce pourquoi l'on écrit $A = (Q^A, T^A)$. L'exposant A des notations Q^A, T^A peut être omis lorsqu'il n'y a pas de confusion possible sur l'automate dont il est question.

4.1.2. Exemple

Soient a, b, c des éléments distincts d'un alphabet X. Les données ci-dessous définissent un automate fini $A = (Q^A, T^A)$ sur X. La *représentation graphique* de cet automate est donnée dans la figure 4.1. Le principe de cette représentation devrait être clair. Les états sont représentés par les sommets d'un diagramme. Une flèche allant d'un somet p à un sommet q et portant un élément a de X représente la transition (p, a, q). On peut représenter plusieurs transitions (p, a, q), (p, a', q), etc. par une seule flèche allant de p à q et portant les symboles $a, a', \ldots$

Ensemble d'états :

$Q^A = \{1, 2, 3, 4, 5\}$.

Ensemble de transitions :

$$T^A = \begin{array}{|l|}\hline (1, a, 1) \\ (1, a, 2) \\ (1, b, 3) \\ (2, c, 1) \\ (3, c, 4) \\ (4, b, 2) \\ (4, a, 5) \\ (4, b, 5) \\ \hline \end{array}$$

Représentation graphique :

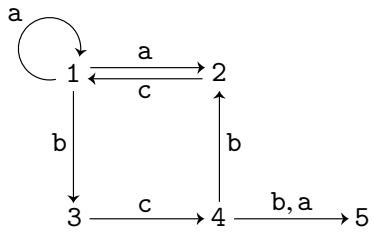


Figure 4.1

4.1.3. États de départ et d'arrivée, étiquette d'une transition

Si un triplet $\alpha = (p, a, q)$ est une transition d'un automate fini, on dit que les états p et q sont respectivement l'*état de départ* et *l'état d'arrivée* de α, et que cette transition *va de l'état p à l'état q*. On dit encore que l'élément médian a est *l'étiquette* ou *le label* de α.

4.1.4. Commentaires sur la définition d'un automate fini

Lorsque nous parlons d'un automate $A = (Q^A, T^A)$ *sur* un ensemble X, cela signifie, d'après la définition 4.1.1, que pour chaque transition (p, x, q) de A l'étiquette x appartient à l'ensemble X. Mais cela ne signifie pas que pour tout élément x de X il existe une transition

$(p, x, q) \in T$ avec cette étiquette. Si X' est un ensemble tel que $X \subset X'$, un automate fini sur X est aussi un automate fini sur X'.

En particulier, l'ensemble X peut être *infini*. Dans l'expression *automate fini sur* X, l'adjectif « fini » se réfère seulement à l'ensemble des états et à l'ensemble des transitions de l'automate, qui doivent être des ensembles finis. Seul un sous-ensemble fini de X est donc utilisé dans les transitions de A. Par exemple, les étiquettes `a`, `b`, `c` des transitions de l'automate 4.1.2 pourraient être des nombres réels et dans ce cas, on pourrait parler d'un automate fini sur $\mathbb{R}$.

Il ne faut pas confondre un automate fini avec sa représentation graphique (fig. 4.1). Un automate fini est un objet mathématique. Dans le langage des informaticiens, on peut dire qu'il s'agit d'une « structure de donnée ».

4.1.5. Relations de transition d'un automate

Définition 1. Soient X un ensemble (alphabet) et $A = (Q^A, T^A)$ un automate fini sur X. Pour tout $x \in X$, nous posons

$$T_x^A = \{(p, q) \mid (p, x, q) \in T^A\}. \tag{1}$$

Cet ensemble de couples est une relation dans l'ensemble d'états Q^A (§1.3.3). Nous l'appelons la *relation de transition élémentaire* de A correspondant à x.

Par définition de T_x^A (1), on a, pour tout $x \in X$:

(2) $(p, q) \in T_x^A \Leftrightarrow (p, x, q) \in T^A$. (Voir théorème 3 de 1.2.3)

(3) $T_x^A \subset Q^A \times Q^A$.

La relation de transition élémentaire T_x^A est vide, $T_x^A = \varnothing$, lorsque x est un élément de X pour lequel il n'y a pas de transition (p, x, q) d'étiquette x dans l'ensemble T^A.

Exemple 1. Nous considérons l'automate A du §4.1.2, où `a`, `b`, `c` sont trois éléments distincts d'un ensemble X, par ailleurs indéterminé. Les relations de transition élémentaires $T_{\mathtt{a}}^A$, $T_{\mathtt{b}}^A$, $T_{\mathtt{c}}^A$ de cet automate sont représentées par les trois diagrammes de la figure 4.2. Ce sont les ensembles de couples

$$T_{\mathtt{a}} = \{1 \mapsto 1, 1 \mapsto 2, 4 \mapsto 5\};$$
$$T_{\mathtt{b}} = \{1 \mapsto 3, 4 \mapsto 2, 4 \mapsto 5\};$$
$$T_{\mathtt{c}} = \{2 \mapsto 1, 3 \mapsto 4\}.$$

On a $T_x^A = \varnothing$ pour tout élément x de X différent de `a`, `b`, `c`.

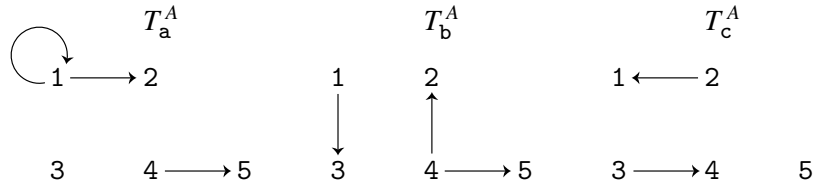


Figure 4.2

Relations de transition $T_{\mathtt{a}}^A$, $T_{\mathtt{b}}^A$, $T_{\mathtt{c}}^A$ de l'automate de la figure 4.1.

Définition 2. Soient X un ensemble et $A = (Q^A, T^A)$ un automate fini sur X. Pour toute séquence $x = \langle\!\langle\, x[1], \ldots, x[n] \,\rangle\!\rangle \in \mathrm{W}(X)$, nous posons

$$\Theta_x^A = T_{x[1]}^A \cdots T_{x[n]}^A = \prod_{i=1}^{\lg(x)} T_{x[i]}^A. \tag{4}$$

Le membre de droite de cette égalité est le produit de composition des relations de transition élémentaires $T^A_{x[i]}$. Il s'agit donc d'une relation dans l'ensemble Q^A. La relation Θ^A_x est appelée la *relation de transition de A correspondant à la séquence x*.

Si $x = \Lambda$ ($n = 0$), le membre de droite de (4) est un produit de zéro facteurs. S'agissant de l'opération produit du monoïde $\mathcal{R}(Q^A)$, le produit de zéro facteurs est par définition l'élément neutre de ce monoïde, à savoir la relation identique Id_{Q^A}. Nous admettons donc que, par définition, pour toute séquence $x \in \mathrm{W}(X)$:

$$\Theta^A_x = \begin{cases} \mathrm{Id}_{Q^A} & \text{si } x = \Lambda; \\ T^A_{x[1]}\Theta^A_{x\downarrow} & \text{si } x \neq \Lambda. \end{cases} \tag{5}$$

Exemple 2. (Suite de l'Exemple 1) Les diagrammes de la figure 4.2, décrivant les relations de transition élémentaires $T^A_{\mathtt{a}}$, $T^A_{\mathtt{b}}$, $T^A_{\mathtt{c}}$, concernant l'automate A du §4.1.2, nous permettent de calculer facilement la relation de transition Θ^A_x correspondant à la séquence $x = \langle\!\langle\, \mathtt{b}, \mathtt{c}, \mathtt{a} \,\rangle\!\rangle$:

$$\Theta^A_x = T^A_{\mathtt{b}} T^A_{\mathtt{c}} T^A_{\mathtt{a}} = \{1 \mapsto 5, 4 \mapsto 1, 4 \mapsto 2\}. \tag{6}$$

La manière de calculer cette relation dans Q^A est fournie par le théorème 2.3.10 sur le produit de composition d'une séquence de relations, qui est expliqué au §2.3.9. La relation Θ^A_x est le produit de composition de la séquence de relations

$$R = \langle\!\langle\, R[1], R[2], R[3] \,\rangle\!\rangle = \langle\!\langle\, T^A_{\mathtt{b}}, T^A_{\mathtt{c}}, T^A_{\mathtt{a}} \,\rangle\!\rangle,$$

autrement dit l'évaluation de cette séquence dans le monoïde $\mathcal{R}(Q^A)$ des relations dans Q^A. D'après le théorème 2.3.10, un couple (p, q) appartient à la relation

$$\Theta^A_x = R[1]R[2]R[3] = eval_{\mathcal{R}(Q^A)}(R)$$

si et seulement s'il existe une séquence

$$s = \langle\!\langle\, s[1], s[2], s[3], s[4] \,\rangle\!\rangle \in \mathrm{W}(Q^A)$$

(séquence d'états de A) telle que :

$$\begin{cases} s[1] = p; \\ s[4] = q; \\ (s[1], s[2]) \in T^A_b; \quad (s[2], s[3]) \in T^A_c; \quad (s[3], s[4]) \in T^A_a. \end{cases} \tag{7}$$

Par exemple, le couple d'états $(p, q) = (4, 2)$ appartient à Θ^A_x parce que la séquence d'états $s = \langle\!\langle\, 4, 2, 1, 2 \,\rangle\!\rangle$ vérifie ces conditions.

Par définition des relation de transition élémentaires de A (2), la troisième ligne des conditions (7) peut s'écrire

$$(s[1], b, s[2]) \in T^A; \quad (s[2], c, s[3]) \in T^A; \quad (s[3], a, s[4]) \in T^A. \tag{8}$$

On note aussi ces trois conditions sous la forme du diagramme suivant, qui se rapporte à celui de l'automate considéré (figure 4.1) :

$$s[1] \xrightarrow{\ \mathtt{b}\ } s[2] \xrightarrow{\ \mathtt{c}\ } s[3] \xrightarrow{\ \mathtt{a}\ } s[4]. \tag{9}$$

Pour déterminer la relation $\Theta^A_x = \Theta^A_{\langle\!\langle \mathtt{b},\mathtt{c},\mathtt{a} \rangle\!\rangle}$ (6), on cherche donc toutes les séquences $s \in \mathrm{W}_4(Q^A)$ qui vérifient (8), ou graphiquement (9), et l'on prend pour chacune d'elles le couple $(s[1], s[4])$.

4.1.6. Exercice

Déterminer les relations de transition $\Theta^A_{\langle\!\langle \mathsf{b,c,b,c} \rangle\!\rangle}$ et $\Theta^A_{\langle\!\langle \mathsf{b,c,b} \rangle\!\rangle}$ de l'automate A de la figure 4.1.

4.1.7. Théorèmes

Soient X un ensemble et $A = (Q^A, T^A)$ un automate fini sur X. On a

(1) $\forall x \in \mathrm{W}(X) : \quad \Theta^A_x \subset Q^A \times Q^A$.

(2) $\Theta^A_\Lambda = \mathrm{Id}_{Q^A}$.

(3) $\forall a \in X : \quad \Theta^A_{\langle\!\langle a \rangle\!\rangle} = T^A_a$.

(4) $\forall x \in \mathrm{W}(X) : \forall y \in \mathrm{W}(X) : \quad \Theta^A_{xy} = \Theta^A_{x \,\mathsf{cat}\, y} = \Theta^A_x \Theta^A_y$ (produit de relations).

Les propositions (1), (2) et (4) peuvent se résumer simplement en disant que la fonction $\Theta^A = (\lambda x \in \mathrm{W}(X) : \Theta^A_x)$ est un homomorphisme du monoïde $\mathrm{W}(X)$ dans le monoïde $\mathcal{R}(Q^A)$ des relations dans Q^A.

Démonstration. Par définition (4.1.5(4)), pour toute séquence $x \in \mathrm{W}(X)$, Θ^A_x est un produit de relations dans Q^A, donc une relation dans Q^A, d'où (1). La propriété (2) fait partie de la définition de Θ^A_x (4.1.5(5)). La proposition (3) découle aussi de 4.1.5(5): $\Theta^A_{\langle\!\langle a \rangle\!\rangle} = T^A_{\langle\!\langle a \rangle\!\rangle[1]} \Theta^A_{\langle\!\langle a \rangle\!\rangle\downarrow} = T^A_a \Theta^A_\Lambda = T^A_a \mathrm{Id}_{Q^A} = T^A_a$. D'après 4.1.5(5) et d'après (3), on a pour tout $x \in \mathrm{W}(X)$:

$$\Theta^A_x = \begin{cases} \mathrm{Id}_{Q^A} & \text{si } x = \Lambda; \\ \Theta^A_{\langle\!\langle x[1] \rangle\!\rangle} \Theta^A_{x\downarrow} & \text{si } x \neq \Lambda. \end{cases}$$

On peut en déduire, d'après le théorème de 2.3.1, que la fonction $\Theta^A = (\lambda x \in \mathrm{W}(X) : \Theta^A_x)$ est un homomorphisme du monoïde $\mathrm{W}(X)$ dans le monoïde $\mathcal{R}(Q^A)$, d'où (4).

4.1.8. Théorème

Soient p, q deux états ($p, q \in Q^A$) d'un automate $A = (Q^A, T^A)$ sur X et soit $x = \langle\!\langle\, x[1], \ldots, x[n] \,\rangle\!\rangle$ une séquence de longueur n sur X. Pour que

$$(p, q) \in \Theta^A_x \,,$$

il faut et il suffit qu'il existe une séquence d'états de A

$$s = \langle\!\langle\, s[1], \ldots, s[n+1] \,\rangle\!\rangle$$

de longueur $n + 1$ telle que:

(1)
$$\begin{cases} s[1] = p; \\ s[n+1] = q; \\ \text{pour tout } k \in [1 \,..\, n] : \quad (s[k], x[k], s[k+1]) \in T^A. \end{cases}$$

Ce théorème généralise ce que nous avons vu dans l'Exemple 2 de 4.1.5. Les conditions (1) s'expriment aussi en disant que l'on a dans l'automate A les transitions

$$p = s[1] \xrightarrow{\;x[1]\;} s[2] \xrightarrow{\;x[2]\;} \cdots \xrightarrow{\;x[n]\;} s[n+1] = q.$$

La démonstration du théorème consiste à remarquer simplement que $(p, q) \in \Theta^A_x$ équivaut par définition à $(p, q) \in T^A_{x[1]} \cdots T^A_{x[n]}$ et à appliquer le théorème 2.3.10 qui caractérise le produit de composition d'une séquence de relations, en tenant compte de ce que $(s[i], s[i+1]) \in T^A_{x[i]}$ équivaut à $(s[i], x[i], s[i+1]) \in T^A$ (4.1.5(2)).

4.1.9. Terminologie : x relie p à q dans A

Étant donné deux états p, q d'un automate $A = (Q^A, T^A)$ sur un ensemble X et une séquence $x \in \mathrm{W}(X)$, nous disons que cette séquence « relie l'état p à l'état q dans A » si le couple (p, q) appartient à la relation de transition Θ_x^A. Nous posons donc comme définition :

$$(\, x \text{ relie } p \text{ à } q \text{ dans } A\,) \;\Leftrightarrow\; (p, q) \in \Theta_x^A.$$

L'introduction de cette terminologie n'apporte pas de nouveau concept, mais seulement une manière de lire la proposition $(p, q) \in \Theta_x^A$.

Ainsi, dans l'exemple 2 de 4.1.5, nous avons vu que le couple d'états (4, 2) appartient à la relation de transition Θ_x^A de l'automate A considéré, correspondant à la séquence $x = \langle\!\langle\, \mathtt{b}, \mathtt{c}, \mathtt{a} \,\rangle\!\rangle$. On peut exprimer cela en disant que cette séquence x relie l'état 4 à l'état 2 dans cet automate.

Pour tout automate A on a $\Theta_\Lambda^A = \mathrm{Id}_{Q^A}$ (4.1.7(2)). Si p et q sont deux états de A, dire que la séquence Λ relie p à q dans A équivaut donc à dire que $p = q$, puisque cela équivaut à $(p, q) \in \mathrm{Id}_{Q^A}$. Autrement dit, dans un automate A, la séquence Λ relie chaque état à lui-même et seulement à lui-même.

4.2 Accepteurs finis

4.2.1. Langages associés aux couples d'états d'un automate

Soit (p, q) un couple d'états d'un automate A sur un ensemble X. Le langage sur X associé à ce couple d'états dans A, noté L_{pq}^A, est *l'ensemble des séquences x sur X qui relient p à q dans A* :

$$L_{pq}^A = \{x \in \mathrm{W}(X) \mid (p, q) \in \Theta_x^A\}.$$

Par définition, on a donc pour toute séquence $x \in \mathrm{W}(X)$:

$$x \in L_{pq}^A \Leftrightarrow (p, q) \in \Theta_x^A. \tag{1}$$

On peut omettre l'exposant A de L_{pq}^A lorsqu'il n'y a pas de confusion possible sur l'automate A dont il est question.

Comme cas particulier de la formule (1), puisque $\Theta_\Lambda^A = \mathrm{Id}_Q$, on a, pour deux états p, q quelconques de A :

$$\Lambda \in L_{pq}^A \Leftrightarrow p = q. \tag{2}$$

4.2.2. Exemple

Soit $X = \{\mathtt{a}, \mathtt{b}\}$ un ensemble à deux éléments distincts et A l'automate sur X représenté dans la figure 4.3, avec $Q^A = \{1, 2, 3, 4, 5\}$. Une expression régulière pour chacun des 25 langages L_{pq}^A $(p, q \in Q^A)$ de cet automate est donnée dans la figure 4.4. Il existe une méthode pour « calculer » ces expressions régulières mais, pour le moment, nous les admettons comme évidentes. Le fait que tous les langages du tableau soient des langages réguliers sur l'alphabet X n'est d'ailleurs pas un hasard. C'est une propriété générale des automates finis que nous démontrerons plus loin (4.3.2).

4.2.3. Accepteurs finis

On appelle *accepteur fini* sur un alphabet X un triplet (A, I, F) dans lequel A est un automate fini sur X et I, F sont deux sous-ensembles de l'ensemble d'états Q^A, appelés respectivement l'ensemble des *états initiaux* et l'ensemble des *états finaux* de l'accepteur.

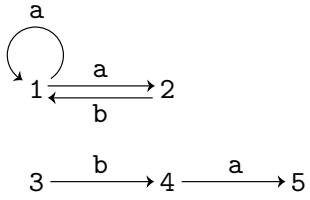


Figure 4.3

	$q=1$	2	3	4	5
$p=1$	$(\mathtt{a}\cup\mathtt{ab})^*$	$(\mathtt{a}\cup\mathtt{ab})^*\mathtt{a}$	$\varnothing$	$\varnothing$	$\varnothing$
2	$\mathtt{b}(\mathtt{a}\cup\mathtt{ab})^*$	$(\mathtt{ba}^*\mathtt{a})^*$	$\varnothing$	$\varnothing$	$\varnothing$
3	$\varnothing$	$\varnothing$	$\{\Lambda\}$	b	ba
4	$\varnothing$	$\varnothing$	$\varnothing$	$\{\Lambda\}$	a
5	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\{\Lambda\}$

Figure 4.4

Langages L^A_{pq} de l'automate A de la fig. 4.3

Ces deux sous-ensembles de Q^A peuvent être quelconques. En particulier ils peuvent être vides ou égaux à Q^A.

On dit qu'un tel triplet (A, I, F) est un automate « muni » d'un ensemble d'états initiaux et d'un ensemble d'états finaux. L'automate A est appelé *l'automate sous-jacent* de l'accepteur (A, I, F). Il arrive souvent que l'on désigne un accepteur (A, I, F) par le nom A de son automate sous-jacent, les données I et F étant sous-entendues.

4.2.4. Exemple

La figure 4.5 représente un accepteur fini (A, I, F) sur un alphabet $X = \{\mathtt{a}, \mathtt{b}\}$, dans lequel l'automate A est caractérisé par

$Q^A = \{1, 2, 3, 4, 5\}$, $T^A = \{(1, \mathtt{a}, 1), (1, \mathtt{a}, 2), (2, \mathtt{b}, 1), (3, \mathtt{b}, 4), (4, \mathtt{a}, 5)\}$

et les ensembles d'états initiaux et finaux sont $I = \{1, 3\}$ et $F = \{1, 5\}$. Les états initiaux sont désignés par une flèche épaisse qui pointe sur eux. Les états finaux par une flèche épaisse qui en sort. Une flèche bidirectionnelle pointe sur les états qui appartiennent à la fois à I et à F.

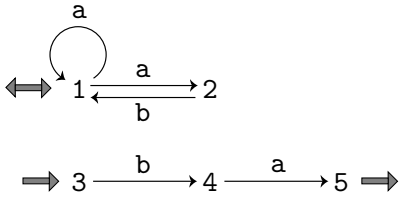


Figure 4.5

4.2.5. Langage d'un accepteur fini

Soit (A, I, F) un accepteur fini sur X. Le langage sur X que l'on dit *défini* par cet accepteur est l'ensemble des séquences X qui relient un état initial quelconque à un état final quelconque dans l'automate A. Nous désignons ce langage par L^A_{IF}. Sa définition formelle est

$$L^A_{IF} = \bigcup_{p\in I}\Big(\bigcup_{q\in F} L^A_{pq}\Big). \tag{1}$$

Par définition de la réunion d'une famille d'ensembles (1.3.19), et par définition des langages L^A_{pq} (4.2.1), on a

$$(2) \qquad x \in L^A_{IF} \Leftrightarrow \exists p \exists q (\, p \in I \text{ et } q \in F \text{ et } x \in L^A_{pq}\,) \Leftrightarrow \exists p \exists q (\, p \in I \text{ et } q \in F \text{ et } (p, q) \in \Theta^A_x\,).$$

On dit que les séquences $x \in L^A_{IF}$ sont les séquences *acceptées* ou par (A, I, F).

En appliquant la formule (2) à la séquence $x = \Lambda$ et en tenant compte de 4.2.1 (2), on obtient

$$(3) \qquad \Lambda \in L^A_{IF} \Leftrightarrow I \cap F \neq \varnothing.$$

Exemple. L'accepteur (A, I, F) de la figure 4.5 a pour automate sous-jacent A celui de la figure 4.3, dont les langages L^A_{pq} sont donnés dans la figure 4.4. Pour cet accepteur, on a, d'après 4.2.5 (1) :

$$\begin{aligned} L^A_{IF} &= L^A_{11} \cup L^A_{15} \cup L^A_{31} \cup L^A_{35} \\ &= L^A_{11} \cup \varnothing \cup \varnothing \cup L^A_{35} = (\mathtt{a} \cup \mathtt{ab})^* \cup \mathtt{ba}. \end{aligned}$$

4.2.6. Exercice

Pour chacun des langages L_i $(i = 1, \ldots, 10)$ de l'exercice 3.4.9, construire un accepteur fini minimal (nombre minimum d'états) sur l'alphabet $X = \{\mathtt{0}, \mathtt{1}, \mathtt{2}\}$ définissant le langage L_i. Les accepteurs proposés doivent permettre de voir facilement que les expressions régulières données pour ces dix langages dans la solution de l'exercice 3.4.9 sont correctes. Pour L_9, en particulier, on peut donner un accepteur à deux états montrant clairement que $L_9 = (\mathtt{1} \cup \mathtt{0}^*\mathtt{2})^*\mathtt{0}^*$.

4.2.7. Définition : langages L^A_{pF}

Si (A, I, F) est un accepteur fini sur X, nous posons pour chaque état p de A :

$$(1) \qquad L^A_{pF} = \bigcup_{q \in F} L^A_{pq}.$$

Par définition de la réunion d'une famille d'ensembles, on a donc

$$(2) \qquad x \in L^A_{pF} \Leftrightarrow \exists q (q \in F \text{ et } x \in L^A_{pq}) \Leftrightarrow \exists q (q \in F \text{ et } (p, q) \in \Theta^A_x).$$

Ainsi, le langage L^A_{pF} est l'ensemble des séquences sur X qui relient l'état p à un état final quelconque dans A. Le langage L^A_{IF} accepté par A (4.2.5 (1)) est la réunion des langages L^A_{pF} tels que $p \in I$:

$$(3) \qquad L^A_{IF} = \bigcup_{p \in I} L^A_{pF}.$$

En appliquant (2) à la séquence $x = \Lambda$, ainsi que 4.2.1 (2), on voit que

$$(4) \qquad \Lambda \in L^A_{pF} \Leftrightarrow p \in F.$$

Exemple. Pour l'accepteur (A, I, F) de la figure 4.5, on a, d'après la figure 4.4:

$$L^A_{1F} = L^A_{11} \cup L^A_{15} = (\mathtt{a} \cup \mathtt{ab})^* \cup \varnothing = (\mathtt{a} \cup \mathtt{ab})^*.$$
$$L^A_{2F} = L^A_{21} \cup L^A_{25} = \mathtt{b}(\mathtt{a} \cup \mathtt{ab})^* \cup \varnothing = \mathtt{b}(\mathtt{a} \cup \mathtt{ab})^*.$$
$$L^A_{3F} = L^A_{31} \cup L^A_{35} = \varnothing \cup \mathtt{ba} = \mathtt{ba}.$$
$$L^A_{4F} = L^A_{41} \cup L^A_{45} = \varnothing \cup \mathtt{a} = \mathtt{a}.$$
$$L^A_{5F} = L^A_{51} \cup L^A_{55} = \varnothing \cup \{\Lambda\} = \{\Lambda\}.$$

4.2.8. Notation

Si (A, I, F) est un accepteur fini sur X, nous posons pour tout $p \in Q$:

(1) $$N^A_{pF} = \{\Lambda\} \cap L^A_{pF}.$$

Les trois formules suivantes découlent de cette définition. La formule (2) découle de (1) d'après 4.2.7(4); (3) découle de (2); (4) découle de (3) parce que $I \cap F \neq \varnothing$ équivaut à « il existe un $p \in I$ tel que $p \in F$ ».

(2) $x \in N^A_{pF} \Leftrightarrow (x = \Lambda \text{ et } p \in F)$.

(3) $$N^A_{pF} = \begin{cases} \{\Lambda\} & \text{si } p \in F; \\ \varnothing & \text{si } p \notin F. \end{cases}$$

(4) $$\bigcup_{p \in I} N^A_{pF} = \begin{cases} \varnothing & \text{si } I \cap F = \varnothing; \\ \{\Lambda\} & \text{si } I \cap F \neq \varnothing. \end{cases}$$

4.2.9. Équations des langages L^A_{pF} d'un accepteur fini

Pour chaque couple d'états (p, q) d'un automate A sur X, nous désignons par E^A_{pq} l'ensemble des séquences *élémentaires* sur X qui relient p à q dans A. Autrement dit nous posons

$$E^A_{pq} = \{x \in L^A_{pq} \mid \lg(x) = 1\}.$$

Par définition, on a donc

(1) $$x \in E^A_{pq} \Leftrightarrow (x \in L^A_{pq} \text{ et } \lg(x) = 1).$$

On en déduit les formules suivantes (démonstration dans [A]):

(2) $\forall a \in X : \quad \langle\!\langle\, a \,\rangle\!\rangle \in E^A_{pq} \Leftrightarrow (p, a, q) \in T^A$.

(3) $E^A_{pq} \neq \varnothing \Leftrightarrow \exists a \in X : (p, a, q) \in T^A$.

Exemple. Pour l'automate fini A de la figure 4.3, les langages E^A_{pq} sont donnés dans le tableau de la figure 4.6. Les symboles a, b représentent naturellement les langages élémentaires $\{\langle\!\langle\, \mathtt{a} \,\rangle\!\rangle\}$, $\{\langle\!\langle\, \mathtt{b} \,\rangle\!\rangle\}$. Dans cet exemple, aucun des langages E^A_{pq} ne contient plus d'une séquence.

p \ q	1	2	3	4	5
1	a	a	∅	∅	∅
2	b	∅	∅	∅	∅
3	∅	∅	∅	b	∅
4	∅	∅	∅	∅	a
5	∅	∅	∅	∅	∅

Figure 4.6
Langages E^A_{pq} de l'automate de la figure 4.3.

Théorème. Soit (A, I, F) un accepteur fini sur X. Pour tout état $p \in Q^A$, on a

$$L^A_{pF} = \bigcup_{q \in Q^A} E^A_{pq} L^A_{qF} \cup N^A_{pF}. \tag{4}$$

Nous donnons ici une justification très informelle de ce théorème — une démonstration rigoureuse est donnée dans l'annexe [A]. Une séquence x appartient par définition au langage L^A_{pF} si elle relie l'état p à un état final (quelconque) dans A. Or cela peut se faire de deux manières. Ou bien x est la séquence vide, qui relie p à lui-même et à aucun autre état, et p appartient à F, ce qui revient à dire que $x \in N^A_{pF}$ (4.2.8(2)). Ou bien $x \neq \Lambda$, et alors la séquence $\langle\!\langle\, x[1] \,\rangle\!\rangle$ relie l'état p à un certain état $q \in Q^A$ lequel est relié lui-même à un état final par la séquence $x\!\downarrow$. Cela revient à dire que l'on a $\langle\!\langle\, x[1] \,\rangle\!\rangle \in E^A_{pq}$ et $x\!\downarrow \in L^A_{qF}$, donc $x \in E^A_{pq} L^A_{qF}$, pour un certain état $q \in Q^A$.

4.2.10. Exemple

Considérons l'accepteur (A, I, F) de la figure 4.5. Si nous écrivons l'équation 4.2.9(4) pour chacun des états $p \in Q^A = \{1, 2, 3, 4, 5\}$ de cet accepteur, nous obtenons le système d'équations

$$\begin{aligned}
L^A_{1F} &= E^A_{11} L^A_{1F} \cup E^A_{12} L^A_{2F} \cup E^A_{13} L^A_{3F} \cup E^A_{14} L^A_{4F} \cup E^A_{15} L^A_{5F} \cup N^A_{1F} \\
L^A_{2F} &= E^A_{21} L^A_{1F} \cup E^A_{22} L^A_{2F} \cup E^A_{23} L^A_{3F} \cup E^A_{24} L^A_{4F} \cup E^A_{25} L^A_{5F} \cup N^A_{2F} \\
L^A_{3F} &= E^A_{31} L^A_{1F} \cup E^A_{32} L^A_{2F} \cup E^A_{33} L^A_{3F} \cup E^A_{34} L^A_{4F} \cup E^A_{35} L^A_{5F} \cup N^A_{3F} \\
L^A_{4F} &= E^A_{41} L^A_{1F} \cup E^A_{42} L^A_{2F} \cup E^A_{43} L^A_{3F} \cup E^A_{44} L^A_{4F} \cup E^A_{45} L^A_{5F} \cup N^A_{4F} \\
L^A_{5F} &= E^A_{51} L^A_{1F} \cup E^A_{52} L^A_{2F} \cup E^A_{53} L^A_{3F} \cup E^A_{54} L^A_{4F} \cup E^A_{55} L^A_{5F} \cup N^A_{5F.}
\end{aligned}$$

En remplaçant les E^A_{pq} par les expressions données dans la figure 4.6. et les N^A_{pF} par $\{\Lambda\}$ ou par $\varnothing$ selon la formule 4.2.8(3), on obtient

$$\begin{aligned}
L^A_{1F} &= \mathtt{a}L^A_{1F} \cup \mathtt{a}L^A_{2F} \cup \varnothing L^A_{3F} \cup \varnothing L^A_{4F} \cup \varnothing L^A_{5F} \cup \{\Lambda\} \\
L^A_{2F} &= \mathtt{b}L^A_{1F} \cup \varnothing L^A_{2F} \cup \varnothing L^A_{3F} \cup \varnothing L^A_{4F} \cup \varnothing L^A_{5F} \cup \varnothing \\
L^A_{3F} &= \varnothing L^A_{1F} \cup \varnothing L^A_{2F} \cup \varnothing L^A_{3F} \cup \mathtt{b}L^A_{4F} \cup \varnothing L^A_{5F} \cup \varnothing \\
L^A_{4F} &= \varnothing L^A_{1F} \cup \varnothing L^A_{2F} \cup \varnothing L^A_{3F} \cup \varnothing L^A_{4F} \cup \mathtt{a}L^A_{5F} \cup \varnothing \\
L^A_{5F} &= \varnothing L^A_{1F} \cup \varnothing L^A_{2F} \cup \varnothing L^A_{3F} \cup \varnothing L^A_{4F} \cup \varnothing L^A_{5F} \cup \{\Lambda\}.
\end{aligned}$$

Compte tenu de la formule $\varnothing B = B\varnothing = \varnothing$ d'algèbre de Kleene (3.2.3(4)), le système se simplifie, finalement sous la forme

$$\begin{cases}
L^A_{1F} = \mathtt{a}L^A_{1F} \cup \mathtt{a}L_{2F} \cup \{\Lambda\} \\
L^A_{2F} = \mathtt{b}L^A_{1F} \\
L^A_{3F} = \mathtt{b}L^A_{4F} \\
L^A_{4F} = \mathtt{a}L^A_{5F} \\
L^A_{5F} = \{\Lambda\}.
\end{cases} \tag{1}$$

Nous allons maintenant *résoudre* ce système d'équations, en utilisant certaines formules générales d'algèbre de Kleene (valables dans toute algèbre de Kleene) et une formule particulière, propre à l'algèbre de Kleene des langages sur un ensemble X. Il ne faut pas oublier que dans toute les équations considérées, les symboles `a` et `b` représentent les *langages élémentaires* $\{\langle\!\langle\, \mathtt{a} \,\rangle\!\rangle\}$, $\{\langle\!\langle\, \mathtt{b} \,\rangle\!\rangle\}$ et non les éléments `a`, `b` de X.

Des trois dernières équations (1), on tire

$$L^A_{4F} = \mathtt{a}L^A_{5F} = \mathtt{a}\{\Lambda\} = \mathtt{a}$$
$$L^A_{3F} = \mathtt{b}L^A_{4F} = \mathtt{ba}.$$

Les deux premières équations (1) entraînent

$$\begin{aligned} L^A_{1F} &= \mathtt{a}L^A_{1F} \cup \mathtt{a}L^A_{2F} \cup \{\Lambda\} \\ &= \mathtt{a}L^A_{1F} \cup \mathtt{ab}L^A_{1F} \cup \{\Lambda\} \\ &= (\mathtt{a} \cup \mathtt{ab})L^A_{1F} \cup \{\Lambda\}. \end{aligned}$$

L'équation

$$L^A_{1F} = (\mathtt{a} \cup \mathtt{ab})L^A_{1F} \cup \{\Lambda\} \tag{2}$$

peut se résoudre par rapport à L^A_{1F} grâce à un théorème particulier propre à l'algèbre de Kleene des langages sur un ensemble X. Ce théorème, appelé le *théorème d'Arden*, est démontré plus loin. Il dit qu'une égalité de langages de la forme

$$S = CS \cup B \tag{3}$$

équivaut à $S = C^*B$ si la séquence vide n'appartient pas au langage C. L'équation (2) est de la forme (3), si l'on prend

$$S = L^A_{1F}, \quad C = \mathtt{a} \cup \mathtt{ab}, \quad B = \{\Lambda\}.$$

Comme $\Lambda \notin (\mathtt{a} \cup \mathtt{ab})$, l'équation (2) équivaut à

$$L^A_{1F} = (\mathtt{a} \cup \mathtt{ab})^*\{\Lambda\},$$

à savoir

$$L^A_{1F} = (\mathtt{a} \cup \mathtt{ab})^*.$$

Pour achever la résolution du système d'équations (1), il reste à porter cette valeur de L^A_{1F} dans la deuxième équation du système, ce qui donne

$$L^A_{2F} = \mathtt{b}(\mathtt{a} \cup \mathtt{ab})^*.$$

Finalement, nous pouvons donner une expression régulière pour le langage L^A_{IF} de cet accepteur en appliquant la formule 4.2.7 (3) avec $I = \{1, 3\}$, ce qui donne :

$$L^A_{IF} = L^A_{1F} \cup L^A_{3F} = (\mathtt{a} \cup \mathtt{ab})^* \cup \mathtt{ba}.$$

4.2.11. Le théorème d'Arden

Soient S, A, B des langages sur X. Si $\Lambda \notin A$, alors

$$S = AS \cup B \;\Leftrightarrow\; S = A^*B.$$

Démonstration : voir [A].

4.2.12. Exercice

(a) Soient a, b, c, d des éléments distincts d'un alphabet X. Construire un accepteur fini (A, I, F) minimal (5 états) pour le langage

$$(\mathtt{a} \cup \mathtt{b} \cup \mathtt{c} \cup \mathtt{d})^*\mathtt{db}^*\mathtt{b}(\mathtt{a} \cup \mathtt{d}^*\mathtt{c}).$$

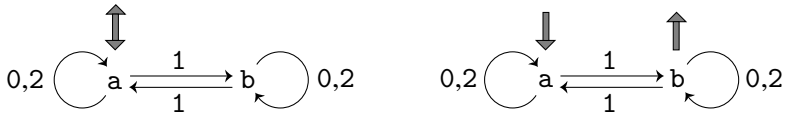
Vérifier la construction par calcul de L^A_{IF} (résolution d'équations), suivant l'exemple 4.2.10.

(b) Même exercice (4 états) pour le langage

$$((\mathtt{a} \cup \mathtt{b})^*\mathtt{bc} \cup \mathtt{c})(\mathtt{b}^* \cup \mathtt{b}^*(\mathtt{a} \cup \mathtt{b})).$$

4.2.13. Exercice

Obtenir par calcul (résolution d'équations) une expression régulière pour les langages sur l'alphabet $X = \{0, 1, 2\}$ qui sont définis par les deux accepteurs suivants, en supposant naturellement que $\mathtt{a} \neq \mathtt{b}$.



Comparer les expressions obtenues avec celles qui sont données dans la solution de l'exercice 3.4.9 pour les langages L_7 et L_8.

4.2.14. Théorème d'Arden généralisé

Un système d'équations de la forme

$$(1) \qquad \begin{cases} S_1 = A_{11}S_1 \cup A_{12}S_2 \cup \ldots \cup A_{1n}S_n \cup B_1 \\ S_2 = A_{21}S_1 \cup A_{22}S_2 \cup \ldots \cup A_{2n}S_n \cup B_2 \\ \ldots \\ S_n = A_{n1}S_1 \cup A_{n2}S_2 \cup \ldots \cup A_{nn}S_n \cup B_n \end{cases}$$

pour des langages $S_1, \ldots, S_n$ sur X, dans lesquelles les A_{ij} et les B_i sont des langages donnés sur X et la séquence Λ n'appartient à aucun des langages A_{ij}, admet une solution $(S_1, \ldots, S_n)$ et une seule. En outre, si tous les langages A_{ij} et B_i sont des langages réguliers sur X, alors les langages $S_1, \ldots, S_n$ de la solution sont des langages réguliers sur X.

Démonstration : voir [A].

4.3 Le théorème de Kleene

4.3.1. Introduction

Le but de cette section est de démontrer le théorème suivant (Kleene) :

Pour qu'un langage L sur un alphabet X soit un langage régulier sur X (3.4.6), il faut et il suffit qu'il existe un accepteur fini (A, I, F) sur X tel que $L = L^A_{IF}$.

4.3.2. Première partie du théorème de Kleene

Pour tout accepteur fini (A, I, F) sur un ensemble X, le langage L^A_{IF} défini par A est un langage régulier sur X.

Démonstration. Les langages L^A_{pF} associés aux états $p \in Q^A$ d'un tel accepteur (4.2.7) vérifient les équations

$$L^A_{pF} = \bigcup_{q \in Q^A} E^A_{pq} L^A_{qF} \cup N^A_{pF}.$$

Ces équations (une équation pour chaque état p) forment un système d'équations qui est de la forme considérée dans le théorème d'Arden généralisé (4.2.14), car la séquence Λ n'appartient à aucun des langages E^A_{pq} (4.2.9(1)). En outre, les langages E^A_{pq} et N^A_{pF} (4.2.8) sont des langages finis sur X, donc des langages réguliers sur X (3.4.7). Par suite (4.2.14),

les langages L^A_{pF} sont réguliers. Le langage L^A_{IF} défini par A est la réunion des L^A_{pF} tels que $p \in I$ (4.2.7(3)), donc il est la réunion d'un nombre fini de langages réguliers. Par suite, c'est un langage régulier sur X.

4.3.3. Théorème

Soient X un ensemble, (A, I, F) un accepteur fini sur X et $(S_p)_{p \in Q^A}$ une famille de langages sur X. Si, pour tout $p \in Q^A$, on a

$$S_p = \bigcup_{q \in Q^A} E^A_{pq} S_q \cup N^A_{pF}, \tag{1}$$

alors, pour tout $p \in Q^A$,

$$S_p = L^A_{pF}. \tag{2}$$

Démonstration. D'après le théorème d'Arden généralisé, le système des équations (1) (une équation pour chaque $p \in Q^A$) possède une solution et une seule (il existe une famille de langages $(S_p)_{p \in Q^A}$ et une seule vérifiant (1)). D'après le théorème 4.2.9, les langages $S_p = L^A_{pF}$ sont précisément solution de ces équations. Donc les équations (1) impliquent les égalités (2).

4.3.4. Théorème : réunion de deux accepteurs disjoints

Soit X un ensemble et soient (A_1, I_1, F_1) et (A_2, I_2, F_2) deux accepteurs finis sur X dont les ensembles d'états Q^{A_1} et Q^{A_2} sont disjoints : $Q^{A_1} \cap Q^{A_2} = \varnothing$. L'accepteur (A, I, F) sur X défini par

$$Q^A = Q^{A_1} \cup Q^{A_2}, \quad T^A = T^{A_1} \cup T^{A_2}, \quad I = I_1 \cup I_2, \quad F = F_1 \cup F_2, \tag{1}$$

est tel que

$$L^A_{IF} = L^{A_1}_{I_1 F_1} \cup L^{A_2}_{I_2 F_2}. \tag{2}$$

Démonstration. La propriété (2) est tout à fait évidente. Le diagramme de l'accepteur (A, I, F) est la simple juxtaposition des diagrammes de (A_1, I_1, F_1) et de (A_2, I_2, F_2), représentée symboliquement dans la figure 4.7. Il est évident qu'une séquence $x \in \mathrm{W}(X)$ relie un état initial à un état final dans l'accepteur (A, I, F) si et seulement si elle relie un état initial à un état final dans l'un ou l'autre des accepteurs (A_1, I_1, F_1), (A_2, I_2, F_2). D'où l'égalité (2).

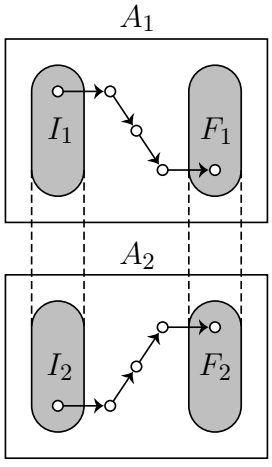


Figure 4.7
Réunion de deux accepteurs disjoints.
$I = I_1 \cup I_2$; $F = F_1 \cup F_2$.

Pour démontrer (2) formellement, nous admettons que, par définition de (A, I, F), c'est-à-dire en vertu de (1), on a

$$(3)\qquad E^A_{pq} = \begin{cases} E^{A_1}_{pq} & \text{si } (p,q) \in Q^{A_1} \times Q^{A_1}; \\ E^{A_2}_{pq} & \text{si } (p,q) \in Q^{A_2} \times Q^{A_2}; \\ \varnothing & \text{si } (p,q) \in (Q^{A_1} \times Q^{A_2}) \cup (Q^{A_2} \times Q^{A_1}). \end{cases}$$

$$N^A_{pF} = \begin{cases} N^{A_1}_{pF_1} & \text{si } p \in Q^{A_1}; \\ N^{A_2}_{pF_2} & \text{si } p \in Q^{A_2}. \end{cases}$$

Une démonstration de (3) est donnée dans [A]. Notre but est de démontrer, à partir de (3), la propriété suivante de l'accepteur A, dont l'égalité (2) se déduit aisément comme on le verra plus bas :

$$(4)\qquad \forall p \in Q^A : \quad L^A_{pF} = \begin{cases} L^{A_1}_{pF_1} & \text{si } p \in Q^{A_1}; \\ L^{A_2}_{pF_2} & \text{si } p \in Q^{A_2}. \end{cases}$$

Pour démontrer cela, nous désignons par S_p le membre de droite de l'égalité de (4), c'est-à-dire que nous posons, pour tout $p \in Q^A$:

$$(5)\qquad S_p = \begin{cases} L^{A_1}_{pF_1} & \text{si } p \in Q^{A_1}; \\ L^{A_2}_{pF_2} & \text{si } p \in Q^{A_2}. \end{cases}$$

Il s'agit donc de démontrer que l'on a $L^B_{pF} = S_p$ pour tout $p \in Q^A$. Pour cela, d'après le théorème 4.3.3, il suffit de montrer que pour tout $p \in Q^A$, on a :

$$(6)\qquad S_p = \bigcup_{q \in Q^A} E^A_{pq} S_q \cup N^A_{pF}.$$

Nous démontrons (6) par disjonction des cas $p \in Q^{A_1}$, $p \in Q^{A_2}$. Si $p \in Q^{A_1}$, on a

$$\begin{aligned} S_p = L^{A_1}_{pF_1} &= \bigcup_{q \in Q^{A_1}} E^{A_1}_{pq} L^{A_1}_{qF_1} \cup N^{A_1}_{pF_1} && \text{(5), 4.2.9(4)} \\ &= \bigcup_{q \in Q^{A_1}} E^A_{pq} S_q \cup N^A_{pF} && \text{(3)} \\ &= \bigcup_{q \in Q^{A_1}} E^A_{pq} S_q \cup \bigcup_{q \in Q^{A_2}} \overbrace{E^A_{pq}}^{\varnothing} S_q \cup N^A_{pF} && \text{(3)} \\ &= \bigcup_{q \in Q^A} E^A_{pq} S_q \cup N^A_{pF.} \end{aligned}$$

Si $p \in Q^{A_2}$, on a une chaîne d'égalités semblable. La propriété (4) est ainsi démontrée. On en déduit immédiatement (2) comme suit, d'après 4.2.7(3) :

$$\begin{aligned} L^A_{IF} = \bigcup_{p \in I} L^A_{pF} = \bigcup_{p \in (I_1 \cup I_2)} L^A_{pF} &= \bigcup_{p \in I_1} L^A_{pF} \cup \bigcup_{p \in I_2} L^A_{pF} \\ &= \bigcup_{p \in I_1} L^{A_1}_{pF_1} \cup \bigcup_{p \in I_2} L^{A_2}_{pF_2} = L^{A_1}_{I_1F_1} \cup L^{A_2}_{I_2F_2.} \end{aligned}$$

4.3.5. Théorème : composition en série de deux accepteurs disjoints

Soit X un ensemble et soient (A_1, I_1, F_1) et (A_2, I_2, F_2) deux accepteurs finis sur X dont les ensembles d'états Q^{A_1} et Q^{A_2} sont disjoints : $Q^{A_1} \cap Q^{A_2} = \varnothing$. L'accepteur (A, I, F)

sur X défini par

$$(1)\qquad \begin{cases} Q^A = Q^{A_1} \cup Q^{A_2}; \\ T^A = T^{A_1} \cup T^{A_2} \cup \{(p,a,q) \mid p \in F_1 \text{ et } \exists i \in I_2 : (i,a,q) \in T^{A_2}\}; \\ I = I_1; \\ F = \begin{cases} F_2 & \text{si } I_2 \cap F_2 = \varnothing; \\ F_1 \cup F_2 & \text{si } I_2 \cap F_2 \neq \varnothing, \end{cases} \end{cases}$$

est tel que

$$(2)\qquad L^A_{IF} = L^{A_1}_{I_1F_1} L^{A_2}_{I_2F_2}.$$

Démonstration. On dit que l'accepteur (A, I, F) défini par (1) est construit par composition en série des accepteurs A_1 et A_2. Cette construction est illustrée par la figure 4.8. Le diagramme de A s'obtient en juxtaposant les diagrammes de A_1 et A_2 et en ajoutant des transitions allant des états finaux de A_1 à certains états de A_2. De façon précise, pour chaque transition (i, a, q) de A_2 telle que $i \in I_2$, on ajoute une transition (p, a, q) pour chaque état final $p \in F_1$. Cela est illustré par les deux flèches en pointillé de la figure.

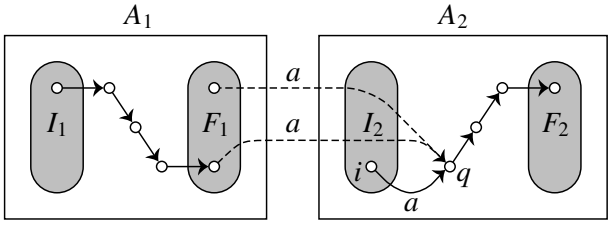


Figure 4.8
Composition en série de deux accepteurs disjoints.
$I = I_1$.
$F = F_2$ si $\Lambda \notin L_2$.
$F = F_1 \cup F_2$ si $\Lambda \in L_2$.

L'égalité (2) est assez évidente « par construction » de l'accepteur A. Sa démonstration formelle [A] procède comme celle de 4.3.4. Elle part des égalités suivantes :

$$(3)\qquad E^A_{pq} = \begin{cases} E^{A_1}_{pq} & \text{si } (p,q) \in Q^{A_1} \times Q^{A_1}; \\ \bigcup\limits_{i \in I_2} E^{A_2}_{iq} & \text{si } (p,q) \in F_1 \times Q^{A_2}; \\ \varnothing & \text{si } (p,q) \in (Q^{A_1} - F_1) \times Q^{A_2}; \\ E^{A_2}_{pq} & \text{si } (p,q) \in Q^{A_2} \times Q^{A_2}; \\ \varnothing & \text{si } (p,q) \in Q^{A_2} \times Q^{A_1} \end{cases}$$

$$N^A_{pF} = \begin{cases} N^{A_2}_{pF_2} & \text{si } I_2 \cap F_2 = \varnothing; \\ N^{A_1}_{pF_1} \cup N^{A_2}_{pF_2} & \text{si } I_2 \cap F_2 \neq \varnothing. \end{cases}$$

4.3.6. Théorème : fermeture d'un accepteur fini

Soient X un ensemble et (A, I, F) un accepteur fini sur X. L'accepteur (B, I', F') défini par

$$(1)\qquad \begin{cases} Q^B = Q^A \cup \{s\} \quad \text{où } s \text{ est un objet tel que } s \notin Q^A; \\ T^B = T^A \cup \{(p,a,q) \mid p \in F \text{ et } \exists i \in I : (i,a,q) \in T^A\}; \\ I' = I \cup \{s\}; \\ F' = F \cup \{s\} \end{cases}$$

est tel que

$$(2)\qquad L^B_{I'F'} = (L^A_{IF})^*.$$

Démonstration. On dit que l'accepteur (B, I', F') défini par (1) est construit par fermeture de l'accepteur (A, I, F). Cette construction est illustrée par la figure 4.9. Le diagramme de B s'obtient en ajoutant à celui de A des transitions allant des états finaux de A à certains états de A. De façon précise, pour chaque transition (i, a, q) de A telle que $i \in I$, on ajoute une transition (p, a, q) pour chaque état final $p \in F$. Cela est illustré par les deux flèches en pointillé de la figure. On ajoute encore un état isolé s, initial et final.

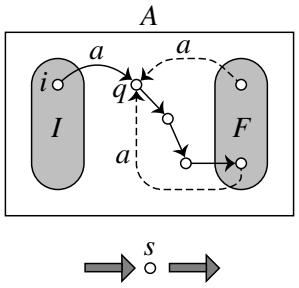


Figure 4.9
Fermeture d'un accepteur fini.

L'égalité (2) est assez évidente « par construction » de l'accepteur B. Sa démonstration formelle [A] procède comme celle de 4.3.4. Elle part des égalités suivantes :

$$(3)\qquad E^B_{pq} = \begin{cases} E^A_{pq} & \text{si } (p,q) \in (Q^A - F) \times Q^A; \\ E^A_{pq} \cup \bigcup\limits_{i \in I} E^A_{iq} & \text{si } (p,q) \in F \times Q^A; \\ \varnothing & \text{si } p = s \text{ ou } q = s; \end{cases}$$

$$N^B_{pF'} = \begin{cases} N^A_{pF} & \text{si } p \in Q; \\ \{\Lambda\} & \text{si } p = s. \end{cases}$$

4.3.7. Théorème de Kleene (deuxième partie)

Pour tout langage régulier L sur un alphabet X, il existe un accepteur fini (A, I, F) sur X tel que $L = L^A_{IF}$.

Démonstration. La démonstration de ce théorème a une grande importance pratique, car elle contient le principe d'un *algorithme de construction* d'un accepteur fini pour un langage régulier L donné.

Soit $\mathfrak{S}$ l'ensemble des langages élémentaires sur X. On rappelle que les langages réguliers sur X (3.4.6) sont les langages régulièrement engendrés par $\mathfrak{S}$ (3.4.4) et constituent l'ensemble de langages noté $\mathsf{Reg}(\mathfrak{S})$. Il s'agit donc de montrer que, quel que soit L,

$$L \in \mathsf{Reg}(\mathfrak{S}) \Rightarrow \begin{pmatrix} \text{il existe un accepteur fini } (A, I, F) \\ \text{sur } X \text{ tel que } L = L^A_{IF} \end{pmatrix}.$$

Désignons par $P(L)$ le membre de droite de l'implication ci-dessus, à savoir la proposition : *il existe un accepteur fini* (A, I, F) *sur* X *tel que* $L = L^A_{IF}$. Nous devons démontrer que

$$\forall L \in \mathsf{Reg}(\mathfrak{S}) : \quad P(L).$$

D'après le schéma de déduction de 3.4.10, il suffit pour cela de démontrer les cinq propositions suivantes, pour des langages L, L' quelconques sur X :

(a) $L \in \mathfrak{S} \Rightarrow P(L)$;

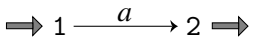
(b) $P(\varnothing)$;

(c) $(P(L) \text{ et } P(L')) \Rightarrow P(L \cup L')$;

(d) $(P(L) \text{ et } P(L')) \Rightarrow P(LL')$;

(e) $P(L) \Rightarrow P(L^*)$.

Proposition (a). La proposition (a) signifie que si L est un langage élémentaire sur X, alors il existe un accepteur fini (A, I, F) sur X tel que $L = L^A_{IF}$. Or cela est évident. Si $L = \{\langle\!\langle\, a \,\rangle\!\rangle\}$, avec $a \in X$, un accepteur pour L est par exemple l'accepteur (A, I, F) représenté par le diagramme



dans lequel $Q^A = \{1, 2\}$, $T^A = \{(1, a, 2)\}$, $I = \{1\}$, $F = \{2\}$.

Proposition (b). La proposition (b) signifie qu'il existe un accepteur fini (A, I, F) sur X tel que $L^A_{IF} = \varnothing$. Cela est aussi évident. On peut prendre pour A n'importe quel accepteur dans lequel $I = \varnothing$ ou $F = \varnothing$, car cela entraîne $L^A_{IF} = \varnothing$ (4.2.5). Le plus simple est l'accepteur (A, I, F) dont l'ensemble d'états Q^A est vide, ce qui entraîne $T^A = \varnothing$ et $I = F = \varnothing$. On peut prendre aussi un accepteur dans lequel les ensembles I et F ne sont pas vides mais aucune séquence $x \in \mathrm{W}(X)$ ne relie un état initial à un état final, par exemple l'accepteur sans transitions ($T^A = \varnothing$) suivant :

Proposition (c). Supposons que deux langages L, L' sur X vérifient $P(L)$ et $P(L')$, à savoir qu'il existe un accepteur fini (A_1, I_1, F_1) sur X tel que $L^{A_1}_{I_1F_1} = L$ et un accepteur fini (A_2, I_2, F_2) sur X tel que $L^{A_2}_{I_2F_2} = L'$. D'après le théorème 4.3.4, il existe un accepteur (A, I, F) sur X tel que $L^A_{IF} = L \cup L'$. Donc $P(L \cup L')$ est vraie. Le théorème en question fournit en outre une construction de l'accepteur (A, I, F) à partir des accepteurs (A_1, I_1, F_1) et (A_2, I_2, F_2).

Proposition (d). Supposons que deux langages L, L' sur X vérifient $P(L)$ et $P(L')$, à savoir qu'il existe un accepteur fini (A_1, I_1, F_1) sur X tel que $L^{A_1}_{I_1F_1} = L$ et un accepteur fini (A_2, I_2, F_2) sur X tel que $L^{A_2}_{I_2F_2} = L'$. D'après le théorème 4.3.5, il existe un accepteur (A, I, F) sur X tel que $L^A_{IF} = LL'$. Donc $P(LL')$ est vraie. Le théorème en question fournit en outre une construction de l'accepteur (A, I, F) à partir des accepteurs (A_1, I_1, F_1) et (A_2, I_2, F_2).

Proposition (e). Supposons qu'un langage L sur X vérifie $P(L)$, à savoir qu'il existe un accepteur fini (A, I, F) sur X tel que $L^A_{IF} = L$. En vertu du théorème 4.3.6, il existe un accepteur (B, I', F') sur X tel que $L^B_{I'F'} = L^*$. Donc $P(L^*)$ est vraie. Le théorème en question fournit une construction de (B, I', F') à partir de (A, I, F).

4.3.8. Exemple

Les constructions d'accepteurs utilisées dans la démonstration du théorème de Kleene (4.3.7) fournissent une méthode systématique (un algorithme) pour construire un accepteur pour un langage régulier L sur un alphabet X à partir d'une construction génératrice régulière de L (3.4.6). Nous illustrons ceci par l'exemple du langage

$$L = (01 \cup 2)^*01 \cup \{\Lambda\}$$

sur l'alphabet $X = \{0, 1, 2\}$. On a pour ce langage la construction génératrice régulière

$$
\begin{aligned}
&L_1 = 0;\\
&L_2 = 1;\\
&L_3 = 01 = L_1L_2;\\
&L_4 = 2;\\
&L_5 = 01 \cup 2 = L_3 \cup L_4;\\
&L_6 = (01 \cup 2)^* = L_5^*;\\
&L_7 = (01 \cup 2)^*01 = L_6L_3;\\
&L_8 = \varnothing;\\
&L_9 = \{\Lambda\} = L_8^*;\\
&L_{10} = (01 \cup 2)^*01 \cup \{\Lambda\} = L_7 \cup L_9.
\end{aligned}
$$

Des accepteurs pour chacun de ces langages sont construits de proche en proche dans la figure 4.10, en appliquant les constructions d'accepteurs fournies par les théorèmes 4.3.4, 4.3.5, 4.3.6. Les états de ces accepteurs sont représentés par des cercles anonymes. Il est entendu que dans chacun de ces accepteurs, tous ces cercles représentent des états distincts. Les arcs en pointillé correspondent aux transitions ajoutées dans une construction.

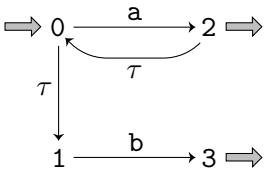
Il est clair que ces constructions fournissent pour L un accepteur qui est loin d'être le plus simple possible. Normalement, cet algorithme de construction se combine avec deux autres algorithmes : l'un qui transforme l'accepteur ainsi construit en un accepteur *déterministe* équivalent; l'autre qui transforme un accepteur déterministe en un accepteur déterministe *minimal* équivalent. Ces questions sont traitées dans les sections 4.4 et 4.5.

4.3.9. Exercice : accepteurs avec transitions invisibles

On ajoute parfois à un alphabet donné X un élément τ (lettre grecque tau), n'appartenant pas à X, et l'on considère qu'une séquence x sur l'alphabet augmenté $X \cup \{\tau\}$ représente la séquence sur X que l'on obtient en supprimant les occurrences de τ dans x s'il y en a. Par exemple, si $X = \{\mathtt{a}, \mathtt{b}\}$, la séquence $\langle\!\langle\, \mathtt{a}, \tau, \mathtt{a}, \tau, \tau, \mathtt{b} \,\rangle\!\rangle$ sur $X \cup \{\tau\}$ représente la séquence $\langle\!\langle\, \mathtt{a}, \mathtt{a}, \mathtt{b} \,\rangle\!\rangle$ sur X.

Un accepteur (A, I, F) sur l'alphabet $X \cup \{\tau\}$, par exemple l'accepteur

(1)



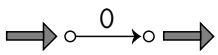
acceptant un langage L sur $X \cup \{\tau\}$ est considéré aussi comme accepteur du langage L' qui est obtenu en ignorant les occurrences de τ dans les séquences de L. Par exemple, le langage sur $X \cup \{\tau\}$ accepté par l'accepteur (1) est le langage

$$L = (\mathtt{a}\tau)^*(\mathtt{a} \cup \tau\mathtt{b}).$$

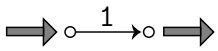
Le langage L' obtenu en ignorant les occurrences de τ est $L' = \mathtt{a}^*(\mathtt{a} \cup \mathtt{b})$. Un accepteur sur l'alphabet $X \cup \{\tau\}$ est appelé parfois un accepteur sur X avec des *transitions invisibles*, celles-ci étant les transitions de la forme (p, τ, q), et le langage L' est appelé le langage sur X accepté par l'accepteur à transitions invisibles.

(a) Montrer par des constructions appropriées (cf. 4.3.4 à 4.3.7) que pour tout langage régulier L sur un alphabet X il existe un accepteur à transitions invisibles pour L possédant un seul état initial i et un seul état final f, ces deux états satisfaisant en outre aux conditions

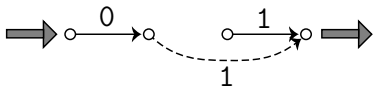
$L_1 = 0$



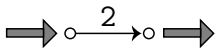
$L_2 = 1$



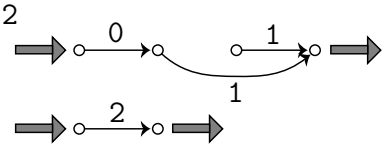
$L_3 = L_1 L_2 = 01$



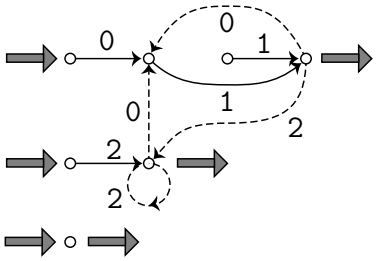
$L_4 = 2$



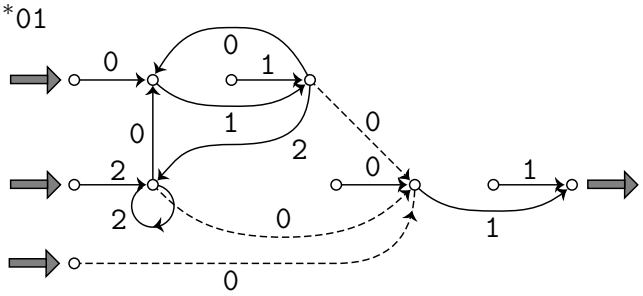
$L_5 = L_3 \cup L_4 = 01 \cup 2$



$L_6 = L_5^* = (01 \cup 2)^*$



$L_7 = L_6 L_3 = (01 \cup 2)^* 01$



$L_8 = \varnothing$ *néant*

$L_9 = L_8^* = \{\Lambda\}$

$L_{10} = L_7 \cup L_9 = (01 \cup 2)^* 01 \cup \{\Lambda\}$

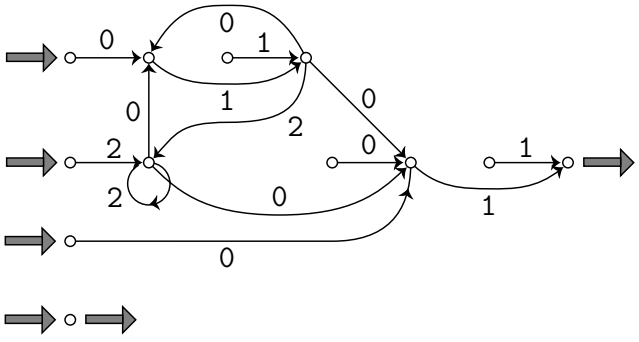
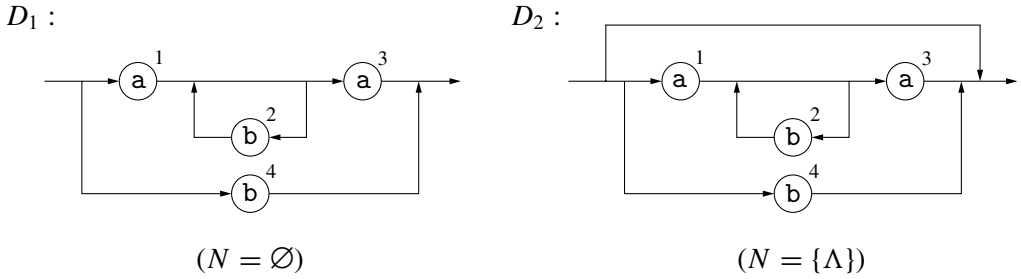


Figure 4.10



$(N = \varnothing)$ $(N = \{\Lambda\})$

Figure 4.11

suivantes : ils sont distincts ($i \neq f$) et il n'y a pas de transition de la forme (f, a, p), c'est-à-dire partant de l'état f, ni de transition de la forme (p, a, i), c'est-à-dire aboutissant à l'état i.

(b) Les constructions proposées dans (a) doivent fournir un algorithme de construction d'un accepteur avec transitions invisibles pour un langage donné par une expression régulière. On demande d'appliquer cet algorithme au langage $L = (\mathtt{ab} \cup \mathtt{a})^*$ sur l'alphabet $X = \{\mathtt{a}, \mathtt{b}\}$.

(c) Donner un procédé de construction d'un accepteur *sans* transitions invisibles sur un alphabet X, acceptant le même langage qu'un accepteur donné *avec* transitions invisibles sur X. Autrement dit, il s'agit de donner un procédé d'élimination des transitions invisibles.

4.3.10. Exercice : diagrammes syntaxiques réguliers

La figure 4.11 donne deux exemples, D_1 et D_2, de ce que nous appelons des *diagrammes syntaxiques réguliers* sur un alphabet $X = \{\mathtt{a}, \mathtt{b}, \dots\}$. Ces diagrammes représentent des langages sur cet alphabet, en l'occurrence les langages réguliers $\mathtt{ab}^*\mathtt{a} \cup \mathtt{b}$ et $\mathtt{ab}^*\mathtt{a} \cup \mathtt{b} \cup \{\Lambda\}$. Formellement, un diagramme syntaxique régulier D sur un alphabet X se compose de six données :

(1) Un ensemble fini P dont les éléments sont appelés les *places* du diagramme et sont représentés par des cercles. Par exemple, dans chacun des diagrammes D_1, D_2, on a $P = \{1, 2, 3, 4\}$.

(2) Une relation R dans l'ensemble P, c'est-à-dire un ensemble $R \subset P \times P$, appelé la *relation de succession* entre places. Cette relation est représentée graphiquement par des « routes à sens unique ». Un couple de places (p, q) appartient à R lorsqu'on peut aller de p à q en suivant ces routes, sans passer par une place intermédiaire. Dans D_1 et D_2, on a $R = \{(1, 2), (1, 3), (2, 2), (2, 3)\}$.

(3) Un ensemble P_I ($P_I \subset P$) de *places initiales*. Graphiquement, ce sont les places auxquelles on peut accéder directement depuis l'entrée du diagramme. Dans les diagrammes D_1 et D_2, on a $P_I = \{1, 4\}$.

(4) Un ensemble P_F ($P_F \subset P$) de *places finales*. Graphiquement, ce sont les places depuis lesquelles on peut accéder directement à la sortie. Dans les diagrammes D_1 et D_2, on a $P_F = \{3, 4\}$.

(5) Une application μ de P dans l'alphabet X associant à chaque place p un élément $\mu(p) \in X$ appelé la *marque* de p. Cet élément est noté dans le cercle qui représente p. Dans D_1 et D_2, on a $\mu(1) = \mathtt{a}$, $\mu(2) = \mathtt{b}$, $\mu(3) = \mathtt{a}$, $\mu(4) = \mathtt{b}$.

(6) Un ensemble N tel que $N = \varnothing$ ou $N = \{\Lambda\}$. On a $N = \varnothing$ pour un diagramme dans lequel on ne peut pas aller de l'entrée à la sortie sans passer par une place (cas de D_1), et $N = \{\Lambda\}$ dans le cas contraire (D_2).

Un diagramme régulier D composé de ces données est le 6-uple

$$D = (P, R, P_I, P_F, \mu, N).$$

Un *chemin* dans D est une séquence de places $\langle\!\langle\, p[1], \ldots, p[n] \,\rangle\!\rangle \neq \Lambda$ telle que $(p[i], p[i+1]) \in R$ pour tout $i \in [1 \,..\, n-1]$ (cette condition est trivialement satisfaite si $n = 1$). Par exemple, la séquence de places $\langle\!\langle\, 2, 2, 2, 3 \,\rangle\!\rangle$ est un chemin dans les diagrammes D_1 et D_2.

Le *langage représenté* par un diagramme régulier $D = (P, R, P_I, P_F, \mu, N)$ sur X est l'ensemble

$$L \cup N,$$

où L est l'ensemble des mots de la forme

$$\langle\!\langle\, \mu(p[1]), \mu(p[2]), \ldots, \mu(p[n]) \,\rangle\!\rangle$$

où $\langle\!\langle\, p[1], \ldots, p[n] \,\rangle\!\rangle$ est un chemin dans D tel que $p[1] \in P_I$ et $p[n] \in P_F$. Un tel chemin est appelé un *parcours* du diagramme.

Par exemple, la séquence de places $\langle\!\langle\, 1, 2, 2, 3 \,\rangle\!\rangle$ est un parcours du diagramme D_2, donc le mot $\langle\!\langle\, \mathtt{a}, \mathtt{b}, \mathtt{b}, \mathtt{a} \,\rangle\!\rangle = \langle\!\langle\, \mu(1), \mu(2), \mu(2), \mu(3) \,\rangle\!\rangle$ appartient au langage représenté par ce diagramme. Ce mot appartient aussi au langage représenté par D_1.

Questions

(a) Montrer au moyen de constructions appropriées (cf. démonstration du théorème de Kleene (4.3.7) sur les accepteurs finis) que tout langage régulier L sur un alphabet X peut être représenté par un diagramme syntaxique régulier sur X. Ces constructions seront définies de manière précise, basée sur la définition formelle d'un tel diagramme donnée ci-dessus. Elles doivent fournir un algorithme permettant de construire un diagramme D représentant L à partir d'une construction génératrice de L (cf. 4.3.8).

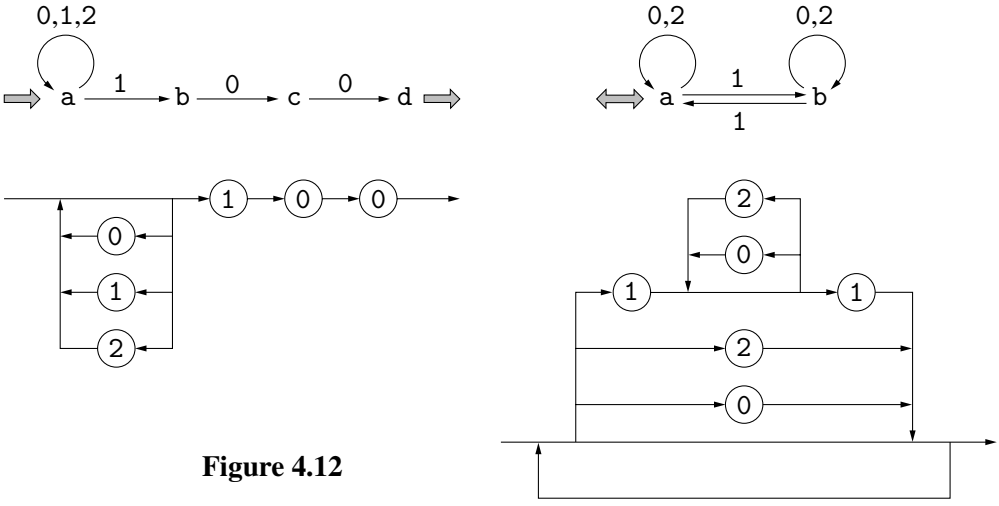
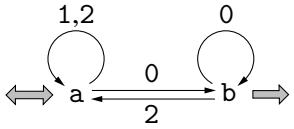


Figure 4.12

(b) Montrer d'une *autre manière* que pour tout langage régulier L sur un alphabet X il existe un diagramme syntaxique régulier qui représente L. La preuve consistera cette fois à donner un procédé (algorithme) de construction d'un diagramme syntaxique pour un langage régulier L à partir d'un *accepteur fini* donné pour L — il en existe un d'après le théorème de Kleene (4.3.7). Deux exemples d'accepteurs sont donnés dans la figure 4.12 avec les

diagrammes syntaxiques qui leur correspondent suivant cet algorithme. Observer que le nombre de places du diagramme syntaxique est égal à celui des transitions de l'accepteur à partir duquel il est construit. On appliquera le procédé de construction proposé à l'accepteur suivant, acceptant le langage L_9 de l'exercice 3.4.9 :



(c) Montrer que pour tout diagramme syntaxique régulier D sur un alphabet X, le langage représenté par D est un langage régulier sur X. Pour cela, on donnera un procédé de construction fournissant, pour tout diagramme D de cette espèce, un accepteur fini (A, I, F) sur X équivalent à D, en ce sens que le langage défini par (A, I, F) est celui qui est représenté par D. On conclura d'après le théorème de Kleene 4.3.2.

(d) Montrer d'une *autre manière* que pour tout diagramme syntaxique régulier D sur un alphabet X, le langage représenté par D est un langage régulier sur X. La preuve consistera cette fois à associer à chaque place p du diagramme le langage L_p qui est l'ensemble des mots de la forme $\langle\!\langle\, \mu(q[1]), \ldots, \mu(q[n]) \,\rangle\!\rangle$ où $\langle\!\langle\, q[1], \ldots, q[n] \,\rangle\!\rangle$ est un chemin dans D tel que $q[1] = p$ et $q[n] \in P_F$, et à montrer que ces langages satisfont à un système d'équations dont la forme permet de répondre à la question.

4.4 Automates finis déterministes

4.4.1. Définitions

On dit qu'un automate fini $A = (Q^A, T^A)$ sur X est *déterministe*, si, pour chaque $x \in X$, la relation relation de transition élémentaire

$$T_x^A = \{(p, q) \mid (p, x, q) \in T\} \qquad (4.1.5\,(1))$$

est une *fonction*, appelée naturellement la *fonction de transition élémentaire* T_x^A de A. Si, en outre, pour chaque $x \in X$, le domaine de cette fonction est égal à Q^A, on dit que A est un automate déterministe *complet*. Dans ce cas, pour chaque $x \in X$, la fonction T_x^A est une application de Q^A dans lui-même.

Un accepteur fini (A, I, F) sur X est dit *déterministe* (resp. *déterministe complet*) lorsque l'automate A est déterministe (resp. déterministe complet) et lorsque l'ensemble I possède un élément i et un seul : $I = \{i\}$.

4.4.2. Exemple

La figure 4.13 contient les diagrammes de trois accepteurs sur l'alphabet $X = \{0, 1\}$, pour un même langage L. Le premier de ces accepteurs n'est pas déterministe. On parle d'un accepteur *non déterministe*. L'automate A n'est pas déterministe, car sa relation de transition élémentaire $T_0^A = \{\mathtt{a} \mapsto \mathtt{a}, \mathtt{a} \mapsto \mathtt{b}\}$ n'est pas une fonction (on suppose naturellement que a, b, c, d sont des objets distincts.

L'accepteur (B, I, F) de la figure 4.13 est déterministe. Ses deux relations de transition élémentaires

$$T_0^B = \{\mathtt{a} \mapsto \mathtt{b}, \mathtt{b} \mapsto \mathtt{b}\}, \qquad T_1^B = \{\mathtt{a} \mapsto \mathtt{c}, \mathtt{b} \mapsto \mathtt{d}, \mathtt{c} \mapsto \mathtt{d}\}$$

sont des fonctions. Comme les domaines de ces fonctions, $\mathrm{Dom}\, T_0^B = \{\mathtt{a}, \mathtt{b}\}$, $\mathrm{Dom}\, T_1^B = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ ne sont pas égaux à l'ensemble d'états $Q^B = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$, l'accepteur B n'est pas complet.

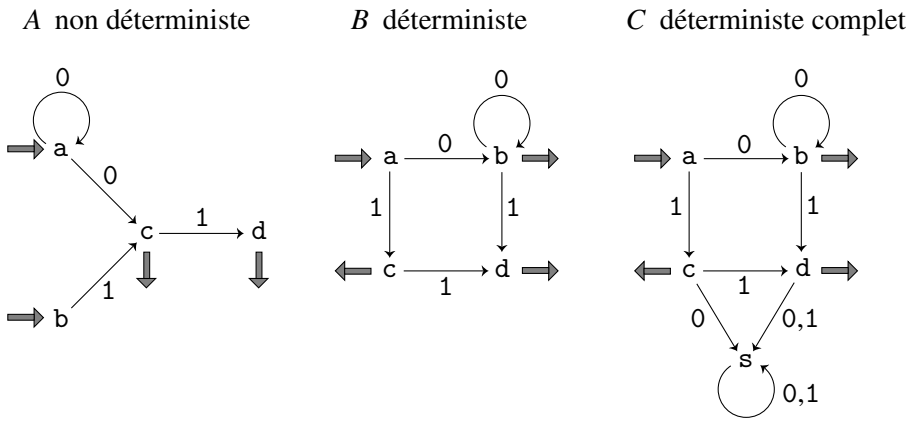


Figure 4.13

Trois accepteurs finis pour le langage $L = \mathtt{0}^+ \cup \mathtt{0}^+\mathtt{1} \cup \mathtt{1} \cup \mathtt{11}$ sur l'alphabet $X = \{\mathtt{0}, \mathtt{1}\}$.

L'accepteur C est construit en ajoutant des transitions à l'accepteur B de manière à le rendre déterministe complet, sans changer le langage accepté. On introduit pour cela un nouvel état `s`, et l'on ajoute des transitions qui ont toutes `s` comme état d'arrivée. Dans cet accepteur, on a

$$T_0^C = \{\mathtt{a} \mapsto \mathtt{b}, \mathtt{b} \mapsto \mathtt{b}, \mathtt{c} \mapsto \mathtt{s}, \mathtt{d} \mapsto \mathtt{s}, \mathtt{s} \mapsto \mathtt{s}\}$$
$$T_1^C = \{\mathtt{a} \mapsto \mathtt{c}, \mathtt{b} \mapsto \mathtt{d}, \mathtt{c} \mapsto \mathtt{d}, \mathtt{d} \mapsto \mathtt{s}, \mathtt{s} \mapsto \mathtt{s}\}.$$

Ce sont des fonctions de domaine $Q^C = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{s}\}$. En outre l'accepteur C a un état initial et un seul. Il est donc déterministe complet.

4.4.3. Théorème

Si A est un automate déterministe sur X, alors, pour toute séquence $x \in \mathrm{W}(X)$, la relation de transition Θ_x^A (4.1.5, Déf. 2) est une fonction, appelée naturellement la *fonction de transition* de A associée à x. Si A est un automate déterministe complet sur X, alors, pour toute séquence $x \in \mathrm{W}(X)$, la fonction de transition Θ_x^A est une application de Q^A dans lui-même ($\Theta_x^A : Q^A \to Q^A$).

Démonstration. Supposons que $A = (Q, T)$ soit un automate déterministe. Nous démontrons la proposition

$$\forall x \in \mathrm{W}(X) : \underbrace{\Theta_x^A \text{ est une fonction}}_{P(x)}$$

par induction structurelle dans $\mathrm{W}(X)$ (2.2.7). La proposition $P(\Lambda)$ est vraie parce que $\Theta_\Lambda^A = \mathrm{Id}_Q$ (4.1.5(5)) et Id_Q est une fonction. Pour tout $a \in X$, on a $\Theta_{\langle\!\langle a \rangle\!\rangle}^A = T_a^A$ (4.1.7(3)) et T_a^A est une fonction par hypothèse (A déterministe). Soient x, y deux séquences sur X et supposons que les relations de transition Θ_x^A et Θ_y^A soient des fonctions. Alors la relation $\Theta_{xy}^A = \Theta_x^A \Theta_y^A = \Theta_y^A \circ \Theta_x^A$ (4.1.7(4)) est une fonction, puisque la composée de deux fonctions est une fonction (1.3.22).

On raisonne de manière semblable, en supposant que A est un automate déterministe complet, pour démontrer

$$\forall x \in \mathrm{W}(X) : \Theta_x^A \text{ est une application de } Q \text{ dans } Q.$$

4.4.4. Théorème

Soit $(A, \{i\}, F)$ un accepteur fini déterministe complet sur X. Quels que soient $p, q \in Q^A$ et $x \in \mathrm{W}(X)$, on a

(1) $(p, q) \in \Theta_x^A \Leftrightarrow q = \Theta_x^A(p)$.

(2) $x \in L_{pF}^A \Leftrightarrow \Theta_x^A(p) \in F$.

Démonstration. Si $p \in Q^A$, la première de ces formules découle immédiatement de ce que Θ_x^A est une fonction de domaine Q^A (4.4.3, 1.3.10). La deuxième formule découle de la première :

$$\begin{aligned} x \in L_{pF}^A &\Leftrightarrow \exists q(q \in F \text{ et } (p, q) \in \Theta_x^A) \qquad 4.2.7\,(2) \\ &\Leftrightarrow \exists q(q \in F \text{ et } q = \Theta_x^A(p)) \\ &\Leftrightarrow \Theta_x^A(p) \in F. \end{aligned}$$

4.4.5. Transformés de l'ensemble d'états initiaux

Soient X un ensemble et (A, I, F) un accepteur fini sur X. Une séquence $x \in \mathrm{W}(X)$ appartient au langage L_{IF}^A, par définition (4.2.5), si et seulement si elle relie un état initial à un état final dans A. Une méthode pratique pour déterminer si x appartient à L_{IF}^A consiste à déterminer l'ensemble S de tous les états de A qui sont reliés par x à un état initial quelconque et à regarder s'il y a dans cet ensemble S un état final. De façon précise, cet ensemble S est l'ensemble des états q de A qui vérifient la condition

$$\exists p(p \in I \text{ et } (p, q) \in \Theta_x^A).$$

Autrement dit, S est le transformé de l'ensemble I par la relation de transition Θ_x^A, ensemble qui est noté $\Theta_x^A\langle I\rangle$ (1.3.23) et qui est caractérisé par :

$$q \in \Theta_x^A\langle I\rangle \Leftrightarrow \exists p(p \in I \text{ et } (p, q) \in \Theta_x^A).$$

Nous dirons que l'ensemble $\Theta_x^A\langle I\rangle$ est le *transformé de l'ensemble I par la séquence x dans A*. La séquence x est acceptée par (A, I, F) si et seulement si l'ensemble $\Theta_x^A\langle I\rangle$ contient un état final. C'est ce qu'énonce de façon générale le théorème suivant, qui est assez évident d'après les remarques qui précèdent.

Théorème. Soient X un ensemble et (A, I, F) un accepteur fini sur X. Pour toute séquence $x \in \mathrm{W}(X)$, on a

$$x \in L_{IF}^A \Leftrightarrow \Theta_x^A\langle I\rangle \cap F \neq \varnothing.$$

Calcul du transformé $\Theta_x^A\langle I\rangle$. Étant donné un accepteur (A, I, F) sur X et une séquence $x = \langle\!\langle\, x[1], \ldots, x[n] \,\rangle\!\rangle \in \mathrm{W}(X)$, il y a deux manières de déterminer l'ensemble $\Theta_x^A\langle I\rangle$. La première est d'appliquer la formule de récurrence

(1) $$\Theta_x^A\langle I\rangle = \begin{cases} I & \text{si } x = \Lambda; \\ T_{x[1]}^A\langle\Theta_{x\downarrow}^A\langle I\rangle\rangle & \text{si } x \neq \Lambda. \end{cases}$$

Cette formule découle de la formule de récurrence 4.1.5 (5) et des théorèmes 3 et 6 de 1.3.23, le premier étant appliqué à la relation composée $T_{x[1]}^A\Theta_{x\downarrow}^A = \Theta_{x\downarrow}^A \circ T_{x[1]}^A$. L'autre méthode est de déterminer pour chaque état $p \in I$ l'ensemble des états q tels que $(p, q) \in \Theta_x^A$ d'après 4.1.8 et de réunir les ensembles ainsi obtenus.

Exemple. Soient $X = \{0, 1\}$ et (A, I, F) l'accepteur de la figure 4.14, avec $Q^A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, où nous supposons que a, b, c sont distincts. On a

$$\Theta_{\langle\!\langle\, 0,1,0 \,\rangle\!\rangle}^A\langle I\rangle = \{\mathtt{a}, \mathtt{b}\}.$$

En posant $x = \langle\!\langle\, 0, 1, 0 \,\rangle\!\rangle$, cela peut se calculer de proche en proche, d'après (1):

$$\begin{aligned}\Theta_x^A\langle I\rangle &= T^A_{x[3]}\langle T^A_{x[2]}\langle T^A_{x[1]}\langle I\rangle\rangle\rangle\\ &= T^A_0\langle T^A_1\langle T^A_0\langle\{\mathtt{a},\mathtt{c}\}\rangle\rangle\rangle\\ &= T^A_0\langle T^A_1\langle\{\mathtt{a},\mathtt{b},\mathtt{c}\}\rangle\rangle\\ &= T^A_0\langle\{\mathtt{b},\mathtt{c}\}\rangle\\ &= \{\mathtt{a},\mathtt{b}\}.\end{aligned}$$

L'autre méthode (4.1.8) est de déterminer toutes les séquences d'états $s = \langle\!\langle\, s[1], \ldots, s[4] \,\rangle\!\rangle$ telles que l'on ait $s[1] \in I$ et telles que l'on ait dans A les transitions

$$s[1] \xrightarrow{\ 0\ } s[2] \xrightarrow{\ 1\ } s[3] \xrightarrow{\ 0\ } s[4].$$

L'ensemble $\Theta_x^A\langle I\rangle$ est l'ensemble des états $s[4]$ de ces séquences d'états. On obtient aussi $\Theta_x^A\langle I\rangle = \{\mathtt{a},\mathtt{b}\}$ et l'on voit que la séquence $x = \langle\!\langle\, 0, 1, 0 \,\rangle\!\rangle$ appartient au langage L^A_{IF} puisque que l'ensemble $\Theta_x^A\langle I\rangle$ contient un état final (état b).

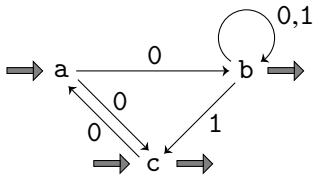


Figure 4.14

4.4.6. L'accepteur déterministe $\mathcal{D}(A)$ associé à un accepteur A

Nous allons présenter dans ce paragraphe une construction qui associe à tout accepteur fini (A, I, F) sur un alphabet X un accepteur fini déterministe complet sur X définissant le même langage que l'accepteur A. Nous désignerons cet accepteur déterministe complet, associé à (A, I, F), par $\mathcal{D}(A, I, F)$ ou simplement par $\mathcal{D}(A)$, les ensembles I et F étant sous-entendus. Nous commençons par un exemple.

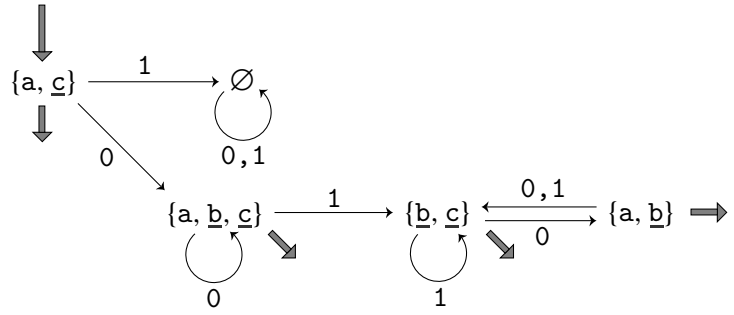
Exemple. L'accepteur déterministe $\mathcal{D}(A)$ associé à l'accepteur (A, I, F) de la figure 4.14, sur l'alphabet $X = \{0, 1\}$, est représenté dans la figure 4.15. On remarque d'abord que chaque état de $\mathcal{D}(A)$ est un sous-ensemble de l'ensemble d'états de A. En fait, les états de $\mathcal{D}(A)$ sont tous les sous-ensembles de Q^A de la forme $\Theta_x^A\langle I\rangle$ tels que $x \in \mathrm{W}(X)$. Bien qu'il existe une infinité de séquences x sur X, il n'existe qu'un nombre fini d'ensembles $\Theta_x^A\langle I\rangle$ distincts, puisque ces transformés sont des sous-ensembles de l'ensemble fini Q^A et un ensemble fini ne possède qu'un nombre fini de sous-ensembles distincts. L'ensemble I est lui-même un état de $\mathcal{D}(A)$, puisqu'il est égal à $\Theta_\Lambda^A\langle I\rangle$ (4.4.5(1)). Cet ensemble I est en outre l'unique état initial de l'accepteur $\mathcal{D}(A)$, cela fait partie de sa définition.

Les transitions de l'accepteur $\mathcal{D}(A)$ sont tous les triplets de la forme $(S, a, T_a^A\langle S\rangle)$ où S est un état de $\mathcal{D}(A)$ et $a \in X$. Par exemple, on a dans $\mathcal{D}(A)$ la transition

$$(\{\mathtt{a},\mathtt{c}\}, 0, \{\mathtt{a},\mathtt{b},\mathtt{c}\})$$

parce que $T_0^A\langle\{\mathtt{a},\mathtt{c}\}\rangle = \{\mathtt{a},\mathtt{b},\mathtt{c}\}$. De cette définition des transitions de $\mathcal{D}(A)$, il découle la propriété essentielle suivante que nous démontrerons plus bas: *une séquence x sur X relie l'état I à un état S dans l'accepteur $\mathcal{D}(A)$ si et seulement si $S = \Theta_x^A\langle I\rangle$.*

Les états finaux de l'accepteur $\mathcal{D}(A)$ sont par définition les états S de $\mathcal{D}(A)$ tels que $S \cap F \neq \varnothing$. Dans cet exemple, ce sont les ensembles S qui contiennent l'un ou l'autre des états b, c (états finaux de A), soulignés dans la figure 4.15. En vertu de la propriété essentielle énoncée ci-dessus et du théorème de 4.4.5, une séquence x relie l'état initial I



	0	1
{a}	{b, c}	∅
{b}	{b}	{b, c}
{c}	{a}	∅
⇒ {a, c}	{a, b, c}	∅
{a, b, c}	{a, b, c}	{b, c}
∅	∅	∅
{b, c}	{a, b}	{b, c}
{a, b}	{b, c}	{b, c}

Figure 4.15
Construction de l'accepteur déterministe complet $\mathcal{D}(A)$ associé à l'accepteur A de la figure 4.14

de $\mathcal{D}(A)$ à un état S final de $\mathcal{D}(A)$ si et seulement si elle appartient au langage L^A_{IF}. Donc le langage défini par $\mathcal{D}(A)$ est le même que le langage défini par A.

La construction de l'accepteur $\mathcal{D}(A)$ s'effectue commodément au moyen d'un tableau (figure 4.15). Dans la première partie de ce tableau on note les transformés par T^A_0 (colonne 0) et par T^A_1 (colonne 1) des sous-ensembles à un seul élément de Q^A ({a}, {b}, {c}). On a par exemple (d'après la figure 4.14) $T^A_0\langle\{\mathtt{a}\}\rangle = \{\mathtt{b}, \mathtt{c}\}$ et $T^A_1\langle\{\mathtt{a}\}\rangle = \varnothing$. Dans la deuxième partie du tableau, toutes les transitions de $\mathcal{D}(A)$ sont déterminées de proche en proche. On part de l'ensemble $I = \{\mathtt{a}, \mathtt{c}\}$, qui est l'état initial de $\mathcal{D}(A)$, et l'on détermine d'abord les transformés $T^A_0\langle I\rangle$ et $T^A_1\langle I\rangle$. Ce sont les ensembles $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ et $\varnothing$, qui sont donc deux états de $\mathcal{D}(A)$ distincts de I. La première partie du tableau sert à calculer commodément le transformé d'un ensemble quelconque $S \subset Q$ au moyen de la formule 1.3.23(6). Par exemple

$$T^A_0\langle\{\mathtt{a}, \mathtt{c}\}\rangle = T^A_0\langle\{\mathtt{a}\}\rangle \cup T^A_0\langle\{\mathtt{c}\}\rangle = \{\mathtt{b}, \mathtt{c}\} \cup \{\mathtt{a}\} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\};$$
$$T^A_1\langle\{\mathtt{a}, \mathtt{c}\}\rangle = T^A_1\langle\{\mathtt{a}\}\rangle \cup T^A_1\langle\{\mathtt{c}\}\rangle = \varnothing \cup \varnothing = \varnothing.$$

On détermine ensuite les transformés par T^A_0 et T^A_1 de ces deux nouveaux ensembles $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ et $\varnothing$. Seul le transformé $T^A_1\langle\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}\rangle = \{\mathtt{b}, \mathtt{c}\}$ est un état de $\mathcal{D}(A)$ distinct des précédents. On détermine donc les transformés par T^A_0 et T^A_1 de ce nouvel ensemble et ainsi de suite. La construction s'arrête lorsqu'on n'obtient plus de nouvel ensemble.

Définition. L'accepteur déterministe $\mathcal{D}(A)$ associé à un accepteur fini (A, I, F) sur X est défini comme suit :

(1) *États de $\mathcal{D}(A)$*

Les états de $\mathcal{D}(A)$ sont tous les transformés de l'ensemble initial I par une séquence $x \in \mathrm{W}(X)$. De façon précise, en désignant par $Q^{\mathcal{D}(A)}$ l'ensemble des états de $\mathcal{D}(A)$, on a

par définition :

$$S \in Q^{\mathcal{D}(A)} \;\Leftrightarrow\; \exists x \in \mathrm{W}(X) :\; S = \Theta_x^A\langle I\rangle.$$

Il est clair que tous les états de $\mathcal{D}(A)$ sont des sous-ensembles de Q^A puisque $I \subset Q^A$ et, pour toute séquence $x \in \mathrm{W}(X)$, Θ_x^A est une relation dans Q^A.

(2) *Transitions de* $\mathcal{D}(A)$

Les transitions de $\mathcal{D}(A)$ sont les triplets $(S, a, T_a^A\langle S\rangle)$ où S est un état de $\mathcal{D}(A)$ et $a \in X$.

Il faut s'assurer évidemment que la troisième composante d'un tel triplet est un état de $\mathcal{D}(A)$, à savoir qu'il existe une séquence $y \in \mathrm{W}(X)$ telle que $T_a^A\langle S\rangle = \Theta_y^A\langle I\rangle$. Montrons-le. Supposons que S soit un état de $\mathcal{D}(A)$ et que $a \in X$. Par définition d'un état de $\mathcal{D}(A)$, il existe une séquence $x \in \mathrm{W}(X)$ telle que $S = \Theta_x^A\langle I\rangle$. Pour une telle séquence x, on a

$$\begin{aligned}
T_a^A\langle S\rangle &= T_a^A\langle\Theta_x^A\langle I\rangle\rangle = (T_a^A \circ \Theta_x^A)\langle I\rangle && \text{1.3.23, théorème 3}\\
&= (\Theta_{\langle\!\langle a\rangle\!\rangle}^A \circ \Theta_x^A)\langle I\rangle && \text{4.1.7(3)}\\
&= (\Theta_x^A\Theta_{\langle\!\langle a\rangle\!\rangle}^A)\langle I\rangle &&\\
&= \Theta_{x\langle\!\langle a\rangle\!\rangle}^A\langle I\rangle && \text{4.1.7(4).}
\end{aligned}$$

Il existe donc une séquence y sur X, à savoir $y = x\langle\!\langle a\rangle\!\rangle$, telle que $T_a\langle S\rangle = \Theta_y^A\langle I\rangle$, donc $T_a\langle S\rangle$ est un état de $\mathcal{D}(A)$ (1).

(3) *État initial de* $\mathcal{D}(A)$

L'unique état initial de $\mathcal{D}(A)$ est le sous-ensemble I de Q. Il s'agit bien d'un état de A, puisque $I = \mathrm{Id}_Q\langle I\rangle = \Theta_\Lambda^A\langle I\rangle$.

(4) *États finaux de* $\mathcal{D}(A)$

Les états finaux de $\mathcal{D}(A)$ sont tous les états S de $\mathcal{D}(A)$ tels que $S \cap F \neq \varnothing$.

4.4.7. Théorèmes : propriétés de l'accepteur $\mathcal{D}(A)$

Soit (A, I, F) un accepteur fini sur X. L'accepteur $\mathcal{D}(A)$ défini par 4.4.6(1) à 4.4.6(4) a les propriétés suivantes :

(1) $\mathcal{D}(A)$ est déterministe et complet.

(2) Pour toute séquence $x \in \mathrm{W}(X)$, on a

$$(I, S) \in \Theta_x^{\mathcal{D}(A)} \Leftrightarrow S = \Theta_x^A\langle I\rangle.$$

(3) Le langage défini par $\mathcal{D}(A)$ est le langage L_{IF}^A défini par A.
On dit que les accepteurs A et $\mathcal{D}(A)$ sont équivalents.

(4) Chaque état de $\mathcal{D}(A)$ est *accessible depuis l'état initial* I, en ce sens que, pour tout état S de $\mathcal{D}(A)$, il existe une séquence $x \in \mathrm{W}(X)$ reliant I à S.

Démonstration : voir [A].

4.4.8. Exercice

Soit $X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{␣}\}$ un alphabet à cinq éléments distincts, dont un élément, désigné par $\mathtt{␣}$ est appelé « espace ». On appelle *mot propre* sur X tout mot sur X dans lequel il n'y a pas d'espace. Construire un accepteur minimal A (non déterministe, quatre états) sur X, acceptant l'ensemble des mots de la forme $x_1y_1 \cdots x_ny_n$, où $n \geq 1$ et :

- les x_i sont des mots propres se terminant par `ab` ou par `ba`;
- $y_i \in \sqcup^+$ pour tout $i \in [1 \,..\, n-1]$ (rappel: $\sqcup^+ = \sqcup\sqcup^*$), donc les x_i sont séparés par un ou plusieurs espaces;
- $y_n \in \sqcup^*$ (il peut y avoir des espaces à la fin).

Construire l'accepteur déterministe complet $\mathcal{D}(A)$.

4.4.9. Théorème

Pour tout langage régulier L sur X, il existe un accepteur fini déterministe complet $(A, \{i\}, F)$ dans lequel tout état est accessible depuis l'état initial et tel que $L = L^A_{iF}$.

Démonstration. Supposons que L soit un langage régulier sur X. D'après le théorème de Kleene (4.3.7), il existe un accepteur fini (A, I, F) sur X tel que $L = L^A_{IF}$. Le théorème découle immédiatement de l'équivalence d'un tel accepteur avec l'accepteur déterministe complet $\mathcal{D}(A)$ correspondant (4.4.7).

4.4.10. Accepteur complémentaire d'un accepteur déterministe complet

Théorème. Soient $(A, \{i\}, F)$ un accepteur déterministe complet sur X, et soit $\overline{F} = Q^A - F$ l'ensemble complémentaire de F par rapport à Q^A. Le langage défini par l'accepteur $(A, \{i\}, \overline{F})$ est le complément du langage accepté par A, par rapport à $\mathrm{W}(X)$:

$$L^A_{i\overline{F}} = \overline{L^A_{iF}} = \mathrm{W}(X) - L^A_{iF}.$$

Nous disons que l'accepteur $(A, \{i\}, \overline{F})$ est l'accepteur complémentaire de $(A, \{i\}, F)$. Les deux accepteurs ont le même automate sous-jacent A et le même état initial i.

Démonstration. Pour toute séquence $x \in \mathrm{W}(X)$, on a

$$\begin{aligned}
x \in \overline{L^A_{iF}} \Leftrightarrow x \notin L^A_{iF} &\Leftrightarrow \Theta^A_x(i) \notin F && 4.4.4(2)\\
&\Leftrightarrow \Theta^A_x(i) \in Q \text{ et } \Theta^A_x(i) \notin F && \Theta^A_x(i) \in Q \text{ est vrai}\\
&\Leftrightarrow \Theta^A_x(i) \in \overline{F} && \overline{F} = Q - F\\
&\Leftrightarrow x \in L^A_{i\overline{F}} && 4.4.4(2).
\end{aligned}$$

Corollaire. Pour tout langage régulier L sur X, le langage complémentaire $\overline{L} = \mathrm{W}(X) - L$ est un langage régulier sur X.

Ce corollaire est une conséquence immédiate du théorème, d'après 4.4.9.

Exemple. Deux accepteurs déterministes complets sur l'alphabet $X = \{0, 1\}$, complémentaires l'un de l'autre, sont représentés dans la figure 4.16. L'état initial des deux est $i = a$. Pour toute séquence $x \in \mathrm{W}(X)$, on a $\Theta^A_x(i) \in F$ ou $\Theta^A_x(i) \in \overline{F}$, dans le sens exclusif du mot « ou ». Donc les langages de ces deux accepteurs sont complémentaires en vertu de 4.4.4(2).

NB. La notion d'accepteur complémentaire d'un accepteur (A, I, F) n'est définie que dans le cas où cet accepteur est déterministe et complet.

4.4.11. Exercice

Nous nous référons à l'exercice 3.4.9. Le langage complémentaire du langage L_9 sur l'alphabet $X = \{0, 1, 2\}$ est l'ensemble des mots sur cet alphabet qui *comportent* une syllabe `01`, à savoir le langage

$$L = (0 \cup 1 \cup 2)^* 01 (0 \cup 1 \cup 2)^*.$$

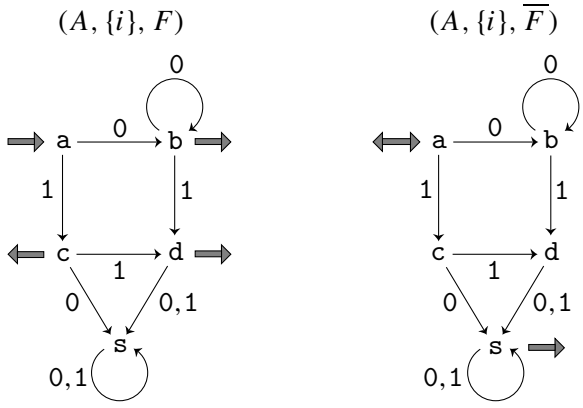


Figure 4.16
Accepteurs complémentaires sur $X = \{0, 1\}$

Construire un accepteur déterministe complet pour L_9 en procédant comme suit : construire un accepteur minimal A (non déterministe) pour L; construire l'accepteur déterministe complet associé $\mathcal{D}(A)$; prendre le complémentaire de celui-ci (4.4.10).

4.4.12. Théorème : intersection de deux langages réguliers

L'intersection $L \cap L'$ de deux langages réguliers L, L' sur un alphabet X est un langage régulier sur X.

Démonstration. En désignant par $\overline{L}$ le complément d'un langage L sur X par rapport à $\mathrm{W}(X)$, on a pour deux langages L, L' sur X :

$$L \cap L' = \overline{\overline{L} \cup \overline{L'}}.$$

Le théorème découle donc du corollaire 4.4.10 et du fait que la réunion de deux langages réguliers sur X est un langage régulier sur X (3.4.7).

4.4.13. Exercice

Soit M l'ensemble des mots sur l'alphabet $X = \{0, 1, 2\}$ qui comportent un nombre pair de 0 *et* un nombre pair de 1.

On a $M = L \cap L'$, où L est l'ensemble des mots sur X qui comportent un nombre pair de 0 et L' l'ensemble des mots sur X qui comportent un nombre pair de 1. On demande de construire un accepteur fini pour M en partant de deux accepteurs (avec des ensembles d'états disjoints) pour les langages $\overline{L}$ et $\overline{L'}$ (voir solution de l'exercice 3.4.9) et en effectuant les constructions d'accepteurs correspondant à la formule $M = \overline{\overline{L} \cup \overline{L'}}$.

4.5 L'accepteur canonique d'un langage régulier

4.5.1. Dérivés d'un langage

Soit L un langage sur X. Pour tout $x \in \mathrm{W}(X)$, on appelle *dérivé* de L (ou *reste* de L) par rapport à x et l'on note $D_x L$ l'ensemble des mots y sur X tels que le mot xy appartient à L :

$$D_x L = \{y \in \mathrm{W}(X) \mid xy \in L\}.$$

Par définition on a donc, quels que soient $x, y \in \mathrm{W}(X)$:

(1) $$y \in D_x L \Leftrightarrow xy \in L.$$

Exemple. Soient $X = \{0, 1, 2\}$ et L_{pair} (resp. L_{impair}) l'ensemble des mots $x \in \mathrm{W}(X)$ qui contiennent un nombre pair (resp. impair) de 1. Pour tout $x \in \mathrm{W}(X)$, on a

$$D_x L_{pair} = \begin{cases} L_{pair} & \text{si } x \in L_{pair}; \\ L_{impair} & \text{si } x \in L_{impair}. \end{cases}$$

En effet, si $x \in L_{pair}$, alors, pour toute séquence $y \in \mathrm{W}(X)$, le nombre d'occurrences de 1 dans la séquence xy est pair si et seulement si $y \in L_{pair}$, donc :

$$\begin{aligned} y \in D_x L_{pair} &\Leftrightarrow xy \in L_{pair} \qquad (1) \\ &\Leftrightarrow y \in L_{pair}, \end{aligned}$$

Par contre, si $x \in L_{impair}$, le nombre d'occurrences de 1 dans la séquence xy est pair si et seulement si $y \in L_{impair}$, donc : $y \in D_x L_{pair} \Leftrightarrow xy \in L_{pair} \Leftrightarrow y \in L_{impair}$.

4.5.2. Théorèmes

Soient L, L' des langages et x, y des séquences sur X. On a :

(1) $D_\Lambda L = L$.

(2) $D_{xy} L = D_y(D_x L)$.

(3) $D_x(L \cup L') = D_x L \cup D_x L'$.

(4) $\Lambda \in D_x L \Leftrightarrow x \in L$.

(5) $D_x \varnothing = \varnothing$.

(6) $D_x(\mathrm{W}(X)) = \mathrm{W}(X)$.

Démonstration : voir [A].

4.5.3. Théorème

Soit $(A, \{i\}, F)$ un accepteur fini déterministe complet sur X, et soient p et q deux états de A. Pour tout $x \in \mathrm{W}(X)$, on a

$$q = \Theta_x^A(p) \Rightarrow L_{qF}^A = D_x L_{pF}^A.$$

Démonstration. Soit $x \in \mathrm{W}(X)$ et supposons que $q = \Theta_x^A(p)$. Pour toute séquence $y \in \mathrm{W}(X)$, on a

$$\begin{aligned} y \in L_{qF}^A &\Leftrightarrow \Theta_y^A(q) \in F && 4.4.4(2) \\ &\Leftrightarrow \Theta_y^A(\Theta_x^A(p)) \in F \\ &\Leftrightarrow (\Theta_y^A \circ \Theta_x^A)(p) \in F \\ &\Leftrightarrow (\Theta_x^A \Theta_y^A)(p) \in F \\ &\Leftrightarrow \Theta_{xy}^A(p) \in F && 4.1.7(4) \\ &\Leftrightarrow xy \in L_{pF}^A && 4.4.4(2) \\ &\Leftrightarrow y \in D_x L_{pF}^A. \end{aligned}$$

Donc $L_{qF}^A = D_x L_{pF}^A$.

4.5.4. Théorème

Soient $(A, \{i\}, F)$ un accepteur déterministe complet sur X dans lequel tous les états sont accessibles depuis l'état initial i et soit $L = L^A_{iF}$. La fonction

$$\left(\lambda p \in Q^A : L^A_{pF} \right) \tag{1}$$

est une application surjective de l'ensemble Q^A dans l'ensemble des dérivés du langage L, à savoir l'ensemble $E = \{D_x L \mid x \in \mathrm{W}(X)\}$.

Démonstration. Dire que la fonction (1) est une application de Q^A dans l'ensemble $E = \{D_x L \mid x \in \mathrm{W}(X)\}$, c'est dire que pour tout $p \in Q^A$, le langage L^A_{pF} appartient à cet ensemble E, autrement dit qu'il existe une séquence $x \in \mathrm{W}(X)$ telle que $L^A_{pF} = D_x L$. Montrons cela. Soit $p \in Q^A$. En vertu de l'hypothèse d'accessibilité faite sur A, il existe une séquence $x \in \mathrm{W}(X)$ telle que $p = \Theta^A_x(i)$. Pour une telle séquence x, on a $L^A_{pF} = D_x L^A_{iF} = D_x L$, d'après 4.5.3. Donc $L^A_{pF} \in E$.

Dire que la fonction (1) est une application *surjective* de Q^A sur E, c'est dire que pour toute séquence $x \in \mathrm{W}(X)$ il existe un $p \in Q^A$ tel que $D_x L = L^A_{pF}$. Montrons cela. Soit $x \in \mathrm{W}(X)$. Comme l'accepteur déterministe A est complet, on a $\mathrm{Dom}\Theta^A_x = Q^A$, donc $i \in \mathrm{Dom}\Theta^A_x$ et par suite $\Theta^A_x(i) \in Q^A$. Soit $p = \Theta^A_x(i)$. D'après 4.5.3, on a $D_x L = L^A_{pF}$.

4.5.5. Théorème

Tout langage régulier L sur un alphabet X possède un nombre fini de dérivés distincts.

Commentaire. Il faut bien distinguer la *famille* $(D_x L)_{x \in \mathrm{W}(X)}$ des dérivés de L, à savoir la fonction $(\lambda x \in \mathrm{W}(X) : D_x L)$ qui fait correspondre à chaque séquence $x \in \mathrm{W}(X)$ le langage $D_x L$ (1.3.18). Le domaine de cette fonction est l'ensemble $\mathrm{W}(X)$, infini si $X \neq \varnothing$. Le théorème affirme que si L est un langage *régulier* sur X, la *portée* de cette fonction, à savoir l'ensemble de langages $E = \{D_x L \mid x \in \mathrm{W}(X)\}$ est un ensemble fini.

Démonstration. Soit L un langage régulier sur X et soit $(A, \{i\}, F)$ un accepteur déterministe complet dans lequel tout état est accessible depuis l'état initial et tel que $L = L^A_{iF}$. Un tel accepteur existe, d'après le théorème de 4.4.9. D'après 4.5.4, il existe une application surjective de l'ensemble fini Q^A dans l'ensemble E des dérivés de L. Cela implique que cet ensemble E est aussi fini.

4.5.6. Langages non réguliers

Le théorème 4.5.5 fournit une méthode pour montrer qu'un langage donné L sur un alphabet X *n'est pas* régulier. Cette méthode consiste à montrer que L possède une infinité de dérivés distincts.

Pratiquement, pour montrer de cette manière qu'un langage L sur X n'est pas régulier, il s'agit de trouver une famille $(x_k)_{k \in \mathbb{N}}$ de mots sur X telle que tous les langages $D_{x_k} L$ soient distincts, c'est-à-dire telle que l'on ait, quels que soient $k, k' \in \mathbb{N}$:

$$k \neq k' \Rightarrow D_{x_k} L \neq D_{x_{k'}} L.$$

Pour prouver cette dernière inégalité, il suffit de trouver un mot qui appartient à l'un des deux langages $D_{x_k} L$, $D_{x_{k'}} L$ et n'appartient pas à l'autre.

Exemple 1. Le langage

$$\begin{aligned} L &= \{\mathtt{0}^n \mathtt{1}^n \mid n \in \mathbb{N}\} \\ &= \{\Lambda, \mathtt{01}, \mathtt{0011}, \mathtt{000111}, \ldots\} \end{aligned}$$

sur l'alphabet $X = \{\mathtt{0}, \mathtt{1}\}$ n'est pas un langage régulier sur X. En effet tous les dérivés $D_{\mathtt{0}^k} L$ $(k \in \mathbb{N})$ sont distincts:

$$\begin{aligned} D_\Lambda L &= L \\ D_{\mathtt{0}} L &= \{\mathtt{1}, \mathtt{011}, \mathtt{00111}, \ldots\} = \{\mathtt{0}^n \mathtt{1}^{n+1} \mid n \in \mathbb{N}\} \\ D_{\mathtt{00}} L &= \{\mathtt{11}, \mathtt{0111}, \mathtt{001111}, \ldots\} = \{\mathtt{0}^n \mathtt{1}^{n+2} \mid n \in \mathbb{N}\} \\ &\ldots \\ D_{\mathtt{0}^k} L &= \{\mathtt{0}^n \mathtt{1}^{n+k} \mid n \in \mathbb{N}\}. \end{aligned}$$

Le mot le plus court de $D_{\mathtt{0}^k} L$ est le mot $\mathtt{1}^k$, de longueur k. Donc, si $k \neq k'$, alors $D_{\mathtt{0}^k} L \neq D_{\mathtt{0}^{k'}} L$, car le mot le plus court de l'un de ces deux langages n'appartient pas à l'autre.

Exemple 2. Soit X un alphabet possédant au moins deux éléments distincts. Le langage

$$L = \{xx \mid x \in \mathrm{W}(X)\}$$

n'est pas un langage régulier sur X.

Démonstration. Par définition de L, on a

$$y \in L \Leftrightarrow \exists x \in \mathrm{W}(X) : y = xx.$$

Par suite, pour tout $x \in \mathrm{W}(X)$ on a $xx \in L$, donc $x \in \mathcal{D}_x L$ (4.5.1(1)). Soient a et b deux éléments distincts de X. On voit facilement que tous les dérivés de L de la forme $D_{a^n b} L$ ($n \in \mathbb{N}$) sont distincts, en ce sens que si $m \neq n$, alors $D_{a^m b} L \neq D_{a^n b} L$.

En effet, supposons que $m \neq n$. Le mot $x = a^m b$ appartient à $D_{a^m b} L$. Montrons qu'il n'appartient pas à $D_{a^n b} L$. On raisonne par l'absurde en supposant que $a^m b \in D_{a^n b} L$. Cette hypothèse entraîne que $a^n b a^m b \in L$, donc que la séquence $a^n b a^m b$ admet une décomposition $a^n b a^m b = xx$, en deux facteurs x identiques. C'est évidemment impossible. Par exemple, le mot $aabaaaab$ ne se décompose pas en deux facteurs identiques.

4.5.7. Exercice

Montrer que les langages suivants ne sont pas des langages réguliers.

(a) L'ensemble des mots sur l'alphabet $X = \{\mathtt{0}, \mathtt{1}\}$ qui comportent le même nombre de $\mathtt{0}$ que de $\mathtt{1}$. $L = \{\Lambda, \mathtt{0011}, \mathtt{0101}, \mathtt{0110}, \mathtt{1001}, \mathtt{1010}, \mathtt{1100}, \ldots\}$.

(b) L'ensemble des mots de la forme $\mathtt{0}^m \mathtt{1}^n$ avec $m \geq n$ ($m, n \in \mathbb{N}$) sur l'alphabet $X = \{\mathtt{0}, \mathtt{1}\}$.

(c) L'ensemble des mots de la forme $\mathtt{a}^{n^2}$ ($n \in \mathbb{N}$) sur un alphabet X tel que $\mathtt{a} \in X$. $L = \{\mathtt{a}^0, \mathtt{a}^1, \mathtt{a}^4, \mathtt{a}^9, \ldots\} = \{\Lambda, \mathtt{a}, \mathtt{aaaa}, \mathtt{aaaaaaaaa}, \ldots\}$.

4.5.8. Exercice

On considère l'alphabet $X = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots, \mathbf{7}\}$ dont les éléments sont les vecteurs-colonnes suivants :

$$\mathbf{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{2} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{3} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \quad \ldots \quad , \quad \mathbf{7} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Une séquence non vide sur cet alphabet, par exemple la séquence

$$x = \langle\!\langle\, \mathbf{2}, \mathbf{7}, \mathbf{4}, \mathbf{0} \,\rangle\!\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{1}$$

peut être considérée comme une matrice à trois lignes, et nous considérons chacune de ces trois lignes comme un nombre représenté dans le système de numération binaire avec les

bits de poids faibles à gauche. Les trois nombres représentés par une séquence

$$x = \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} \cdots \begin{bmatrix} a_n \\ b_n \\ c_n \end{bmatrix}$$

de longueur $n \neq 0$ sont donc

$$\begin{aligned} \alpha(x) &= 2^0 a_1 + 2^1 a_2 + \cdots + 2^{n-1} a_n \\ \beta(x) &= 2^0 b_1 + 2^1 b_2 + \cdots + 2^{n-1} b_n \\ \gamma(x) &= 2^0 c_1 + 2^1 c_2 + \cdots + 2^{n-1} c_n. \end{aligned}$$

Soit L l'ensemble des séquences x non vides ($x \neq \Lambda$) sur l'alphabet X pour lesquelles le nombre $\gamma(x)$ est le produit arithmétique des nombres $\alpha(x)$ et $\beta(x)$. Par exemple, la séquence (1) appartient au langage L, puisque les nombres $\alpha(x)$, $\beta(x)$, $\gamma(x)$, pour cette séquence x, sont respectivement 2, 3 et 6.

On demande de montrer que L n'est pas un langage régulier sur X. Pour cela, considérer les dérivés de L par rapport aux séquences x_k de la forme

$$x_k = \mathbf{0}^k\mathbf{3} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \cdots \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{k} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \qquad (k \geq 1).$$

4.5.9. Exercice

(a) En se basant sur l'exercice 3.1.10, démontrer les formules suivantes, pour deux langages A, B quelconques sur un alphabet X et une séquence quelconque $x \in \mathrm{W}(X)$:

(1) $$y \in D_x(AB) \Leftrightarrow \left\{ \begin{array}{l} y \in (D_x A)B \\ \textbf{ou} \\ \text{il existe une séquence } u \text{ sur } X \text{ telle que } y \in D_u B \\ \text{et telle que } x = au \text{ pour une séquence } a \in A. \end{array} \right\}$$

(2) Si $x \neq \Lambda$,

$$y \in D_x(A^*) \Leftrightarrow \left\{ \begin{array}{l} \text{il existe une séquence } a \text{ sur } X \text{ telle que } y \in (D_a A)A^* \\ \text{et telle que } x = \alpha a \text{ pour une séquence } \alpha \in A^*. \end{array} \right\}$$

(b) Montrer que l'on peut déduire directement de ces formules et de 4.5.2(3) que tout langage régulier A sur X possède un nombre fini de dérivés distincts. On raisonne pour cela par induction structurelle dans l'ensemble des langages réguliers (3.4.10, 3.4.11), en considérant la proposition $P(A)$ suivante :

A possède un nombre fini de dérivés distincts.

Il suffit de montrer que cette proposition est vraie pour le langage $A = \varnothing$ ainsi que pour tout langage élémentaire $A = \{\langle\!\langle\, a \,\rangle\!\rangle\}$ sur X, et, qu'en vertu des formules mentionnées, $P(A)$ et $P(B)$ impliquent $P(A \cup B)$; $P(A)$ et $P(B)$ impliquent $P(AB)$; $P(A)$ implique $P(A^*)$.

4.5.10. L'accepteur canonique d'un langage régulier. Exemple

Soit L l'ensemble des mots sur l'alphabet $X = \{0, 1, 2\}$ qui se terminent par 100 (Exercie 3.4.9). On a pour ce langage l'expression régulière

(1) $$L = (0 \cup 1 \cup 2)^* 100.$$

	D_0	D_1	D_2
L	L	$L \cup 00$	L
$L \cup 00$	$L \cup 0$	$L \cup 00$	L
$L \cup 0$	$L \cup \{\Lambda\}$	$L \cup 00$	L
$L \cup \{\Lambda\}$	L	$L \cup 00$	L

Figure 4.17

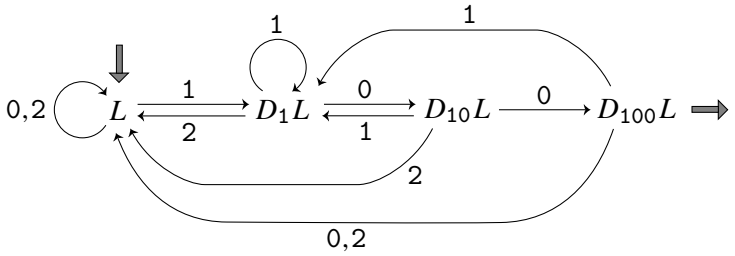


Figure 4.18

L'accepteur canonique $\Delta(L)$ du langage $L = (0 \cup 1 \cup 2)^*100$
sur l'alphabet $X = \{0, 1, 2\}$

Il est facile de voir que L possède exactement quatre dérivés distincts, à savoir

$$(2) \qquad \begin{cases} D_\Lambda L = L \\ D_1 L = L \cup 00 \\ D_{10} L = L \cup 0 \\ D_{100} L = L \cup \{\Lambda\}. \end{cases}$$

Le fait qu'il n'y a pas de dérivé de L autre que l'un des quatre langages (2) se voit en construisant un tableau des dérivés de L tel que celui de la figure 4.17. Dans la première ligne de ce tableau, on note les dérivés de L par rapport aux séquences élémentaires 0, 1, 2: $D_0 L = L$, $D_1 L = L \cup 00$, $D_2 L = L$. Ces trois dérivés se déterminent aisément d'après la définition de L et celle des dérivés (4.5.1). Par exemple, $x \in D_0 L \Leftrightarrow 0x \in L$ et il est clair que $0x$ appartient à L, c'est-à-dire se termine par 100 si et seulement si x se termine par 100, donc si et seulement si $x \in L$. D'où $D_0 L = L$. $D_1 L = L \cup 00$ parce que $x \in D_1 L \Leftrightarrow 1x \in L \Leftrightarrow 1x$ se termine par $100 \Leftrightarrow (x \in L$ ou $x = 00)$. La ligne suivante contient les dérivés du nouveau langage $L \cup 00$ obtenu dans la première ligne par rapport aux trois séquences élémentaires et fait apparaître un nouveau langage, à savoir $D_0(L \cup 00) = L \cup 0$. Cette égalité s'obtient en appliquant la formule 4.5.2(3):

$$D_0(L \cup 00) = D_0 L \cup D_0(00) = L \cup 0.$$

Ce langage est le dérivé $D_{10}L$, puisque $D_{10}L = D_0(D_1 L)$ (4.5.2(2)). De même, on a $D_1(L \cup 00) = D_1 L \cup D_1(00) = (L \cup 00) \cup \varnothing = L \cup 00$. La troisième ligne contient les dérivés du langage $L \cup 0$ par rapport aux séquences élémentaires et fait appparaître le nouveau langage $D_0(L \cup 0) = D_0 L \cup D_0 0 = L \cup \{\Lambda\}$. La quatrième ligne contient les dérivés de ce dernier langage par rappport aux séquences élémentaires et ne contient pas de nouveau langage.

Il est évident que les quatre langages (2) sont distincts, car le mot 00 n'appartient qu'au deuxième, le mot 0 n'appartient qu'au troisième et le mot Λ n'appartient qu'au quatrième. Le tableau des dérivés de L permet de construire l'accepteur représenté dans la figure 4.18, appelé *l'accepteur canonique du langage* L et noté $\Delta(L)$. Les états de cet accepteur sont

par définition les dérivés du langage L. Les transitions de $\Delta(L)$ sont tous les triplets

$$(M, a, D_{\langle\!\langle a \rangle\!\rangle} M)$$

tels que M est un dérivé de L et $a \in X$. L'état initial de $\Delta(L)$ est le langage $L = \Delta_\Lambda L$. Les états finaux sont les dérivés $D_x L$ tels que $\Lambda \in D_x L$. Il n'y a que le dérivé D_{100} qui satisfait à cette condition. L'accepteur $\Delta(L)$ est évidemment déterministe et complet. Nous verrons qu'il définit le langage L et qu'il possède le nombre minimum d'états possible pour un accepteur déterministe complet définissant L.

4.5.11. Définition générale de l'accepteur canonique d'un langage régulier

À tout langage régulier L sur un alphabet X on associe un accepteur fini déterministe complet sur X appelé *l'accepteur canonique de L sur X*. Nous notons cet accepteur $\Delta(L)$ et il est défini comme suit :

(1) *États de $\Delta(L)$*

Les états de $\Delta(L)$ sont tous les dérivés de L. Autrement dit, l'ensemble des états de L est l'ensemble $\{D_x L \mid x \in \mathrm{W}(X)\}$, qui est fini d'après 4.5.5.

(2) *Transitions de $\Delta(L)$*

Les transitions de $\Delta(L)$ sont tous les triplets de la forme

$$(M, a, D_{\langle\!\langle a \rangle\!\rangle} M)$$

où M est un état de $\Delta(L)$ et $a \in X$. Notons que dans un tel triplet, $D_{\langle\!\langle a \rangle\!\rangle} M$ est bien un état de $\Delta(L)$. En effet, dire que M est un état de $\Delta(L)$ c'est dire que M est un dérivé de L, à savoir qu'il existe une séquence $x \in \mathrm{W}(X)$ telle que $M = D_x L$. Cela entraîne que $D_{\langle\!\langle a \rangle\!\rangle} M$ est aussi un dérivé de L, puisque $D_{\langle\!\langle a \rangle\!\rangle}(D_x L) = D_{x\langle\!\langle a \rangle\!\rangle} L$ (4.5.2(2)).

(3) *État initial de $\Delta(L)$*

L'unique état initial de l'accepteur $\Delta(L)$ est l'état $L = D_\Lambda L$.

(4) *États finaux de $\Delta(L)$*

Les états finaux de $\Delta(L)$ sont tous les états M de $\Delta(L)$ tels que $\Lambda \in M$. D'après 4.5.2(4), ces états de $\Delta(L)$ sont les dérivés de L par rapport aux séquences x telles que $x \in L$.

4.5.12. Théorème : propriétés de l'accepteur $\Delta(L)$

Soit L un langage régulier L sur X. L'accepteur $\Delta(L)$ défini par 4.5.11(1) à 4.5.11(4) a les propriétés suivantes :

(1) $\Delta(L)$ est déterministe et complet.

(2) Pour tout $x \in \mathrm{W}(X)$, on a, quels que soient les états M, M' de $\Delta(L)$:

$$M' = \Theta_x^{\Delta(L)}(M) \;\Leftrightarrow\; M' = D_x M.$$

(3) Le langage défini par $\Delta(L)$ est L.

(4) Chaque état de $\Delta(L)$ est accessible depuis l'état initial L, en ce sens que, pour tout état M de $\Delta(L)$, il existe une séquence $x \in \mathrm{W}(X)$ reliant L à M $\quad(M = \Theta_x^{\Delta(L)}(L))$.

(5) Tout accepteur déterministe complet définissant L possède un nombre d'états qui est supérieur ou égal à celui de $\Delta(L)$.

Démonstration. Les propriétés (1) à (4) sont démontrées dans [A]. Nous démontrons ici la propriété (5). Supposons que $(A, \{i\}, F)$ soit un accepteur déterministe complet tel que

$L = L_{iF}^{A}$. Nous considérons d'abord le cas où tous les états de A sont accessibles depuis l'état initial i. D'après le théorème 4.5.4, il existe alors une application surjective de l'ensemble Q^A dans l'ensemble des états de $\Delta(L)$. Lorsqu'il existe une application surjective d'un ensemble fini E dans un ensemble fini E', le nombre d'éléments de E est supérieur ou égal à celui de E'. Donc le nombre d'états de A est supérieur ou égal au nombre d'états de $\Delta(L)$.

Dans le cas où A possède des états inaccessibles depuis l'état i, ces états et les transitions qui en partent ne jouent aucun rôle dans l'acceptation des séquences. On peut les supprimer sans changer le langage défini par l'accepteur. On obtient ainsi un accepteur déterministe complet pour le langage L n'ayant que des états accessibles depuis i. Le nombre d'états de cet accepteur est donc supérieur ou égal à celui de $\Delta(L)$. Le nombre d'états de A est a fortiori supérieur ou égal à celui de $\Delta(L)$.

4.5.13. Exercice

Soit L l'ensemble des mots sur l'alphabet $X = \{0, 1, 2\}$ qui comportent un nombre pair de 0 et un nombre pair de 1. Montrer que L possède un nombre fini de dérivés distincts et donner le diagramme de l'accepteur $\Delta(L)$.

4.5.14. Exercice

On considère l'alphabet $X = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots, \mathbf{7}\}$ de l'exercice 4.5.8. Une séquence non vide sur cet alphabet, par exemple la séquence

$$x = \langle\!\langle\, \mathbf{6}, \mathbf{3}, \mathbf{4}, \mathbf{0} \,\rangle\!\rangle = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{1}$$

peut être considérée comme une matrice à trois lignes, et nous considérons chacune de ces trois lignes comme étant la représentation d'un nombre dans le système de numération binaire avec les *poids faibles à gauche* (cf. Exercice 4.5.8).

Soit L l'ensemble des séquences x non vides sur l'alphabet X pour lesquelles le nombre $\gamma(x)$ est la somme arithmétique des nombres $\alpha(x)$ et $\beta(x)$. Par exemple, la séquence (1) appartient au langage L, puisque les nombres $\alpha(x)$, $\beta(x)$, $\gamma(x)$, pour cette séquence x, sont respectivement 2, 3 et 5.

On demande de montrer que L est un langage régulier en montrant qu'il possède 4 dérivés distincts. Pour cela considérer les quatre types de séquences suivants sur X, en observant que toute séquence sur X est de l'un de ces quatre types :

(a) la séquence vide Λ;
(b) les séquences x qui appartiennent à L;
(c) les séquences x non vides ($x \neq \Lambda$) qui n'appartiennent pas à L mais telles que $D_x(L) \neq \varnothing$;
(d) les séquences x qui n'appartiennent pas à L et telles que $D_x(L) = \varnothing$.

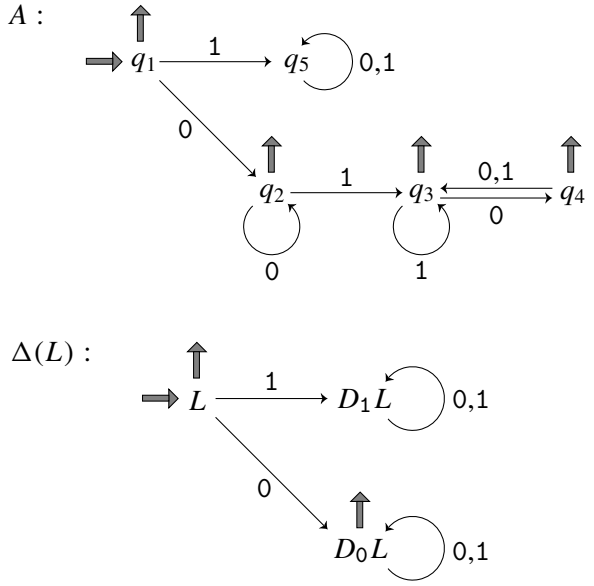
Commencer par donner des exemples de séquences des types (c) et (d).

Décrire les dérivés de L par rapport aux quatre types de séquences (a), (b), (c), (d) et construire l'accepteur canonique de L.

4.6 Minimisation d'un accepteur déterministe

4.6.1. Introduction

Le but de cette section est de présenter un algorithme pour la construction de l'accepteur canonique $\Delta(L)$ d'un langage L à partir d'un accepteur déterministe complet $(A, \{i\}, F)$



p	L^A_{pF}
q_1	L
q_2	D_0L
q_3	D_0L
q_4	D_0L
q_5	D_1L

Figure 4.19

Un accepteur déterministe complet A non minimal pour le langage $L = \{\Lambda\} \cup 0(0 \cup 1)^*$ sur $X = \{0, 1\}$ et l'accepteur canonique $\Delta(L)$.

quelconque acceptant L. En vertu de 4.5.12 (5), l'accepteur $\Delta(L)$ est appelé aussi l'accepteur déterministe complet *minimal* pour le langage L. Sa construction, à partir d'un accepteur déterministe complet $(A, \{i\}, F)$ quelconque (non minimal en général) pour L est appelée « opération de *minimisation* » ou *réduction* de A.

Nous supposerons toujours que l'accepteur de départ $(A, \{i\}, F)$ est non seulement déterministe et complet, mais encore que tous ses états sont accessibles depuis l'état initial i. Si cette condition n'est pas satisfaite, on peut toujours faire en sorte qu'elle le devienne en supprimant les états inaccessibles depuis i, ce qui ne change pas le langage accepté.

4.6.2. Exemple

Dans la figure 4.19, l'accepteur déterministe complet A sur $X = \{0, 1\}$ est celui de la figure 4.15 dont on a renommé les états $q_1, \ldots, q_5$. Il est facile de voir que le langage accepté par A est le langage $L = \{\Lambda\} \cup 0(0 \cup 1)^*$, autrement dit se compose de la séquence vide et de toutes les séquences qui commencent par 0. Ce langage possède trois dérivés distincts (L, D_0L, D_1L) et l'accepteur canonique $\Delta(L)$ est représenté dans la figure 4.19. L'accepteur A n'est donc pas minimal, en admettant évidemment que l'ensemble $Q = \{q_1, \ldots, q_5\}$ se compose de cinq objets *distincts*.

Dans cette figure, l'accepteur $\Delta(L)$ est déjà construit. Le problème que nous traitons dans cette section est celui d'une méthode de construction. Dans cet exemple, nous pourrions évidemment justifier la forme de $\Delta(L)$ en démontrant les égalités suivantes,[7] la première à partir de la donnée de A et les suivantes d'après la définition des dérivés d'un langage (4.5.1) et les formules de dérivation (4.5.2) :

$$L = L^A_{q_1 F} = \{\Lambda\} \cup 0(0 \cup 1)^*$$
$$D_0L = \{\Lambda\} \cup (0 \cup 1)^*$$

[7] Le lecteur est invité d'ailleurs à le faire.

$$
\begin{aligned}
D_1 L &= \varnothing \\
D_0(D_0 L) &= \{\Lambda\} \cup (0 \cup 1)^* = D_0 L \\
D_1(D_0 L) &= \{\Lambda\} \cup (0 \cup 1)^* = D_0 L \\
D_0(D_1 L) &= \varnothing = D_1 L \\
D_1(D_1 L) &= \varnothing = D_1 L.
\end{aligned}
$$

Cependant, la démonstration de ces égalités n'est pas un procédé algorithmique. Notre but est de construire $\Delta(L)$ à partir de A *sans avoir à déterminer* L *et ses dérivés.*

4.6.3. États équivalents

Deux états p et q d'un accepteur déterministe $(A, \{i\}, F)$ sont dits *équivalents* si l'on a

$$L^A_{pF} = L^A_{qF}. \tag{1}$$

L'ensemble de tous les états $p \in Q^A$ qui sont équivalents à un même état $q \in Q^A$ est appelé une *classe d'états équivalents* ou *classe d'équivalence* d'états de A. Les classes d'équivalence forment une partition de l'ensemble Q^A, à savoir que chaque état p appartient à une classe d'équivalence et une seule.

La première étape de l'algorithme de minimisation de A consiste à déterminer les classes d'équivalence d'états de A, mais sans devoir calculer les langages L^A_{pF} $(p \in Q^A)$ et les comparer.

Nous donnerons cette première partie de l'algorithme de minimisation plus loin. Nous voulons d'abord présenter la deuxième partie, plus simple, dans laquelle on construit l'accepteur canonique $\Delta(L)$ à partir de la donnée des classes d'états équivalents de A, déterminées dans la première étape.

4.6.4. Construction de $\Delta(L^A_{iF})$ à partir des classes d'états équivalents de A

Pour illustrer la deuxième étape de l'algorithme de minimisation, nous reprenons l'exemple de la figure 4.19 et nous admettons que les classes d'équivalence d'états de A, déterminées dans la première étape, sont les ensembles suivants :

$$\{q_1\}, \quad \{q_2, q_3, q_4\}, \quad \{q_5\}. \tag{1}$$

Ces trois ensembles forment bien une partition de l'ensemble $Q^A = \{q_1, \ldots, q_5\}$. Elle est mise en évidence dans le tableau de la figure 4.19 par les lignes de division en pointillé. Par définition d'une classe d'états équivalents, dire que les ensembles (1) sont les classes d'états équivalents de A signifie que l'on a parmi les langages L^A_{pF} $(p \in Q^A)$ les relations suivantes :

$$
\begin{aligned}
&L^A_{q_2F} = L^A_{q_3F} = L^A_{q_4F}; \\
&L^A_{q_1F} \neq L^A_{q_2F}; \quad L^A_{q_1F} \neq L^A_{q_5F}; \quad L^A_{q_2F} \neq L^A_{q_5F}.
\end{aligned} \tag{2}
$$

D'après le théorème 4.5.4 et d'après (2), on peut dire que l'ensemble des dérivés de L est l'ensemble de langages

$$\{L^A_{q_1F}, L^A_{q_2F}, L^A_{q_5F}\}.$$

Ces trois langages sont aussi les trois états de l'accepteur $\Delta(L)$. En outre, comme $q_2 = \Theta^A_{\langle\!\langle 0 \rangle\!\rangle}(q_1)$ et $q_5 = \Theta^A_{\langle\!\langle 1 \rangle\!\rangle}(q_1)$, on peut écrire, d'après 4.5.3 :

$$L^A_{q_1F} = L, \quad L^A_{q_2F} = D_{\langle\!\langle 0 \rangle\!\rangle} L, \quad L^A_{q_5F} = D_{\langle\!\langle 1 \rangle\!\rangle} L. \tag{3}$$

C'est ce qu'indique le tableau de la figure 4.19.

Le théorème 4.5.3 permet en outre de construire l'accepteur $\Delta(L)$, c'est-à-dire de déterminer toutes ses transitions à partir de celles de A. En effet, T^A étant l'ensemble des transitions de A et $T^{\Delta(L)}$ celui des transitions de $\Delta(L)$, on a l'implication

$$(4) \qquad (p, a, q) \in T^A \Rightarrow (L^A_{pF}, a, L^A_{qF}) \in T^{\Delta(L)}.$$

La preuve de cette implication est la suivante : si (p, a, q) est une transition de A, alors on a $q = \Theta^A_{\langle\!\langle a \rangle\!\rangle}(p)$ donc, d'après 4.5.3,

$$L^A_{qF} = D_{\langle\!\langle a \rangle\!\rangle} L^A_{pF}.$$

Comme L^A_{pF} et L^A_{qF} sont des états de $\Delta(L)$, le triplet (L^A_{pF}, a, L^A_{qF}) est une transition de $\Delta(L)$ (4.5.11 (2)).

Toutes les transitions de $\Delta(L)$ peuvent se déduire des transitions de A d'après la formule (4) et les égalités (2). Par exemple, puisque $(q_1, 0, q_2)$ est une transition de A, $(L^A_{q_1F}, 0, L^A_{q_2F})$ est une transition de $\Delta(L)$. L'ensemble des transitions de $\Delta(L)$ est déterminé de cette manière dans le tableau de la figure 4.20. On remplace partout $L^A_{q_3F}$ et $L^A_{q_4F}$ par $L^A_{q_2F}$ en vertu de l'égalité de ces langages (2), de manière à n'utiliser que trois expressions $(L^A_{q_1F}, L^A_{q_2F}, L^A_{q_5F})$ pour les trois dérivés de L. La partie inférieure du tableau, sous le double trait, est inutile : elle ne fournit que des transitions de $\Delta(L)$ figurant déjà dans la partie supérieure.

A	$\Delta(L)$
$(q_1, 0, q_2)$	$(L^A_{q_1F}, 0, L^A_{q_2F})$
$(q_1, 1, q_5)$	$(L^A_{q_1F}, 1, L^A_{q_5F})$
$(q_2, 0, q_2)$	$(L^A_{q_2F}, 0, L^A_{q_2F})$
$(q_2, 1, q_3)$	$(L^A_{q_2F}, 1, L^A_{q_3F}) = (L^A_{q_2F}, 1, L^A_{q_2F})$
$(q_5, 0, q_5)$	$(L^A_{q_5F}, 0, L^A_{q_5F})$
$(q_5, 1, q_5)$	$(L^A_{q_5F}, 1, L^A_{q_5F})$
$(q_3, 0, q_4)$	$(L^A_{q_3F}, 0, L^A_{q_4F}) = (L^A_{q_2F}, 0, L^A_{q_2F})$
$(q_3, 1, q_3)$	$(L^A_{q_3F}, 1, L^A_{q_3F}) = (L^A_{q_2F}, 1, L^A_{q_2F})$
$(q_4, 0, q_3)$	$(L^A_{q_4F}, 0, L^A_{q_3F}) = (L^A_{q_2F}, 0, L^A_{q_2F})$
$(q_4, 1, q_3)$	$(L^A_{q_4F}, 0, L^A_{q_3F}) = (L^A_{q_2F}, 1, L^A_{q_2F})$

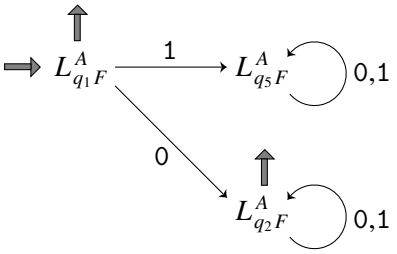


Figure 4.20

Construction de l'accepteur canonique correspondant à l'accepteur A de la figure 4.19 à partir de la donnée des classes d'états équivalents de A : $\{q_1\}, \{q_2, q_3, q_4\}, \{q_5\}$.

L'accepteur $\Delta(L)$ obtenu est représenté par le diagramme de la figure 4.20. L'état initial et les états terminaux sont déterminés d'après la définition 4.5.11 : l'état initial est $L = L^A_{q_1F}$; les états finaux sont les dérivés de L qui contiennent la séquence vide, donc les langages $L^A_{q_iF}$ tels que $q_i \in F$. D'après les égalités (3), l'accepteur $\Delta(L)$ construit dans la figure 4.20 est bien celui qui est donné dans la figure 4.19.

4.6.5. Détermination des classes d'états équivalents

Nous exposons maintenant l'algorithme de détermination des classes d'états équivalents d'un accepteur déterministe complet $(A, \{i\}, F)$ sur un alphabet X, en l'illustrant toujours avec l'exemple de la figure 4.19. Il s'agit de déterminer l'ensemble des couples d'états (p, q) de A tels que $L^A_{pF} = L^A_{qF}$.

La méthode consiste à partir de l'ensemble de tous les couples d'états de A, à savoir l'ensemble $Q^A \times Q^A$, et à « supprimer » dans cet ensemble les couples d'états (p, q) pour lesquels on sait que $L^A_{pF} \neq L^A_{qF}$. L'ensemble $Q^A \times Q^A$ peut être représenté par l'ensemble des cases d'un tableau carré tel que celui de la figure 4.21, dans lequel chaque couple d'états (p, q) est représenté par la case « d'adresse » (p, q), où p est l'adresse de la ligne et q l'adresse de la colonne de cette case. Nous ignorons pour le moment le contenu de certaines de ces cases. Initialement, on part d'un tableau vide. Lorsque, d'après certains critères, on sait que pour un couple d'états (p, q) on a $L^A_{pF} \neq L^A_{qF}$, on biffe la case (p, q).

Premier critère d'élimination. Le premier de ces critères est que si l'un des états p, q appartient à F et l'autre n'appartient pas à F, alors on a $L^A_{pF} \neq L^A_{qF}$, car la séquence Λ appartient à l'un de ces langages et n'appartient pas à l'autre. Suivant ce critère, on élimine les couples (q_1, q_5), (q_2, q_5), (q_3, q_5), (q_4, q_5) et les couples symétriques (q_5, q_1), (q_5, q_2), (q_5, q_3), (q_5, q_4). Ce sont les cases biffées de la figure 4.21. En pratique, vu la symétrie de la relation $L^A_{pF} \neq L^A_{qF}$ et le fait que cette relation n'a jamais lieu pour un couple « diagonal » (p, q) tel que $p = q$, on ne considère que la partie du tableau située au-dessus de la diagonale, entourée en trait fort, et l'on ne dessine pas le reste. Chaque case d'adresse (p, q) de cette zône représente aussi la case symétrique, d'adresse (q, p).

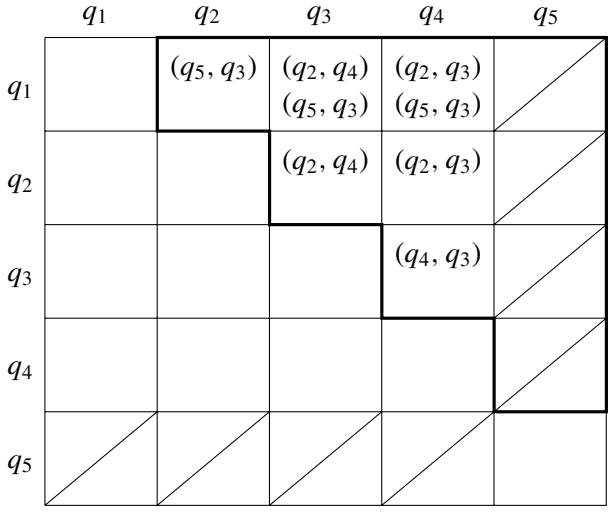


Figure 4.21

Détermination des couples d'états équivalents
de l'accepteur A de la figure 4.19 (début).

Deuxième critère d'élimination. Si, pour deux états p, q de A et pour un $a \in X$, les états $p' = T^A_a(p)$ et $q' = T^A_a(q)$ ne sont pas équivalents ($L^A_{p'F} \neq L^A_{q'F}$), alors les états p et q ne sont pas équivalents : $L^A_{pF} \neq L^A_{qF}$.

Démontrons cette assertion. Soient $p' = T^A_a(p)$ et $q' = T^A_a(q)$, autrement dit (4.1.7 (3))

$p' = \Theta^A_{\langle\!\langle a \rangle\!\rangle}(p)$ et $q' = \Theta^A_{\langle\!\langle a \rangle\!\rangle}(q)$. D'après 4.5.3, on a

$$L^A_{p'F} = D_{\langle\!\langle a \rangle\!\rangle} L^A_{pF} \text{ et } L^A_{q'F} = D_{\langle\!\langle a \rangle\!\rangle} L^A_{qF},$$

donc $L^A_{pF} = L^A_{qF} \Rightarrow L^A_{p'F} = L^A_{q'F}$. Par contraposition, si $L^A_{p'F} \neq L^A_{q'F}$ alors $L^A_{pF} \neq L^A_{qF}$.

Pour appliquer ce critère, on note dans la case de chaque couple d'états (p, q) pas encore éliminé les couples d'états (p', q') pour lesquels il existe un $a \in X$ tel que $p' = T^A_a(p)$ et $q' = T^A_a(q)$. Ce sont les inscriptions qui se trouvent dans les cases de la figure 4.21. On s'abstient de noter les couples (p', q') tels que $p' = q'$, pour lesquels on n'aura évidemment pas $L^A_{p'F} \neq L^A_{q'F}$. Le critère s'applique alors comme suit : si une case d'adresse (p, q) contient un couple (p', q') tel que la case d'adresse (p', q') est biffée, alors on biffe la case (p, q). On *répète* cette opération *tant qu'il existe un case non biffée contenant un couple* (p', q') *tel que la case d'adresse* (p', q') *est biffée*. Le tableau final que l'on obtient dans notre exemple est donné dans la figure 4.22. En général, il faut passer plusieurs fois le tableau en revue, car chaque élimination d'une case peut entraîner l'élimination d'autres cases.

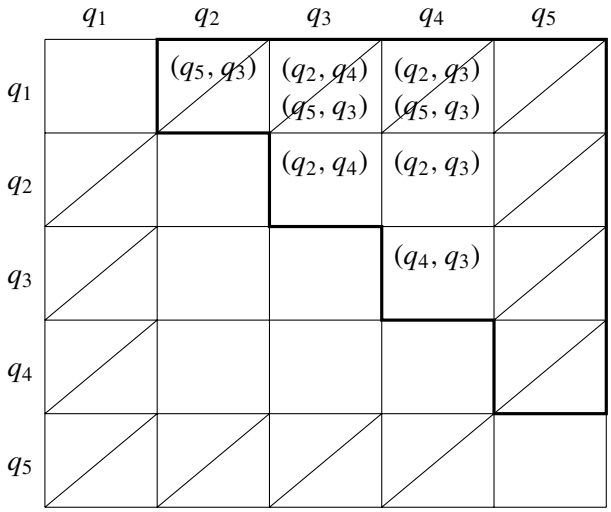


Figure 4.22

Détermination des couples d'états équivalents
de l'accepteur A de la figure 4.19 (fin)

Lorsque ce processus est terminé, on sait qu'en vertu des critères d'élimination appliqués, on a $L^A_{pF} \neq L^A_{qF}$ pour chaque couple d'états (p, q) tel que la case d'adresse (p, q) est biffée. Mais qu'en est-il des cases non biffées ? Est-on sûr que $L^A_{pF} = L^A_{qF}$ pour chaque case non biffée d'adresse (p, q) ? La réponse est certainement oui pour les cases diagonales. Elle est aussi affirmative pour les autres, mais cela demande une démonstration. De façon plus précise, on démontre la propriété suivante :

Propriété. L'ensemble des couples d'états (p, q) de A correspondant aux cases non biffées du tableau à la fin du processus (figure 4.22) est l'ensemble des couples d'états équivalents de A.

Dans cet exemple, ce sont : les couples diagonaux (q_i, q_i) ainsi que les couples

$$(q_2, q_3), \quad (q_2, q_4), \quad (q_3, q_4),$$

d'où les classes d'équivalence $\{q_1\}, \{q_2, q_3, q_4\}, \{q_5\}$ dont nous sommes partis au paragraphe précédent (4.6.4(1)) pour construire l'accepteur $\Delta(L)$ du langage $L = L^A_{q_1F}$ (figure 4.19).

La démonstration de cette propriété est donnée dans [A].

4.6.6. Exercice

Construire l'accepteur déterministe minimal correspondant à l'accepteur déterministe complet $(A, \{i\}, F)$ de la figure 4.23 sur l'alphabet $X = \{\mathtt{a}, \mathtt{b}\}$, à savoir l'accepteur canonique du langage $L = L^A_{iF}$.

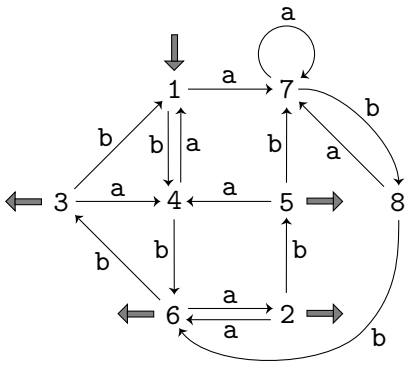


Figure 4.23

Chapitre 5 Machines

5.1 L'algèbre de Kleene des relations dans un ensemble

5.1.1. Introduction

Nous avons vu au §1.6.3 que l'ensemble $\mathcal{R}(E)$ des relations dans un ensemble E, muni de l'opération de produit de composition $RS = S \circ R$ et de la relation identique Id_E est un monoïde. Nous allons voir maintenant qu'il s'agit non seulement d'un monoïde, mais encore d'une algèbre de Kleene (3.2.2). Premièrement, les éléments de l'ensemble $\mathcal{K} = \mathcal{R}(E)$ sont des ensembles, puisqu'une relation dans E est par définition (1.3.3) un sous-ensemble de $E \times E$. Deuxièmement, la réunion d'une famille quelconque $(R_i)_{i \in I}$ de tels ensembles est encore une relation dans E. Formellement, si $\mathcal{K} = \mathcal{R}(E)$, alors

$$(\forall i \in I : R_i \in \mathcal{K}) \Rightarrow \Big(\bigcup_{i \in I} R_i \Big) \in \mathcal{K}.$$

Ceci est la deuxième propriété caractéristique d'une algèbre de Kleene (3.2.2). Il reste à vérifier, troisièmement, que le produit de composition est distributif par rapport à la réunion, en ce sens que, pour toute famille $(R_i)_{i \in I}$ de relations dans E et pour toute relation S dans E, on a

$$\Big(\bigcup_{i \in I} R_i \Big) S = \bigcup_{i \in I} (R_i S) \quad \text{et} \quad S\Big(\bigcup_{i \in I} R_i \Big) = \bigcup_{i \in I} (S R_i).$$

Ces formules sont démontrées dans [A].

Toutes les définitions et tous les théorèmes du chapitre 3 sur les algèbres de Kleene en général s'appliquent évidemment au cas particulier l'algèbre de Kleene $\mathcal{K} = \mathcal{R}(E)$. Notamment, toutes les formules des paragraphes 3.2.3 à 3.2.8 sont vraies si A, B, C désignent des relations dans un ensemble E, si tous les produits sont des produits de composition de relations et si le terme $I_{\mathcal{K}}$ désigne la relation Id_E. Au lieu de A, B, C, lorsqu'il s'agit de relations, nous utilisons de préférence les variables R, S, T.

Les puissances R^n d'une relation R dans E et sa fermeture de Kleene R^* sont définies conformément à 3.2.4 et 3.2.5 par

$$\begin{aligned} & R^0 = \mathrm{Id}_E. \\ \forall n \in \mathbb{N} : \quad & R^{n+1} = R(R^n). \\ & R^* = \bigcup_{n \in \mathbb{N}} R^n. \end{aligned}$$

Il sera aussi question, dans ce chapitre, des relations dans un ensemble E qui sont régulièrement engendrées par un ensemble $\mathfrak{S}$ de relations dans E au sens de 3.4.4, c'est-à-dire des relations que l'on peut construire à partir de relations $R, S, T, \ldots$ appartenant à $\mathfrak{S}$ et/ou de la relation $\varnothing$ en effectuant un nombre fini d'opérations de réunion, de produit ou de fermeture de Kleene, comme cela se fait dans une construction génératrice régulière de base $\mathfrak{S}$. C'est dire que nous allons appliquer au cas de l'algèbre de Kleene $\mathcal{K} = \mathcal{R}(E)$ les notions des paragraphes 3.4.1 à 3.4.5.

Nous verrons que les expressions régulières construites avec des termes représentant des relations constituent une sorte de « langage de programmation ». Le but principal de cette

section est de permettre au lecteur de se familiariser avec ces expressions, comme il a pu le faire dans la section 3.4 avec les expressions régulières représentant des langages.

5.1.2. La relation définie par un programme

La notion de « programme » dont il est parfois question dans ce cours n'est pas une notion formelle, faisant l'objet d'une définition mathématique. Nous ne définissons pas de langage de programmation. Nous utilisons seulement quelques notations usuelles de la programmation impérative : déclaration de variables, instructions d'affectations, séquences d'instructions, boucles **while**, instructions de test **if**. Toutes les variables d'un programme sont globales et déclarées au début du programme, voire dans le contexte. Il n'y a pas de varables locales, déclarées dans des « blocs ». Exemple :

(1)
$$\begin{array}{l}\text{Var } x, y : \mathbb{N};\\ \textbf{while } y \neq 0 \textbf{ do}\\ \quad y := y - 1;\\ \quad x := x + 1\\ \textbf{od}.\end{array}$$

Les mots symétriques **do**, **od** sont des parenthèses, respectivement ouvrante et fermante.

Ce que nous appelons *état*, ou plus précisément *état de mémoire*, d'un tel programme est une combinaison de valeurs de ses variables. Dans l'exemple (1), un état de mémoire du programme est un couple de valeurs $(x, y) \in \mathbb{N} \times \mathbb{N}$. L'ensemble des états de mémoire de ce programme (on parle aussi de « l'espace » des états de mémoire) est l'ensemble

$$E = \mathbb{N} \times \mathbb{N}.$$

En partant d'un état de mémoire initial d'un programme, on peut (si l'état initial le permet) exécuter le programme et aboutir à un état de mémoire final. L'ensemble des couples d'états de mémoire qui sont formés ainsi par l'état de mémoire initial et l'état de mémoire final d'une exécution du programme est une relation dans l'ensemble des états de mémoire, qui caractérise « l'effet global » du programme. Dans l'exemple (1), il est évident que cette relation R est l'ensemble des couples de la forme

$$(x, y) \mapsto (x + y, 0)$$

tels que $(x, y) \in \mathbb{N} \times \mathbb{N}$ ([8]). Avec la notation ensembliste standard :

$$R = \{(x, y) \mapsto (x + y, 0) \mid (x, y) \in \mathbb{N} \times \mathbb{N}\}.$$

Nous appelons cette relation R la *relation définie par le programme*.

Il se trouve, dans cet exemple, que cette relation est une fonction et que le domaine de cette fonction est l'ensemble $E = \mathbb{N} \times \mathbb{N}$ tout entier.

Dans d'autres exemples, le domaine de R n'est pas l'ensemble des états de mémoire E tout entier, mais seulement un sous-ensemble propre de E, parce que, pour certains états initiaux z, l'exécution du programme ne se termine pas. Il y a deux raisons possibles pour cela :

(a) Le programme reste bloqué en cours d'exécution, parce que l'état de mémoire à ce stade ne permet pas d'exécuter l'instruction suivante — par exemple on ne peut pas exécuter l'instruction $y := y - 1$ sur une variable y de type $\mathbb{N}$ lorsque l'état de mémoire est tel que $y = 0$;

---

[8] On rappelle que $a \mapsto b$ est une notation du couple (a, b). Donc une expression de la forme $(x, y) \mapsto (x', y')$ représente le couple de couples $((x, y), (x', y'))$.

(b) L'exécution du programme tourne dans une boucle infinie.

Dans les deux cas, il n'existe pas d'état de mémoire z' tel que $(z, z') \in R$. Cela fait partie de la définition de R, qui est l'ensemble des couples d'états de mémoire (z, z') pour lesquels il y a une exécution *achevée* et *finie* du programme partant de z et aboutissant à z'.

5.1.3. Expression régulière de la relation définie par un programme

Un programme impératif (au sens simple décrit ci-dessus) est un assemblage d'instructions simples telles que des affectations (par exemple $x := x + 1$). La séquence d'opérations simples effectuées dans une exécution du programme dépend du résultat de certains « tests » effectués à certains moments, par exemple le test de la condition $y \neq 0$ dans le programme 5.1.2(1).

Nous allons voir qu'à chaque instruction (affectation) et à chaque condition testée dans un programme on peut associer de manière naturelle une relation dans l'ensemble E des états de mémoire du programme, de telle manière que la relation R définie par le programme (5.1.2) s'exprime sous la forme d'une expression régulière construite avec les relations associées aux instructions et conditions testées et les opérations de l'algèbre de Kleene $\mathcal{K} = \mathcal{R}(E)$.

Ce paragraphe n'est qu'une introduction à ce genre d'expression régulière, au moyen de l'exemple de programme 5.1.2(1).

Relations associées aux instructions simples. Étant donné un état $(x, y) \in \mathbb{N} \times \mathbb{N}$, l'exécution de l'instruction $x := x + 1$ transforme cet état en l'état $(x+1, y)$. Cette opération « incrémente » la première composante d'un état. Nous lui associons la relation Inc_1 suivante dans l'ensemble $E = \mathbb{N} \times \mathbb{N}$:

$$Inc_1 = \{(x, y) \mapsto (x + 1, y) \mid (x, y) \in \mathbb{N} \times \mathbb{N}\}. \tag{1}$$

L'instruction $y := y - 1$ peut être exécutée, à partir d'un état de mémoire (x, y), si et seulement si $y \neq 0$. Elle « décrémente » la deuxième composante de l'état et nous lui associons la relation Dec_2 suivante dans E:

$$Dec_2 = \{(x, y) \mapsto (x, y - 1) \mid (x, y) \in \mathbb{N} \times \mathbb{N} \text{ et } y \neq 0\}. \tag{2}$$

Les relations Inc_1 et Dec_2 sont des fonctions. Le domaine de la première est l'ensemble $E = \mathbb{N} \times \mathbb{N}$, alors que le domaine de la seconde est le sous-ensemble $\mathbb{N} \times (\mathbb{N} - \{0\})$ de E.

Si les instructions $y := y + 1$, $x := x - 1$ figuraient dans le programme, elles auraient pour relations correspondantes dans E les relations

$$Inc_2 = \{(x, y) \mapsto (x, y + 1) \mid (x, y) \in \mathbb{N} \times \mathbb{N}\}.$$
$$Dec_1 = \{(x, y) \mapsto (x - 1, y) \mid (x, y) \in \mathbb{N} \times \mathbb{N} \text{ et } x \neq 0\}.$$

Si l'instruction $x := x + y$ figurait dans le programme, la relation correspondante (à laquelle nous ne donnons pas de nom particulier) serait l'ensemble de couples d'états de mémoire

$$\{(x, y) \mapsto (x + y, y) \mid (x, y) \in \mathbb{N} \times \mathbb{N}\}.$$

Relation associée à une séquence d'instructions. Posons-nous la question suivante: quelle relation dans l'ensemble $E = \mathbb{N} \times \mathbb{N}$ correspond à la *séquence d'instructions*

$$y := y - 1;\ x := x + 1 \tag{3}$$

du programme 5.1.2(1) et comment cette relation peut-elle s'exprimer au moyen des relations Dec_2, Inc_1 qui correspondent à ces instructions?

La réponse est assez évidente. Soit S la relation dans l'ensemble E définie par le « programme » (3). Un couple d'états $(x, y) \mapsto (x', y')$ appartient à cette relation S

si et seulement si l'on peut passer de (x, y) à (x', y') en effectuant successivement les opérations: décrémenter la deuxième composante puis incrémenter la première. Cela est possible évidemment si et seulement si $y > 0$ et $(x', y') = (x + 1, y - 1)$, mais l'essentiel, dans la notion de séquencement ou exécution successive de ces deux opérations, est qu'il existe un état intermédiaire (a, b) tel que

$$(x, y) \mapsto (a, b) \in Dec_2 \text{ et } (a, b) \mapsto (x', y') \in Inc_1.$$

Autrement dit, la relation S n'est autre que la relation composée

$$Dec_2 Inc_1.$$

Ceci est une propriété générale des programmes: *le séquencement d'instructions se traduit par le produit de composition des relations correspondantes dans l'ensemble des états de mémoire*.

Relation associée à un test. Considérons maintenant de façon générale les opérations de « test » que comporte en général l'exécution d'un programme. Dans une telle opération, il s'agit de distinguer les états de mémoire qui satisfont à une certaine condition, donc qui appartiennent à un sous-ensemble A déterminé de l'ensemble des états de mémoire E. Par exemple, les états de mémoire du programme 5.1.2(1) qui satisfont à la condition $y \neq 0$ appartiennent au sous-ensemble

$$A = \{(x, y) \mid (x, y) \in \mathbb{N} \times \mathbb{N} \text{ et } y \neq 0\} \tag{4}$$

de l'ensemble $E = \mathbb{N} \times \mathbb{N}$.

Au lieu de considérer l'ensemble A des états qui vérifient une certaine condition, il est plus utile de considérer la *relation identique* Id_A de ce sous-ensemble, car celle-ci est une relation dans E qui peut se combiner suivant les opérations de l'algèbre de Kleene $\mathcal{R}(E)$ avec les relations associées aux instructions d'affectation d'un programme. C'est cela qui permet d'exprimer la relation définie par un programme sous la forme d'une expression régulière. Dans l'exemple (4), on a

$$\mathrm{Id}_A = \{(x, y) \mapsto (x, y) \mid (x, y) \in \mathbb{N} \times \mathbb{N} \text{ et } y \neq 0\}.$$

La relation définie par le programme 5.1.2(1) peut s'exprimer plus simplement si l'on considère l'ensemble complémentaire de (4), à savoir l'ensemble des états de mémoire (x, y) qui vérifient la condition $y = 0$. Cet ensemble s'écrit simplement

$$\{(x, 0) \mid x \in \mathbb{N}\}$$

et nous désignerons par Nul_2 sa relation identique:

$$Nul_2 = \{(x, 0) \mapsto (x, 0) \mid x \in \mathbb{N}\}.$$

Expression régulière du programme 5.1.2(1). Nous avons vu au §5.1.2 que la relation définie par le programme 5.1.2(1) est la relation

$$R = \{(x, y) \mapsto (x + y, 0) \mid (x, y) \in \mathbb{N} \times \mathbb{N}\}.$$

Nous affirmons que cette relation peut s'exprimer au moyen des relations Inc_1, Dec_2, Nul_2 définies ci-dessus et des opérations de l'algèbre de Kleene $\mathcal{K} = \mathcal{R}(E)$ par l'expression:

$$R = (Dec_2 Inc_1)^* Nul_2. \tag{5}$$

Nous n'allons pas démontrer cette assertion maintenant et le lecteur n'est pas censé voir maintenant qu'elle est vraie. Comme nous l'avons dit précédemment, le but de la présente

section est de permettre au lecteur de se familiariser suffisamment avec l'algèbre de Kleene des relations dans un ensemble E pour que, en particulier, l'égalité (5) devienne évidente.

5.1.4. Trajectoires d'une relation dans un ensemble

Nous commençons notre étude de l'algèbre de Kleene $\mathcal{R}(E)$, pour un ensemble E quelconque, par celle des puissances R^n d'une relation R dans E, dont la définition est rappelée au §5.1.1.

Remarque préliminaire. Si E' est un ensemble tel que $E \subset E'$, toute relation R dans E est aussi une relation dans E'. La relation R^0 peut être la relation Id_E ou la relation $\mathrm{Id}_{E'}$, selon que l'on considère le monoïde $\mathcal{R}(E)$ ou le monoïde $\mathcal{R}(E')$. Il y a donc une certaine ambiguïté dans la notation R^0. Mais pratiquement, l'ensemble E considéré sera toujours clairement indiqué dans le contexte.

La même remarque peut être faite à propos de R^*, puisque R^* est la réunion de toutes les puissances R^n, donc contient R^0.

Par contre, pour un entier naturel $n \neq 0$, la relation R^n ne dépend pas du monoïde $\mathcal{R}(E)$ ou $\mathcal{R}(E')$ considéré.

Définition. Si R est une relation dans un ensemble E, nous appelons *trajectoire* de R dans E toute séquence $x \in \mathrm{W}(E)$ telle que $x \neq \Lambda$ et telle que

$$\forall k \in [1 \,..\, \lg(x) - 1] : \quad (x[k], x[k+1]) \in R. \tag{1}$$

Notons que toute séquence $x \in \mathrm{W}_1(E)$ est une trajectoire de R, puisque la condition (1) est trivialement satisfaite si $\lg(x) = 1$. Cette notion est illustrée par la figure 5.1.

Si a et b sont deux éléments de E, nous disons qu'une trajectoire x d'une relation R dans E *va de a à b* si

$$x[1] = a \ \text{ et } \ x[\lg(x)] = b.$$

En particulier, pour tout $a \in E$, la séquence élémentaire $x = \langle\!\langle\, a \,\rangle\!\rangle$ est une trajectoire de R allant de a à a.

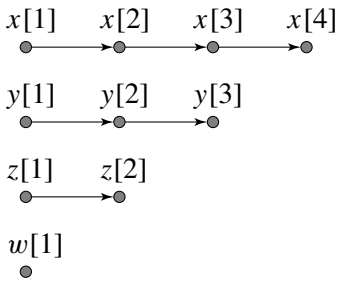


Figure 5.1
Trajectoires x, y, z, w de longueurs 4, 3, 2, 1 d'une relation R dans un ensemble E. Chaque flèche représente un couple d'éléments de E appartenant à R. Tous les points ne représentent pas nécessairement des éléments distincts de E.

Théorème. Soient R une relation dans E, a et b des éléments de E. On a

$$\forall n \in \mathbb{N} : \quad (a, b) \in R^n \ \Leftrightarrow \ \left(\begin{array}{l} \text{il existe une trajectoire de } R \text{ dans } E \\ \text{de longueur } n+1 \text{ allant de } a \text{ à } b \end{array} \right). \tag{2}$$

$$(a, b) \in R^* \ \Leftrightarrow \ \left(\begin{array}{l} \text{il existe une trajectoire de } R \text{ dans } E \\ \text{de longueur} \geq 1 \text{ allant de } a \text{ à } b \end{array} \right). \tag{3}$$

Démonstration. Ce théorème est un cas particulier du théorème 2.3.10 sur l'évaluation d'une séquence de relations. Étant donné une relation R dans E, la relation R^n peut s'écrire en effet

$$R^n = \prod_{i=1}^{n} S[i] \quad \text{avec} \quad S[i] = R \ \text{ pour } i = 1, \ldots, n.$$

Plus précisément, comme R^n est la n-ième itérée de R ($R^n = \mathrm{it}_R(n)$) dans le monoïde $\mathcal{R}(E)$, on a, d'après le théorème de 2.3.7, $R^n = eval_{\mathcal{R}(E)}(S)$, où S est la séquence constante $\big(\lambda i \in [1 \,..\, n] : R\big)$. La formule (2) découle donc du théorème 2.3.10, appliqué à la séquence de relations S. La formule (3) découle de (2) puisque $(a, b) \in R^*$ équivaut à $\exists n \in \mathbb{N}$: $(a, b) \in R^n$.

5.1.5. Exercice

Dans chacun des exercices suivants, (1) à (6), on donne un ensemble E et une relation R dans E. Il est demandé de caractériser, pour tout $n \in \mathbb{N}$, la relation R^n dans E par une égalité de la forme

$$R^n = \{(x, y) \in E \times E \mid P(x, y)\},$$

où $P(x, y)$ est une proposition exprimant une propriété du couple (x, y). Il est entendu que l'on considère les puissances de R dans le monoïde $\mathcal{R}(E)$, donc que $R^0 = \mathrm{Id}_E$. On demande de caractériser de la même manière la relation R^* dans E et de vérifier que l'on a $R^*R^* = R^*$, conformément à la formule 3.2.7 (2).

(1) $E = \mathbb{N}$. $R = \{(x, x + 1) \mid x \in \mathbb{N}\}$.

(2) $E = \mathbb{N}$. $R = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid x < y\}$.

(3) $E = \mathbb{R}$. $R = \{(x, y) \in \mathbb{R} \times \mathbb{R} \mid x < y\}$.

(4) $E = \mathbb{R}$. $R = \{(x, -x) \mid x \in \mathbb{R}\}$.

(5) $E = \mathbb{R}$. $R = \{(x, -x) \mid x \in \mathbb{R} \text{ et } x \geq 0\}$.

(6) $E = \mathbb{R}$. $R = \{(x, y) \in \mathbb{R} \times \mathbb{R} \mid |x| < |y| \text{ et } xy < 0\}$.

5.1.6. Fermeture réflexive et transitive d'une relation

Les propriétés de *réflexivité* et de *transitivité* d'une relation R dans un ensemble E ont été définies au §1.4.1 (relations d'ordre). Rappelons que R est dite *réflexive* dans E si l'on a $(x, x) \in R$ pour tout $x \in E$, autrement dit si $\mathrm{Id}_E \subset R$.

R est dite *transitive* si l'on a $\forall x \forall y \forall z$: $(x\,R\,y \text{ et } y\,R\,z) \Rightarrow x\,R\,z$. Cette propriété peut s'exprimer simplement par $RR \subset R$. (Voir [A].)

La fermeture de Kleene R^* d'une relation R dans E, au sens de l'algèbre de Kleene $\mathcal{K} = \mathcal{R}(E)$, possède les propriétés suivantes, d'après 3.2.6:

(1) $\mathrm{Id}_E \subset R^*$.
(2) $R^*R^* \subset R^*$.
(3) $R \subset R^*$.
(4) Pour toute relation M dans E: $(\mathrm{Id}_E \subset M \text{ et } MM \subset M \text{ et } R \subset M) \Rightarrow R^* \subset M$.

Les trois premières peuvent s'énoncer en disant que R^* est une relation réflexive et transitive qui contient R. La quatrième signifie que toute relation réflexive et transitive M dans E qui contient R ($R \subset M$) contient R^*. On exprime le tout en disant que R^* est *la plus petite relation réflexive et transitive dans E contenant R*.

Pour ces raisons, la relation R^* est appelée souvent la *fermeture réflexive et transitive* de R dans E.

5.1.7. Les relations dans un ensemble vues comme des instructions

Aux paragraphes 5.1.2 et 5.1.3, nous avons introduit la relation R définie par un programme dans l'espace des états de mémoire de celui-ci et les relations dans cet espace correspondant aux instructions et conditions du programme.

Pour se familiariser avec l'algèbre de Kleene des relations dans un ensemble E, il est commode de considérer n'importe quelle relation R dans E comme étant la relation définie par un programme — un programme dont on ne connaît pas le texte et dont tout ce qu'on connaît est la relation qu'il définit. Cela revient à considérer n'importe quelle relation R comme l'ensemble des couples (état initial, état final) des exécutions d'un programme. Ce programme n'étant d'ailleurs pas précisé, on peut toujours admettre qu'il se compose d'une seule instruction. Finalement, on peut aller jusqu'à employer le mot « programme » ou le mot « instruction » à la place du mot « relation », ce qui revient à ne plus considérer dans un programme ou dans une instruction que la relation correspondante dans un espace d'états. Il faut seulement prendre garde à ne pas se laisser entraîner par cette terminologie à attribuer aux relations et à leurs combinaisons des propriétés qu'elles n'ont pas.

Dans le cas d'une relation dans E qui est une fonction de domaine E, par exemple la relation

$$Inc_1 = \{(x, y) \mapsto (x + 1, y) \mid (x, y) \in E\}$$

dans l'ensemble $E = \mathbb{N} \times \mathbb{N}$ (5.1.3), l'interprétation de la relation comme une « instruction » est si évidente qu'elle ne peut pas donner lieu à interprétation erronée. Pour tout état initial (x, y), l'exécution de l'instruction conduit à l'état $(x + 1, y)$.

Dans le cas d'une relation dans E qui est une fonction dont le domaine n'est pas l'ensemble E tout entier, par exemple la relation

$$R = Dec_2 = \{(x, y - 1) \mapsto (x, y) \mid (x, y) \in E \text{ et } y \neq 0\}$$

dans $E = \mathbb{N} \times \mathbb{N}$, l'interprétation de la relation en tant qu'instruction est aussi claire mais nécessite tout de même une précision. Pour un état initial qui n'appartient pas au domaine de R, l'instruction en question ne peut pas être exécutée. On peut imaginer la machine qui doit l'exécuter comme étant « bloquée ».

L'interprétation d'une relation R (dans un ensemble E) en tant qu'instruction demande à être bien précisée dans le cas où R n'est pas une fonction. Pour un état initial z qui n'appartient pas au domaine de R, l'instruction ne peut pas être exécutée, comme dans le cas précédent. Pour un état initial $z \in \mathrm{Dom}R$ pour lequel il existe un seul état z' tel que $(z, z') \in R$, l'exécution de R conduit évidemment à cet état z'. C'est lorsqu'il existe plusieurs états z' tels que $(z, z') \in R$, voire une infinité de tels états, que l'interprétation de R comme « instruction » n'est pas évidente. Dans ce cas on ne peut plus parler de *l'exécution* (au singulier) de l'instruction R, mais de différentes manières possibles d'exécuter cette instruction, ou de différentes exécutions possibles de R, chacune d'elles conduisant à un état différent.

Lorsqu'une telle « instruction » est combinée avec d'autres instructions du même genre dans un programme, une exécution du programme entier se décompose en une séquence d'exécutions possibles de ses instructions. Mais, en général, seules certaines des exécutions possibles des instructions individuelles du programme peuvent se combiner pour former une exécution du programme tout entier. Le cas le plus simple est celui d'un « programme » de la forme RS, produit de composition de deux relations dans E, discuté ci-dessous.

Produit de deux relations. Soient R, S des relations dans E. Une exécution de « l'instruction composée » RS, à partir d'un état initial $x \in E$, se compose d'une exécution de R, conduisant à un état y, suivie d'une exécution de S à partir de cet état y, conduisant à un état z. Nous ne faisons ainsi que traduire la formule

$$(x, z) \in RS \Leftrightarrow \exists y((x, y) \in R \text{ et } (y, z) \in S).$$

Une exécution de S à partir d'un état y n'est possible que si $y \in \mathrm{Dom}S$. Donc, à partir d'un état initial x, il est possible d'exécuter RS si et seulement s'il est possible d'exécuter

R d'une manière qui conduit à un état y tel que $y \in \mathrm{Dom}S$. En d'autres termes, on a la formule suivante, caracérisant le domaine de la relation RS:

$$x \in \mathrm{Dom}(RS) \Leftrightarrow \exists y(x\, R\, y \text{ et } y \in \mathrm{Dom}S). \tag{1}$$

Démonstration:

$$\begin{aligned} x \in \mathrm{Dom}(RS) &\Leftrightarrow \exists z((x, z) \in RS) \\ &\Leftrightarrow \exists z \exists y(x\, R\, y \text{ et } y\, S\, z) \\ &\Leftrightarrow \exists y \exists z(x\, R\, y \text{ et } y\, S\, z) \\ &\Leftrightarrow \exists y(x\, R\, y \text{ et } \exists z(y\, S\, z)) \\ &\Leftrightarrow \exists y(x\, R\, y \text{ et } y \in \mathrm{Dom}S). \end{aligned}$$

De la formule (1) on tire que $x \in \mathrm{Dom}(RS) \Rightarrow x \in \mathrm{Dom}R$, d'où la formule

$$\mathrm{Dom}(RS) \subset \mathrm{Dom}R. \tag{2}$$

Celle-ci s'explique aussi intuitivement en termes d'exécutions: si $x \in \mathrm{Dom}(RS)$, l'instruction composée RS peut être exécutée à partir de l'état x et, comme une exécution de RS commence par une exécution de R, cela signifie que R peut être exécutée à partir de x, donc que $x \in \mathrm{Dom}R$.

Par contraposition, on a $x \notin \mathrm{Dom}R \Rightarrow x \notin \mathrm{Dom}(RS)$. L'instruction RS ne peut pas être exécutée à partir d'un état de mémoire x tel que $x \notin \mathrm{Dom}R$.

Instructions `NOP`. Le symbole `NOP`, abréviation de « no operation », est utilisé dans certains langages de programmation comme instruction signifiant « ne rien faire ». Mais « ne rien faire » signifie: ne rien changer à l'état de mémoire du programme. En tant que relation dans un ensemble E, l'opération `NOP` peut *toujours* être exécutée, c'est-à-dire à partir de tout élément $x \in E$. Le domaine de cette relation est donc égal à E et comme son exécution ne change pas l'état, on voit que cette relation n'est autre que la fonction identique Id_E, que l'on pourrait noter aussi $\mathtt{NOP}_E$. Le fait qu'on puisse insérer arbitrairement de telles instructions dans un programme sans changer l'effet de celui-ci correspond à la propriété de neutralité de Id_E pour le produit de composition avec une relation R quelconque dans E:

$$\mathrm{Id}_E\, R = R\, \mathrm{Id}_E = R.$$

NB. Si $E \neq \emptyset$, l'opération « ne rien faire » dans E ($\mathtt{NOP}_E$), en tant que relation dans E, n'est donc pas la relation $\emptyset$, comme on pourrait être tenté de le penser. La relation $\emptyset$, dont le domaine est $\emptyset$, n'est exécutable à partir d'*aucun* élément de E. En outre elle n'est pas neutre puisque, pour toute relation R dans E, on a $\emptyset R = R\,\emptyset = \emptyset$ (3.2.3(4)).

Réunions de relations. Considérons la réunion

$$U = \bigcup_{i \in I} R_i$$

d'une famille $(R_i)_{i \in I}$ de relations dans E. Cette réunion est une relation dans E, comme nous l'avons déjà relevé (5.1.1). Par définition de la réunion d'une famille d'ensembles (1.3.19), on a

$$(x, y) \in U \Leftrightarrow \exists i(i \in I \text{ et } (x, y) \in R_i).$$

On peut commenter cette formule en disant qu'une exécution de l'instruction U, à partir d'un état initial x, consiste à choisir un élément $i \in I$ tel que l'instruction R_i soit exécutable à partir de x et à exécuter celle-ci.

L'instruction U est exécutable à partir d'un état x si et seulement s'il existe un $i \in I$ tel que l'instruction R_i soit exécutable à partir de x. Autrement dit, on a la formule suivante (démonstration dans [A]):

$$\mathrm{Dom}\Big(\bigcup_{i\in I} R_i\Big) = \bigcup_{i\in I}\big(\mathrm{Dom}\, R_i\big). \tag{3}$$

L'exemple principal de réunion d'une famille infinie de relations est la fermeture de Kleene d'une relation R dans E,

$$R^* = \bigcup_{n\in\mathbb{N}} R^n.$$

Une exécution de l'instruction R^*, à partir d'un état $a \in E$, consiste à choisir un entier $n \in \mathbb{N}$ tel que l'on puisse exécuter R^n à partir de a et à exécuter R^n. D'après le théorème de 5.1.4, on peut exécuter R^n à partir de a si et seulement s'il existe une trajectoire $x = \langle\!\langle\, x[1], \ldots, x[n+1] \,\rangle\!\rangle$ de R dans E telle que $x[1] = a$. Une exécution possible de R^n à partir de a conduit aux dernier élément, $b = x[n+1]$, d'une telle séquence. En particulier, on peut toujours exécuter $R^0 = \mathrm{Id}_E$, c'est-à-dire « ne rien faire ».

5.1.8. Instructions de test

Nous appelons *instruction de test* dans un ensemble E toute relation dans E qui est de la forme Id_A avec $A \subset E$. Les instructions de test dans E sont donc les fonctions identiques des parties de E. Nous avons vu un exemple d'une telle instruction au §5.1.3, à savoir la relation

$$Nul_2 = \{(x, y) \mapsto (x, y) \mid (x, y) \in \mathbb{N} \times \mathbb{N} \text{ et } y = 0\},$$

qui est la relation identique du sous-ensemble $A = \mathbb{N} \times \{0\}$ de l'ensemble $E = \mathbb{N} \times \mathbb{N}$.

Cette terminologie n'est pas très usuelle[9]. En général une « opération de test » est conçue comme une opération qui retourne pour chaque argument donné une valeur « booléenne »: vrai ou faux, 0 ou 1, etc. Par exemple, étant donné un ensemble $A \subset E$, la fonction F de domaine E telle que

$$\forall x \in E : \quad F(x) = \begin{cases} 1 & \text{si } x \in A; \\ 0 & \text{si } x \notin A \end{cases}$$

pourrait être appelée la fonction de test de la condition $x \in A$ pour les éléments x de E. Ce n'est pas cela que nous appelons une *instruction de test dans* E. Pour le même ensemble A, l'instruction de test $G = \mathrm{Id}_A$ est la fonction G de *domaine* A telle que

$$G(x) = x \quad \text{pour tout } x \in A.$$

Cette instruction ne peut pas être exécutée à partir d'un état x tel que $x \notin A$. Elle peut être exécutée seulement à partir des états $x \in A$, et son exécution laisse ces états inchangés.

5.1.9. Composition d'une instruction de test avec une autre instruction

Les instructions de test (5.1.8) ne sont utiles que combinées avec d'autres instructions. Nous discutons dans ce paragraphe les relations composées de la forme

$$\mathrm{Id}_A R, \qquad R\,\mathrm{Id}_A,$$

où R est une relation dans un ensemble E et $A \subset E$.

---

[9] Elle est empruntée à l'ouvrage de R. W. Floyd et R. Beigel, *The Language of Machines*, Computer Science Press, 1994.

Relation $\mathrm{Id}_A R$. La relation $\mathrm{Id}_A R$ est caractérisée par la formule

(1) $$(x, y) \in \mathrm{Id}_A R \Leftrightarrow (x \in A \text{ et } (x, y) \in R).$$

Démonstration :

$$\begin{aligned}(x, y) \in \mathrm{Id}_A R &\Leftrightarrow \exists a((x, a) \in \mathrm{Id}_A \text{ et } (a, y) \in R)\\ &\Leftrightarrow \exists a(x \in A \text{ et } a = x \text{ et } (a, y) \in R) \qquad (1.3.8)\\ &\Leftrightarrow (x \in A \text{ et } (x, y) \in R).\end{aligned}$$

La relation $\mathrm{Id}_A R$ est donc un sous-ensemble de R, à savoir l'ensemble des couples $(x, y) \in R$ tels que $x \in A$. Un exemple est donné dans la figure 5.2.

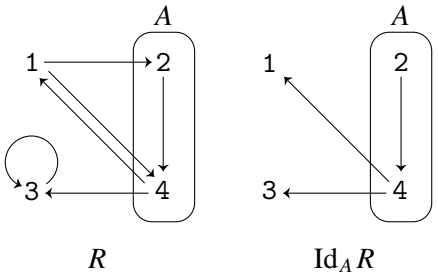


Figure 5.2
$\mathrm{Id}_A R$ est l'ensemble des couples (x, y) tels que $(x, y) \in R$ et $x \in A$.

Si l'on interprète la relation $\mathrm{Id}_A R$ come une « instruction », on peut dire que cette instruction peut être exécutée à partir d'un état x si et seulement si cet état appartient à la fois à l'ensemble A et au domaine de R. Autrement dit, on a la formule suivante (démonstration dans [A]) :

(2) $$\mathrm{Dom}(\mathrm{Id}_A R) = A \cap \mathrm{Dom}\, R.$$

À partir d'un état x tel que $x \in A$, les exécutions possibles de $\mathrm{Id}_A R$ sont celles de R. On peut dire que le rôle de Id_A devant R, dans la combinaison $\mathrm{Id}_A R$, est seulement d'empêcher l'exécution de R à partir des états qui n'appartiennent pas à A. Si le domaine de R est contenu dans A, l'instruction Id_A devant R n'a pas d'effet. Formellement (démonstration dans [A]), on a :

(3) $$\mathrm{Dom}\, R \subset A \;\Rightarrow\; \mathrm{Id}_A R = R.$$

La relation $R\,\mathrm{Id}_A$. La combinaison $R\,\mathrm{Id}_A$, symétrique de la précédente, est caractérisée par la formule

(4) $$(x, y) \in R\,\mathrm{Id}_A \Leftrightarrow ((x, y) \in R \text{ et } y \in A).$$

On peut dire que le rôle de Id_A après R, dans la combinaison $R\mathrm{Id}_A$, est de ne « permettre » que les exécutions de R qui aboutissent à un état y appartenant à l'ensemble A.

Exemple. Au paragraphe 5.1.3, nous avons affirmé que la relation

(5) $$R = \{(x, y) \mapsto (x + y, 0) \mid (x, y) \in \mathbb{N} \times \mathbb{N}\}$$

dans l'ensemble $E = \mathbb{N} \times \mathbb{N}$ qui est définie par le programme 5.1.2(1), peut s'exprimer par

(6) $$R = (Dec_2 Inc_1)^* Nul_2$$

au moyen des relations

$$\begin{aligned}Inc_1 &= \{(x, y) \mapsto (x + 1, y) \mid (x, y) \in \mathbb{N} \times \mathbb{N}\}\\ Dec_2 &= \{(x, y) \mapsto (x, y - 1) \mid (x, y) \in \mathbb{N} \times \mathbb{N} \text{ et } y \neq 0\}\\ Nul_2 &= \{(x, 0) \mapsto (x, 0) \mid x \in \mathbb{N}\}.\end{aligned}$$

Nous voulons maintenant démontrer l'égalité (6). La relation Nul_2 est la fonction identique du sous-ensemble $A = \mathbb{N} \times \{0\}$ de E. Soient

$$z = (x, y), \quad z' = (x', y')$$

deux éléments de E. D'après la formule (4), on a

$$\begin{aligned}(z, z') \in (Dec_2Inc_1)^*Nul_2 &\Leftrightarrow \big((z, z') \in (Dec_2Inc_1)^* \text{ et } z' \in \mathbb{N} \times \{0\}\big)\\ &\Leftrightarrow \big((z, z') \in (Dec_2Inc_1)^* \text{ et } y' = 0\big)\\ &\Leftrightarrow \big(\exists n \in \mathbb{N} : (z, z') \in (Dec_2Inc_1)^n \text{ et } y' = 0\big).\end{aligned}$$

À partir de l'état $z = (x, y)$, on peut exécuter n fois Dec_2Inc_1 si et seulement si $y \geq n$, et cette exécution conduit à l'état $(x + n, y - n)$. Formellement, pour tout $n \in \mathbb{N}$, on a

$$(z, z') \in (Dec_2Inc_1)^n \Leftrightarrow (y \geq n \text{ et } x' = x + n \text{ et } y' = y - n).$$

Par suite, il existe un seul $n \in \mathbb{N}$ tel que l'on ait $(z, z') \in (Dec_2Inc_1)^n$ et $y' = 0$, à savoir $n = y$. On a donc

$$\begin{aligned}(z, z') \in (Dec_2Inc_1)^*Nul_2 &\Leftrightarrow (z, z') \in (Dec_2Inc_1)^y\\ &\Leftrightarrow (y \geq y \text{ et } x' = x + y \text{ et } y' = y - y)\\ &\Leftrightarrow z' = (x + y, 0)\\ &\Leftrightarrow (z, z') \in R. \qquad (5)\end{aligned}$$

L'égalité (6) est ainsi démontrée.

On peut dire que l'exécution du programme $(Dec_2Inc_1)^*Nul_2$, à partir d'un état de mémoire $z = (x, y)$, consiste à exécuter l'instruction Dec_2Inc_1 le nombre de fois qu'il faut pour aboutir à un état (x', y') satisfaisant la condition $y' = 0$, c'est-à-dire un état (x', y') à partir duquel on peut exécuter l'instruction de test Nul_2. Ce nombre de fois est égal à y.

On peut relever que la relation $(Dec_2Inc_1)^*$ n'est pas une fonction, mais que la relation composée $(Dec_2Inc_1)^*Nul_2$ en est une.

5.1.10. Instructions « while A do R »

Exemple 1. Soient $a, b, c, \ldots, i$ des éléments distincts d'un ensemble E et soit R la relation dans E représentée par le diagramme de la figure 5.3. Les flèches en trait plein et en pointillé représentent les couples d'éléments de E appartenant à R. On considère particulièrement le sous-ensemble $A = \{a, b, c, d\}$ de E. Les flèches en trait plein correspondent aux couples $(x, y) \in R$ tels que $x \in A$, autrement dit appartenant à la relation $\mathrm{Id}_A R$. Les flèches en pointillé correspondent aux couples $(x, y) \in R$ tels que $x \notin A$.

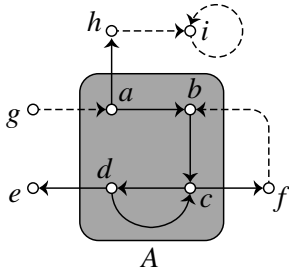


Figure 5.3

Considérons les trajectoires suivantes de R (5.1.4):

$$\begin{aligned}&\langle\!\langle\, h \,\rangle\!\rangle\\ &\langle\!\langle\, a, h \,\rangle\!\rangle\\ &\langle\!\langle\, a, b, c, f \,\rangle\!\rangle\\ &\langle\!\langle\, a, b, c, d, e \,\rangle\!\rangle\\ &\langle\!\langle\, a, b, c, d, c, f \,\rangle\!\rangle\\ &\langle\!\langle\, a, b, c, d, c, d, e \,\rangle\!\rangle.\end{aligned}$$

Ce sont des trajectoires de R avec la propriété suivante : leur dernier élément et seulement leur dernier élément appartient à l'ensemble $\overline{A}$, complémentaire de A par rapport à E. Une

telle trajectoire de R, de longueur $n+1$, est une séquence $\langle\!\langle\, x[1], \ldots, x[n+1] \,\rangle\!\rangle$ d'éléments de E telle que :

$$\forall k \in [1 \,..\, n] : \quad (x[k], x[k{+}1]) \in R \text{ et } x[k] \in A;\\ x[n{+}1] \in \overline{A}.$$

D'après 5.1.9(1), la première de ces conditions peut s'écrire

$$\forall k \in [1 \,..\, n] : \quad (x[k], x[k{+}1]) \in \mathrm{Id}_A R.$$

Les trajectoires de R considérées sont donc des trajectoires de la relation $\mathrm{Id}_A R$. Ce sont plus précisément les trajectoires de $\mathrm{Id}_A R$ dont le dernier élément appartient à $\overline{A}$.

Considérons maintenant l'ensemble des couples (a, b) d'éléments de E qui sont le premier et le dernier élément d'une trajectoire de R du genre considéré et désignons cet ensemble par S. Les couples suivants appartiennent à S : (h, h), (a, h), (a, f), (a, e), mais aussi (b, f), (b, e), etc. D'après ce qui précède, on a

$$\begin{aligned} (x, y) \in S &\Leftrightarrow \big((x, y) \in (\mathrm{Id}_A R)^* \text{ et } y \in \overline{A}\big) && \text{théorème de 5.1.4} \\ &\Leftrightarrow (x, y) \in (\mathrm{Id}_A R)^* \mathrm{Id}_{\overline{A}} && 5.1.9(4) \end{aligned}$$

donc $S = (\mathrm{Id}_A R)^* \mathrm{Id}_{\overline{A}}$.

Remarquons finalement que la relation S peut être considérée comme la relation dans E qui correspond à l'instruction suivante : *tant que l'on est dans A exécuter R*, ou

while A **do** R.

Une telle instruction s'exprime donc par $(\mathrm{Id}_A R)^* \mathrm{Id}_{\overline{A}}$.

Exemple 2. Considérons à nouveau l'exemple de programme du §5.1.2 :

(1)
```
Var x, y : ℕ;
while y ≠ 0 do
    y := y − 1;
    x := x + 1
od.
```

D'après ce qui précède, la relation R dans l'ensemble $E = \mathbb{N} \times \mathbb{N}$ définie par ce programme peut s'écrire

$$R = (\mathrm{Id}_A Dec_2 Inc_1)^* \mathrm{Id}_{\overline{A}},$$

où A désigne l'ensemble des éléments (x, y) de E tels que $y \neq 0$. Cette expression de R peut se simplifier parce que A est le domaine de l'opération Dec_2, donc $\mathrm{Id}_A Dec_2 = Dec_2$ (5.1.9(3)). D'autre part, la relation $\mathrm{Id}_{\overline{A}}$ est celle que nous avons désigné précédemment par Nul_2. Nous retrouvons ainsi l'expression $R = (Dec_2 Inc_1)^* Nul_2$.

5.1.11. Exercice

Soient R_1, R_2 les relations dans l'ensemble E des états de mémoire d'un programme qui correspondent à deux instructions I_1, I_2 et soit A le sous-ensemble de E formé par les états de mémoire qui vérifient une certaine condition C. Exprimer les relations qui correspondent aux trois instructions « Ada » suivantes :

(1) **if** C **then** I_1 **else** I_2 **end if**.

(2) **if** C **then** I_1 **end if**.

(3)
```
loop
    I1
    exit when C
    I2
end loop.
```

5.2 Relation définie par une machine

5.2.1. Introduction

Nous allons introduire dans cette section le concept principal de ce chapitre: celui de machine. Le terme de « machine » évoque d'abord un appareil physique, une machine physique — de traitement de l'information bien entendu. Mais au courant du siècle passé (le vingtième) le terme a reçu aussi un sens mathématique, un sens de machine *abstraite* de traitement de l'information, et ceci avant même l'apparition des premiers ordinateurs.

Le but d'une telle définition est d'avoir un concept simple, précis et suffisamment général pour servir de modèle de la notion de machine dans l'étude de questions touchant à l'essence de cette notion et faisant abstraction des détails matériels ou logiciels des machines réelles. Il s'agit notamment de questions sur les limites des machines — sur ce qu'elles ne peuvent pas faire. Pour ce genre de questions, un modèle mathématique faisant abstraction de tous les détails non essentiels des machines réelles est ce qui convient.

La notion de machine que nous allons présenter remplacera aussi celle de « programme » dont nous avons parlé dans la section 5.1, dont le principal défaut est l'absence d'une définition formelle précise. Pour que cela soit bien clair, nous revenons encore sur le programme 5.1.2(1). Nous avons considéré comme une évidence, au §5.1.2, que la relation définie par ce programme dans l'ensemble $E = \mathbb{N} \times \mathbb{N}$ est la relation

$$R = \{(x, y) \mapsto (x + y, 0) \mid (x, y) \in E\}. \tag{1}$$

Cependant nous ne pourrions pas le démontrer, parce que ni la notion de programme ni celle de relation définie par un programme n'ont fait l'objet de définitions formelles. Ce sont deux notions intuitives.

Il en ira différemment avec la notion de machine. Comme introduction, nous présentons dans la figure 5.4 une machine qui correspond au programme en question. On voit qu'il s'agit simplement d'un accepteur fini sur l'ensemble $X = \{Inc_1, Dec_2, Nul_2\}$, lequel est un *ensemble de relations*. Une machine n'est pas autre chose qu'un accepteur fini sur un ensemble de relations. Nous utiliserons donc les notions du chapitre 4. Notamment, le langage accepté par un tel accepteur est un ensemble de *séquences de relations*. À partir de là, on peut donner une définition générale précise de la relation définie par une machine et *démontrer*, en particulier, que la relation définie par la machine de la figure 5.4 est bien la relation (1).

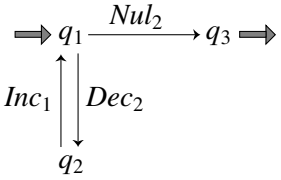


Figure 5.4

5.2.2. Accepteurs finis interprétés sur un monoïde M

Le rôle premier d'un accepteur fini sur un ensemble X est de représenter un langage sur X. C'est le rôle des accepteurs finis que nous avons étudié dans le chapitre 4 et que nous rappelons brièvement en prenant comme exemple l'accepteur (A, I, F) représenté dans la figure 5.5(a), avec $Q^A = \{q_1, q_2, q_3\}$, $T^A = \{(q_1, \beta, q_2), (q_2, \alpha, q_1), (q_1, \gamma, q_3)\}$, $I = \{q_1\}$, $F = \{q_3\}$.

Nous supposons que les états q_1, q_2, q_3 de cet accepteur sont distincts (ce sont par exemple les entiers 1, 2, 3). Nous supposons que α, β, γ sont trois éléments quelconques, distincts ou non, d'un ensemble X que nous ne précisons pas et qui peut avoir d'autres

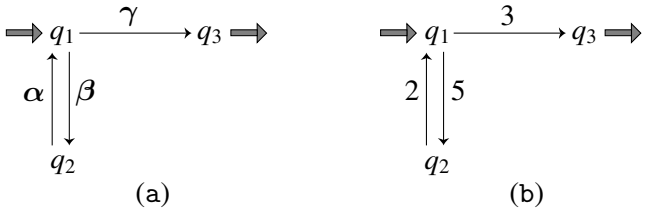


Figure 5.5

éléments. Le langage sur X qui est dit défini ou accepté par A est l'ensemble des séquences x sur X qui relient l'état q_1 à l'état q_3. Cet ensemble

$$L^A_{IF} \subset \mathrm{W}(X)$$

a pour éléments les séquences suivantes sur X :

$$\begin{array}{l}\langle\!\langle\, \gamma \,\rangle\!\rangle \\ \langle\!\langle\, \beta, \alpha, \gamma \,\rangle\!\rangle \\ \langle\!\langle\, \beta, \alpha, \beta, \alpha, \gamma \,\rangle\!\rangle \\ \langle\!\langle\, \beta, \alpha, \beta, \alpha, \beta, \alpha, \gamma \,\rangle\!\rangle \\ \ldots \text{ etc.}\end{array}$$

Supposons maintenant que l'ensemble X soit muni d'une opération binaire associative, notée au moyen d'un signe $*$, admettant un élément neutre e, autrement dit que $(X, *, e)$ soit un monoïde. Pour nous raccorder aux notations de la section 2.3, posons $X = M$. Alors, à toute séquence

$$x = \langle\!\langle\, x[1], \ldots, x[n] \,\rangle\!\rangle \in \mathrm{W}(X) = \mathrm{W}(M),$$

on peut associer l'élément

$$eval_M(x) = x[1] * x[2] * \cdots * x[n] \in M.$$

Si, pour chaque séquence x appartenant au langage L^A_{IF} accepté par A, on prend l'image de x par cette fonction d'évaluation, on obtient un sous-ensemble de M, à savoir l'ensemble

$$eval_M\langle L^A_{IF}\rangle = \{eval_M(x) \mid x \in L^A_{IF}\}.$$

Cet ensemble est le transformé du langage L^A_{IF} par la fonction d'évaluation $eval_M$. Lorsqu'on s'intéresse à cet ensemble plutôt qu'au langage L^A_{IF} lui-même, on dit que l'accepteur A est *interprété* dans le monoïde M.

Exemple 1. Supposons que $(M, *, e)$ soit le monoïde multiplicatif des entiers naturels $(\mathbb{N}, \cdot, 1)$ et que les éléments α, β, γ de M soient respectivement les entiers 2, 5, 3 (figure 5.5(b)). Le langage L^A_{IF} de l'accepteur considéré se compose alors des séquences

$$\begin{array}{l}\langle\!\langle\, 3 \,\rangle\!\rangle \\ \langle\!\langle\, 5, 2, 3 \,\rangle\!\rangle \\ \langle\!\langle\, 5, 2, 5, 2, 3 \,\rangle\!\rangle \\ \langle\!\langle\, 5, 2, 5, 2, 5, 2, 3 \,\rangle\!\rangle \\ \ldots \text{ etc.}\end{array}$$

Lorsqu'on interprète l'accepteur A dans le monoïde $(\mathbb{N}, \cdot, 1)$, on s'intéresse au transformé de cet ensemble par la fonction $eval_M$, à savoir à l'ensemble des entiers :

$$\begin{array}{l}3 \\ 5 \cdot 2 \cdot 3 \\ 5 \cdot 2 \cdot 5 \cdot 2 \cdot 3 \\ 5 \cdot 2 \cdot 5 \cdot 2 \cdot 5 \cdot 2 \cdot 3 \\ \ldots \text{ etc.}\end{array}$$

L'accepteur A, ainsi interprété, représente l'ensemble des entiers de la forme $3 \cdot 10^n$.

Exemple 2. Supposons toujours que $M = \mathbb{N}$ et que α, β, γ soient les nombres 2, 5, 3, mais que l'on interprète l'accepteur A dans le monoïde additif $(\mathbb{N}, +, 0)$. Alors le sous-ensemble de $\mathbb{N}$ représenté par A est l'ensemble des entiers

$$\begin{array}{l} 3 \\ 5+2+3 \\ 5+2+5+2+3 \\ 5+2+5+2+5+2+3 \\ \cdots \end{array}$$

autrement dit l'ensemble des entiers naturels de la forme $7n + 3$.

Exemple 3. Lorsque le monoïde M, dont α, β, γ (figure 5.5(a)) sont des éléments par hypothèse, est le monoïde des relations dans un ensemble E, $M = \mathcal{R}(E)$, c'est-à-dire lorsque α, β, γ sont des relations dans un ensemble E, alors l'accepteur considéré, interprété dans le monoïde $\mathcal{R}(E)$, est appelé une *machine*.

5.2.3. Définition : machines et types de machines

Soient E un ensemble et Ω un ensemble de relations dans E, autrement dit $\Omega \subset \mathcal{R}(E)$. On appelle *machine de type* (E, Ω) tout accepteur fini M sur l'ensemble Ω.

Cette définition comporte en fait deux parties.

Premièrement, un *type de machine* est un couple (E, Ω) dans lequel E est un ensemble et Ω est un ensemble de relations dans E. Ces relations sont appelées les *opérations* de ce type de machine et l'on parle de l'ensemble Ω comme étant le *répertoire d'opérations* du type de machine (E, Ω). Ces relations sont d'ailleurs souvent des fonctions.

Deuxièmement, une *machine* de type (E, Ω) est un accepteur fini sur l'ensemble Ω.

Nous dirons qu'une machine M de type (E, Ω) *agit dans l'ensemble* E.

5.2.4. Exemple

Nous reprenons l'exemple de 5.2.1, en posant $E = \mathbb{N} \times \mathbb{N}$ et $\Omega = \{\alpha, \beta, \gamma\}$ avec

$$\begin{aligned} \alpha = Inc_1 &= \{(x, y) \mapsto (x+1, y) \mid (x, y) \in E\} \\ \beta = Dec_2 &= \{(x, y+1) \mapsto (x, y) \mid (x, y) \in E\} \\ \gamma = Nul_2 &= \{(x, 0) \mapsto (x, 0) \mid x \in \mathbb{N}\} = \mathrm{Id}_{(\mathbb{N} \times \{0\})}. \end{aligned}$$

L'accepteur (M, I, F) qui est représenté de deux manières dans la figure 5.6 est une machine de type (E, Ω).

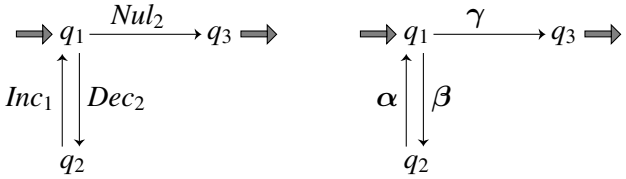


Figure 5.6

Le langage $L^M_{IF} = L^M_{q_1 q_3}$ sur Ω accepté par M est l'ensemble de séquences suivant :

$$(1) \qquad \left\{ \begin{array}{l} \langle\!\langle\, Nul_2 \,\rangle\!\rangle, \\ \langle\!\langle\, Dec_2, Inc_1, Nul_2 \,\rangle\!\rangle, \\ \langle\!\langle\, Dec_2, Inc_1, Dec_2, Inc_1, Nul_2 \,\rangle\!\rangle, \\ \langle\!\langle\, Dec_2, Inc_1, Dec_2, Inc_1, Dec_2, Inc_1, Nul_2 \,\rangle\!\rangle, \\ \cdots \end{array} \right\} = L^M_{IF.}$$

Il s'agit d'un ensemble de séquences de relations dans E. L'évaluation de chacune de ces séquences, dans le monoïde $\mathcal{R}(E)$, est le produit de composition des relations de la séquence. On obtient ainsi l'ensemble de relations :

$$(2)\qquad \left\{\begin{array}{l} Nul_2, \\ Dec_2Inc_1Nul_2, \\ Dec_2Inc_1Dec_2Inc_1Nul_2, \\ Dec_2Inc_1Dec_2Inc_1Dec_2Inc_1Nul_2, \\ \ldots \end{array}\right\} = \{eval_{\mathcal{R}(E)}(\sigma) \mid \sigma \in L^M_{IF}\}.$$

L'ensemble (2) est le transformé de l'ensemble (1) par la fonction d'évaluation $eval_{\mathcal{R}(E)} : \mathrm{W}(\mathcal{R}(E)) \to \mathcal{R}(E)$.

Lorsqu'il est dit dans la définition 5.2.3 qu'une machine est un accepteur *interprété*, cela signifie en l'occurrence que l'on s'intéresse plutôt à l'ensemble de relations (2) qu'à l'ensemble de séquences de relations (1).

Mais on va plus loin encore. On s'intéresse en fait à la réunion de toutes les relations de l'ensemble (2). La relation ainsi obtenue est la relation dite *définie* par la machine. On désigne cette relation par $|M|$. Comme l'ensemble (2) est l'ensemble des relations de la forme $(Dec_2Inc_1)^nNul_2$ ($n \in \mathbb{N}$), on a

$$(3)\qquad \begin{aligned} |M| &= \bigcup_{n \in \mathbb{N}} (Dec_2Inc_1)^nNul_2 \\ &= \Big(\bigcup_{n \in \mathbb{N}} (Dec_2Inc_1)^n\Big)Nul_2 = (Dec_2Inc_1)^*Nul_2. \end{aligned}$$

Cette notion fait l'objet de la définition générale suivante (5.2.5).

5.2.5. Relation définie par une machine

Soient (E, Ω) un type de machine et (M, I, F) une machine de type (E, Ω). Nous appelons *relation définie par M dans E* la relation

$$(1)\qquad |M| = \bigcup_{\sigma \in L^M_{IF}} eval_{\mathcal{R}(E)}(\sigma).$$

Pour que cette définition soit bien claire, nous nous référons encore à l'exemple précédent : pour chaque séquence $\sigma = \langle\!\langle\, \sigma[1], \ldots, \sigma[n] \,\rangle\!\rangle$ appartenant au langage L^M_{IF} (5.2.4(1)), on forme le produit de relations

$$eval_{\mathcal{R}(E)}(\sigma) = \sigma[1] \cdots \sigma[n] = \prod_{i=1}^{\mathrm{lg}(\sigma)} \sigma[k]$$

puis on prend la réunion des relations $eval_{\mathcal{R}(E)}(\sigma)$ ainsi formées (5.2.4(2)).

Par définition de $|M|$, on a

$$(2)\qquad \boxed{(a, b) \in |M| \;\Leftrightarrow\; \exists\sigma\Big(\sigma \in L^M_{IF} \text{ et } (a, b) \in \prod_{i=1}^{\mathrm{lg}(\sigma)} \sigma[k]\Big).}$$

Si l'on interprète M comme étant un programme dont l'espace des états de mémoire est l'ensemble E et dont les instructions simples appartiennent à l'ensemble Ω, on peut dire qu'une exécution de ce programme, à partir d'un état de mémoire initial a, consiste à exécuter une séquence d'instructions σ appartenant au langage sur Ω accepté par M.

5.2.6. Expression régulière de la relation définie par une machine

Nous avons vu (5.2.4(3)) que pour la machine M de la figure 5.6 la relation $|M|$ définie par M dans l'ensemble $E = \mathbb{N} \times \mathbb{N}$ peut s'exprimer au moyen des relations $\alpha = Inc_1$, $\beta = Dec_2$, $\gamma = Nul_2$ par

$$|M| = (\beta\alpha)^*\gamma. \tag{1}$$

Cette expression régulière frappe immédiatement comme étant exactement celle que l'on utiliserait pour exprimer le *langage* L_{IF}^M sur l'alphabet $\Omega = \{\alpha, \beta, \gamma\}$ accepté par M :

$$L_{IF}^M = (\beta\alpha)^*\gamma. \tag{2}$$

S'agit-il d'une coïncidence fortuite ou d'une propriété générale des machines? La réponse, comme nous allons le voir, est qu'il s'agit bien d'une propriété générale.

Remarquons d'abord qu'en toute rigueur l'une ou l'autre des égalités (1), (2) doit être incorrecte, sinon on peut en déduire $L_{IF}^M = |M|$, ce qui est absurde: le langage L_{IF}^M est un ensemble de séquences sur l'alphabet Ω et n'est pas la même chose que la relation $|M|$, laquelle est un sous-ensemble de $E \times E$.

L'erreur est vite trouvée. Elle est dans (2), où l'on écrit abusivement α, β, γ pour désigner les *langages élémentaires* $\{\langle\!\langle\, \alpha \,\rangle\!\rangle\}, \{\langle\!\langle\, \beta \,\rangle\!\rangle\}, \{\langle\!\langle\, \gamma \,\rangle\!\rangle\}$ sur l'alphabet $\Omega = \{\alpha, \beta, \gamma\}$, comme cela se fait généralement dans les expressions régulières utilisées pour représenter des langages réguliers (3.4.8). Les opérations de produit et de fermeture de Kleene dans (2) sont celles de l'algèbre de Kleene $\mathcal{L}(\Omega)$ des langages sur l'alphabet Ω et l'égalité rigoureusement correcte qui correspond à (2) est

$$L_{IF}^M = \Big(\{\langle\!\langle\, \beta \,\rangle\!\rangle\}\{\langle\!\langle\, \alpha \,\rangle\!\rangle\} \Big)^* \{\langle\!\langle\, \gamma \,\rangle\!\rangle\}. \tag{3}$$

Dans (1), par contre, il n'y a aucun abus de notation: α, β, γ désignent des relations dans E; les opérations de produit et de fermeture de Kleene sont celles de l'algèbre de Kleene $\mathcal{R}(E)$. L'expression $(\beta\alpha)^*\gamma$ désigne bien une relation dans E.

Il y a néanmoins une similitude remarquable entre les expressions (1) et (3). On passe de la première à la seconde en remplaçant α, β, γ respectivement par $\{\langle\!\langle\, \alpha \,\rangle\!\rangle\}$, $\{\langle\!\langle\, \beta \,\rangle\!\rangle\}$, $\{\langle\!\langle\, \gamma \,\rangle\!\rangle\}$ et vice-versa. On peut dire que le langage L_{IF}^M et la relation $|M|$ peuvent être « construits de la même manière » en partant d'une part des langages élémentaires $\{\langle\!\langle\, \alpha \,\rangle\!\rangle\}$, $\{\langle\!\langle\, \beta \,\rangle\!\rangle\}$, $\{\langle\!\langle\, \gamma \,\rangle\!\rangle\}$ et d'autre part des relations α, β, γ.

Que signifie exactement « construits de la même manière »? Cela signifie que l'on peut donner pour le langage L_{IF}^M et pour la relation $|M|$ des constructions génératrices semblables (figure 5.7). La première est une construction génératrice de base

$$\mathfrak{S} = \big\{\{\langle\!\langle\, \alpha \,\rangle\!\rangle\}, \{\langle\!\langle\, \beta \,\rangle\!\rangle\}, \{\langle\!\langle\, \gamma \,\rangle\!\rangle\}\big\}$$

dans l'algèbre de Kleene $\mathcal{L}(\Omega)$ des langages sur l'alphabet Ω. $\mathfrak{S}$ est l'ensemble des langages élémentaires sur Ω. La seconde est une construction génératrice de base $\Omega = \{\alpha, \beta, \gamma\}$ dans l'algèbre de Kleene $\mathcal{R}(E)$ des relations dans E.

L'adjectif « semblables », pour ces constructions génératrices, est encore vague. L'expression exacte de cette similitude est que, pour chaque $k \in [1 \,..\, 6]$ les propositions suivantes sont vraies: Si L_k est un langage élémentaire $\{\langle\!\langle\, \sigma \,\rangle\!\rangle\}$, alors R_k est la relation σ. Si $L_k = \varnothing$ (ce qui n'a pas lieu dans cet exemple), alors $R_k = \varnothing$. Si $L_k = L_i \cup L_j$ (ce qui n'a pas lieu), alors $R_k = R_i \cup R_j$. Si $L_k = L_i L_j$, alors $R_k = R_i R_j$. Si $L_k = L_i^*$, alors $R_k = R_i^*$.

Le théorème suivant montre que l'existence de constructions génératrices régulières semblables (dans ce sens) pour le langage L_{IF}^M et pour la relation $|M|$ définie par une machine (M, I, F), est une propriété générale des machines.

$L_1 = \{\langle\!\langle\, \alpha \,\rangle\!\rangle\}$			$R_1 = \alpha$		
$L_2 = \{\langle\!\langle\, \beta \,\rangle\!\rangle\}$			$R_2 = \beta$		
$L_3 = L_2 L_1$	$= \{\langle\!\langle\, \beta \,\rangle\!\rangle\}\{\langle\!\langle\, \alpha \,\rangle\!\rangle\}$		$R_3 = R_2 R_1$	$= \beta\alpha$	
$L_4 = L_3^*$	$= (\{\langle\!\langle\, \beta \,\rangle\!\rangle\}\{\langle\!\langle\, \alpha \,\rangle\!\rangle\})^*$		$R_4 = R_3^*$	$= (\beta\alpha)^*$	
$L_5 = \{\langle\!\langle\, \gamma \,\rangle\!\rangle\}$			$R_5 = \gamma$		
$L_6 = L_4 L_5$	$= (\{\langle\!\langle\, \beta \,\rangle\!\rangle\}\{\langle\!\langle\, \alpha \,\rangle\!\rangle\})^*\{\langle\!\langle\, \gamma \,\rangle\!\rangle\}.$		$R_6 = R_4 R_5$	$= (\beta\alpha)^*\gamma.$	

Figure 5.7

5.2.7. Théorème

Soient (E, Ω) un type de machine et (M, I, F) une machine de type (E, Ω). Soient encore $\mathfrak{S}$ l'ensemble des langages élémentaires sur Ω et

$$\langle\!\langle\, L_1, \ldots, L_n \,\rangle\!\rangle$$

une construction génératrice de base $\mathfrak{S}$ dans $\mathcal{L}(\Omega)$ telle que $L_n = L_{IF}^M$. Posons enfin, pour tout $k \in [1 \,..\, n]$:

$$R_k = \bigcup_{\sigma \in L_k} eval_{\mathcal{R}(E)}(\sigma).$$

Pour tout $k \in [1 \,..\, n]$ on a, quels que soient $\alpha \in \Omega$ et $i, j \in [1 \,..\, k-1]$:

$$\begin{aligned} L_k = \{\langle\!\langle\, \alpha \,\rangle\!\rangle\} &\Rightarrow R_k = \alpha; \\ L_k = \varnothing &\Rightarrow R_k = \varnothing; \\ L_k = L_i \cup L_j &\Rightarrow R_k = R_i \cup R_j; \\ L_k = L_i L_j &\Rightarrow R_k = R_i R_j; \\ L_k = (L_i)^* &\Rightarrow R_k = (R_i)^*. \end{aligned}$$

Par suite, la séquence $\langle\!\langle\, R_1, \ldots, R_n \,\rangle\!\rangle$ est une construction génératrice de base Ω dans $\mathcal{R}(E)$. On a en outre $R_n = |M|$.

Démonstration: voir [A].

5.2.8. Le théorème de Kleene pour les machines

Le théorème 5.2.7 montre que la relation $|M|$ définie par une machine M de type (E, Ω) est régulièrement engendrée par l'ensemble de relations Ω dans l'algèbre de Kleene $\mathcal{R}(E)$ (3.4.4), puisqu'il existe une construction génératrice $\langle\!\langle\, R_1, \ldots, R_n \,\rangle\!\rangle$ de base Ω dans $\mathcal{R}(E)$ telle que $R_n = |M|$. On a donc

$$|M| \in \mathsf{Reg}(\Omega),$$

suivant la notation introduite au §3.4.4. L'énoncé de ce fait constitue la première partie du théorème suivant, que nous appelons le « théorème de Kleene pour les machines ».

Théorème. Soient (E, Ω) un type de machine. Pour toute machine M de type (E, Ω), on a $|M| \in \mathsf{Reg}(\Omega)$. Réciproquement, pour toute relation R dans E qui est régulièrement engendrée par Ω, c'est-à-dire pour toute relation $R \in \mathsf{Reg}(\Omega)$, il existe une machine M de type (E, Ω) telle que $R = |M|$.

Démonstration: voir [A].

5.2.9. Complément sur l'opérateur Reg

Les théorèmes suivants sont un complément à ceux du §3.4.5.

Théorèmes. Soient $\mathcal{K}$ une algèbre de Kleene, $\mathfrak{S} \subset \mathcal{K}$ et $\mathfrak{S}' \subset \mathcal{K}$. On a

(1) $\mathfrak{S} \subset \mathsf{Reg}(\mathfrak{S})$.

(2) $\mathfrak{S} \subset \mathfrak{S}' \Rightarrow \mathsf{Reg}(\mathfrak{S}) \subset \mathsf{Reg}(\mathfrak{S}')$.

(3) $\mathsf{Reg}(\mathsf{Reg}(\mathfrak{S})) = \mathsf{Reg}(\mathfrak{S})$.

Ces formules sont intuitivement évidentes lorsqu'on interprète $\mathsf{Reg}(\mathfrak{S})$ comme étant l'ensemble des éléments de $\mathcal{K}$ qui peuvent être construits à partir d'éléments de $\mathfrak{S} \cup \{\emptyset\}$ en effectuant un nombre fini de réunions, produits, fermetures de Kleene. Une démonstration précise est donnée dans [A].

5.2.10. Machines et types de machines équivalents

Ce qui est important, dans une machine M agissant dans un ensemble E, c'est la relation $|M|$ qu'elle définit dans E. C'est pourquoi deux machines M, M' agissant dans le même ensemble E, donc de types respectifs (E, Ω), (E, Ω'), sont dites *équivalentes* si $|M| = |M'|$.

Dans la même optique, ce qui est important dans un type de machine (E, Ω), ce sont les relations dans E que l'on peut définir au moyen de machines de ce type, c'est-à-dire de machines utilisant le répertoire d'instructions Ω. Deux types (E, Ω), (E, Ω') de machines agissant dans le même ensemble E sont dits *équivalents* si, pour toute machine de l'un de ces types il existe une machine équivalente de l'autre type. D'après le théorème de Kleene (5.2.8), cela revient à dire que toute relation dans E qui est régulièrement engendrée par Ω est régulièrement engendrée par Ω' et réciproquement, autrement dit que les ensembles de relations $\mathsf{Reg}(\Omega)$ et $\mathsf{Reg}(\Omega')$ sont égaux.

Exemple 1. Les machines M' et M'' représentées dans la figure 5.8, agissant dans l'ensemble $E = \mathbb{N} \times \mathbb{N}$, sont équivalentes à la machine M de la figure 5.6, car on a évidemment

$$|M| = |M'| = |M''| = (Dec_2 Inc_1)^* Nul_2.$$

En posant comme précédemment $\Omega = \{Dec_2, Inc_1, Nul_2\}$ (5.2.4), on peut dire que M est de type (E, Ω) mais que M' et M'' ne sont pas de ce type puisque ces deux accepteurs possèdent des transitions (q_i, σ, q_j) telles que $\sigma \notin \Omega$. Par contre, on peut dire que M' et M'' sont de type $(E, \mathsf{Reg}(\Omega))$ puisque, pour chacune de leurs transitions (q_i, σ, q_j), la relation σ appartient à $\mathsf{Reg}(\Omega)$.

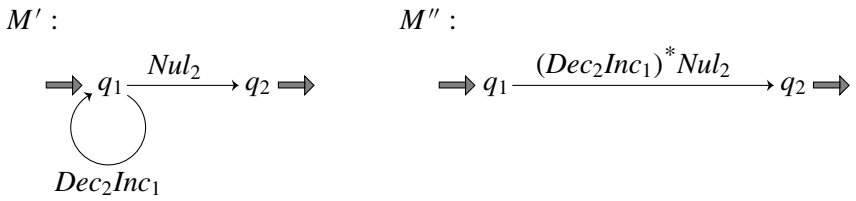


Figure 5.8

En fait, les types de machines (E, Ω) et $(E, \mathsf{Reg}(\Omega))$ sont équivalents, et ceci est valable pour n'importe quel type de machine (E, Ω), en vertu de l'égalité

$$\mathsf{Reg}(\Omega) = \mathsf{Reg}(\mathsf{Reg}(\Omega)),$$

qui est vraie d'après 5.2.9(3). Ceci est une équivalence de type fréquemment utilisée, en ce sens que l'on remplace souvent une machine M de type (E, Ω) par une machine M' de type $(E, \mathsf{Reg}(\Omega))$ telle que $|M'| = |M|$. En particulier, on peut toujours choisir pour M' une

machine ayant seulement deux états et une transition, comme la machine M'' de la figure 5.8, à savoir la transition (q_1, σ, q_2) où $\sigma = |M|$.

Exemple 2. Une autre équivalence de types de machine fréquemment utilisée est celle d'un type (E, Ω) quelconque et du type

$$(E,\ \Omega \cup \{\mathrm{Id}_E\}).$$

Vérifions l'équivalence de ces deux types en posant $\Omega' = \Omega \cup \{\mathrm{Id}_E\}$ et en montrant que $\mathrm{Reg}(\Omega) = \mathrm{Reg}(\Omega')$. Comme $\Omega \subset \Omega'$, on a $\mathrm{Reg}(\Omega) \subset \mathrm{Reg}(\Omega')$ d'après 5.2.9(2). Démontrons l'inclusion inverse. Comme $\Omega \subset \mathrm{Reg}(\Omega)$ (5.2.9(1)) et $\{\mathrm{Id}_E\} \subset \mathrm{Reg}(\Omega)$ (3.4.5(2)), on a $\Omega' \subset \mathrm{Reg}(\Omega)$, donc $\mathrm{Reg}(\Omega') \subset \mathrm{Reg}(\mathrm{Reg}(\Omega)) = \mathrm{Reg}(\Omega)$ d'après 5.2.9(2) et 5.2.9(3).

En vertu de cette équivalence, l'instruction neutre Id_E, souvent notée $\mathtt{NOP}_E$ ou simplement `NOP`, est fréquemment utilisée dans une machine, en plus des instructions d'un répertoire Ω donné.

5.3 Assertations de machines

5.3.1. Exemple

Soient encore une fois $E = \mathbb{N} \times \mathbb{N}$, $\Omega = \{Inc_1, Dec_2, Nul_2\}$ et (M, I, F) la machine de la figure 5.6, de type (E, Ω). Cette machine est représentée encore dans la figure 5.9, où elle est munie d'une *assertation*. Cela signifie qu'à chaque état $p \in Q$ est associé un ensemble $A_p \subset E$. On suppose que m et n sont deux entiers naturels fixés. Les ensembles associés aux états q_1, q_2, q_3 sont respectivement

$$\begin{aligned} A_{q_1} &= \{(x, y) \in E \mid x + y = m + n\};\\ A_{q_2} &= \{(x, y) \in E \mid x + y + 1 = m + n\};\\ A_{q_3} &= \{(x, y) \in E \mid x + y = m + n \text{ et } y = 0\}. \end{aligned}$$

En outre, un ensemble initial $A_I \subset E$ et un ensemble final $A_F \subset E$ sont donnés, à savoir

$$\begin{aligned} A_I &= \{(x, y) \in E \mid (x, y) = (m, n)\};\\ A_F &= \{(x, y) \in E \mid x = m + n\}. \end{aligned}$$

Il est clair que ces ensembles ne sont pas tous notés de la manière la plus simple possible. On pourrait surtout écrire plus simplement

$$A_I = \{(m, n)\}, \qquad A_{q_3} = \{(m + n, 0)\}.$$

La notation utilisée pour les ensembles A_{q_i}, A_I, A_F dans la figure 5.9 est uniforme, à savoir que chacun d'eux est représenté par une expression de la forme

$$\{(x, y) \in E \mid P(x, y)\},$$

désignant l'ensemble des couples $(x, y) \in E$ qui vérifient une certaine proposition, ou assertion, $P(x, y)$. Seule la proposition $P(x, y)$ change d'un de ces ensembles à l'autre. Cela suggère immédiatement la notation abrégée de la figure 5.10, où chaque ensemble $\{(x, y) \mid P(x, y)\}$ de l'assertation est désigné simplement par $P(x, y)$. On parle de *l'assertion* $P(x, y)$. Mais ceci n'est qu'une simplification de la notation. Nous nous référons à la figure 5.9 pour la suite de la discussion.

Le système d'ensembles A_{q_i}, A_I, A_F n'est pas quelconque. Il possède certaines propriétés, *en vertu desquelles* il constitue — par définition — une *assertation* de la machine

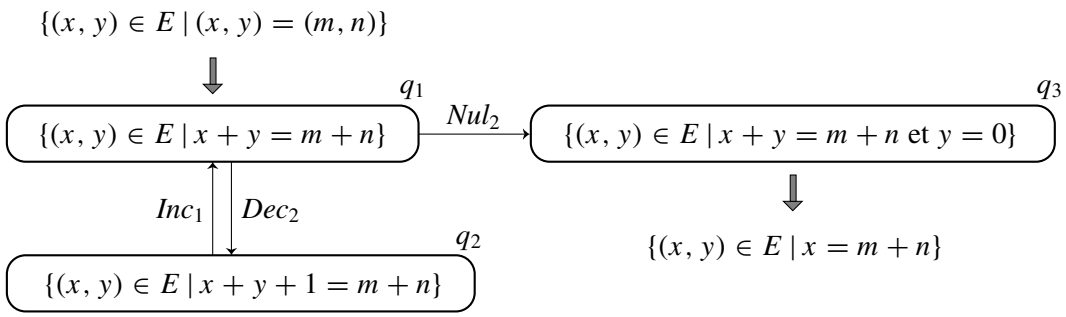


Figure 5.9

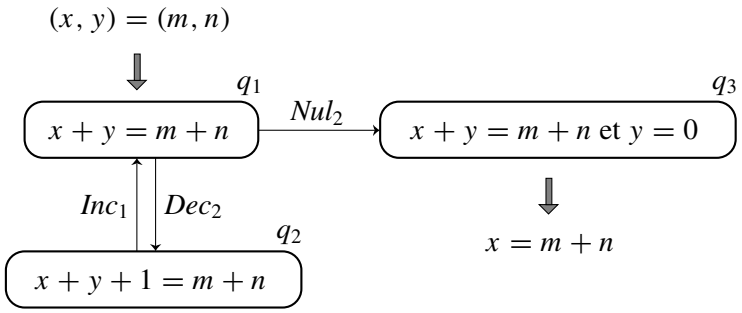


Figure 5.10

considérée. Ces propriétés sont les suivantes. Premièrement, on a

$$A_I \subset A_{q_1} \text{ et } A_{q_3} \subset A_F. \tag{1}$$

Ces inclusions sont évidentes. L'unique élément de A_I est le couple $(x, y) = (m, n)$ et pour ce couple on a évidemment $x + y = m + n$, donc il appartient à A_{q_1}. L'ensemble A_{q_3} possède un élément et un seul, à savoir le couple $(m + n, 0)$. Il est donc contenu dans l'ensemble A_F qui est l'ensemble des couples dont la première projection est égale à $m + n$.

Dans cet exemple, l'accepteur M possède un seul état initial (q_1) et un seul état final (q_3). S'il y avait plusieurs états initiaux q_i (resp. plusieurs états finaux q_j), l'inclusion $A_I \subset A_{q_i}$ (resp $A_{q_j} \subset A_F$) devrait être vérifiée pour chaque état $q_i \in I$ (resp. pour chaque état $q_j \in F$).

Deuxièmement, on a les inclusions suivantes :

$$Dec_2\langle A_{q_1}\rangle \subset A_{q_2}, \quad Inc_1\langle A_{q_2}\rangle \subset A_{q_1}, \quad Nul_2\langle A_{q_1}\rangle \subset A_{q_3}. \tag{2}$$

La règle générale est que pour chaque transition (q_i, α, q_j) de la machine, le transformé de l'ensemble A_{q_i} par la relation α est inclus dans A_{q_j}. Les inclusions (2) se démontrent à partir de la donnée des ensembles A_{q_i} et de la définition des relations Dec_2, Inc_1, Nul_2 (5.2.4). Cette démonstration est donnée plus bas pour la première des trois inclusions.

La *propriété principale* de l'assertion, qui *découle* de (1) et (2), est la suivante :

$$|M|\langle A_I\rangle \subset A_F. \tag{3}$$

Le transformé de l'ensemble initial de l'assertion par la relation $|M|$ définie par la machine est inclus dans l'ensemble final de l'assertion. En d'autres termes, si (x, y) et (x', y') sont l'état de mémoire initial et l'état de mémoire final d'une exécution du « programme » M et si $(x, y) \in A_I$, c'est-à-dire si $(x, y) = (m, n)$, alors $(x', y') \in A_F$, c'est-à-dire $x' = m + n$.

Nous ne démontrerons pas l'inclusion (3) pour cet exemple particulier, mais dans le cas général d'une machine et d'une assertion quelconques, après avoir donné la définition

générale de ce concept (5.3.2).

Démonstration des inclusions (2). Les inclusions de cette forme se démontrent en appliquant la formule générale 1.3.23(4) sur le transformé d'un ensemble A par une relation R. Nous démontrons ainsi la première des inclusions (2). D'après la formule mentionnée, il s'agit de montrer que pour deux éléments

$$z = (x, y) \text{ et } z' = (x', y')$$

quelconques de $E = \mathbb{N} \times \mathbb{N}$, on a

(4) $$(z \in A_{q_1} \text{ et } (z, z') \in Dec_2) \Rightarrow z' \in A_{q_2}.$$

Par définition de A_{q_1}, Dec_2 (5.2.4) et A_{q_2}, on a

$$\begin{aligned} z \in A_{q_1} &\Leftrightarrow x + y = m + n; \\ (z, z') \in Dec_2 &\Leftrightarrow (y \neq 0 \text{ et } z' = (x, y - 1)); \\ z' \in A_{q_2} &\Leftrightarrow x' + y' + 1 = m + n. \end{aligned}$$

Supposons que $z \in A_{q_1}$ et $(z, z') \in Dec_2$, c'est-à-dire que $x + y = m + n$ et $y \neq 0$ et $z' = (x, y - 1)$. Alors $x' = x$ et $y' = y - 1$, donc $x' + y' + 1 = m + n$. D'où $z' \in A_{q_2}$.

5.3.2. Définition : assertion d'une machine

Soient (E, Ω) un type de machine et (M, I, F) une machine de type (E, Ω). On appelle *assertation* de M la donnée, pour chaque état $p \in Q^M$, d'un ensemble $A_p \subset E$, et en outre, celle d'un ensemble $A_I \subset E$ et d'un ensemble $A_F \subset E$, ces données satisfaisant aux conditions suivantes :

(1) $\forall p \in I : \quad A_I \subset A_p.$

(2) $\forall q \in F : \quad A_q \subset A_F.$

(3) Pour chaque transition (p, α, q) de M : $\quad \alpha\langle A_p \rangle \subset A_q.$

Les ensembles A_I et A_F sont appelés respectivement *l'ensemble initial* et *l'ensemble final* de l'assertation. La propriété suivante d'une assertation de M est une conséquence de la propriété (3) :

(4) Pour toute séquence $\sigma = \langle\!\langle\, \sigma[1], \ldots, \sigma[n] \,\rangle\!\rangle$ sur l'alphabet Ω, si p et q sont deux états de M tels que σ relie p à q dans M, alors on a $(\sigma[1] \cdots \sigma[n])\langle A_p \rangle \subset A_q$, autrement dit

$$\big(eval_{\mathcal{R}(E)}(\sigma)\big)\langle A_p \rangle \subset A_q.$$

Démonstration : voir [A].

5.3.3. Propriété principale d'une assertation

Soient (E, Ω) un type de machine, (M, I, F) une machine de type (E, Ω) et soient A_I, A_F l'ensemble initial et l'ensemble final d'une assertation de M. On a

$$|M|\langle A_I \rangle \subset A_F.$$

Démonstration. D'après la formule 1.3.23(4), il suffit de montrer que

$$\big(a \in A_I \text{ et } (a, b) \in |M|\big) \Rightarrow b \in A_F.$$

Supposons donc que $a \in A_I$ et $(a, b) \in |M|$. Nous devons en déduire $b \in A_F$. Par définition de $|M|$ (5.2.5), et plus précisément d'après 5.2.5(2), il existe une séquence $\sigma \in L^M_{IF}$ telle que

(1) $$(a, b) \in eval_{\mathcal{R}(E)}(\sigma).$$

Une telle séquence σ relie un état $p \in I$ à un état $q \in F$ dans M et, par définition d'une assertation, on a $A_I \subset A_p$ et $A_q \subset A_F$. Comme $a \in A_I$ par hypothèse, on a donc

$$a \in A_p. \tag{2}$$

Comme σ relie p à q, on a, d'après 5.3.2(4),

$$\big(eval_{\mathcal{R}(E)}(\sigma)\big)\langle A_p\rangle \subset A_q. \tag{3}$$

De (1), (2) et (3), on déduit $b \in A_q$ (1.3.23(4)). Comme $A_q \subset A_F$, il vient $b \in A_F$.

5.3.4. Relations d'affectation

Au §5.1.2 nous avons considéré le programme

```
Var x, y : ℕ;
while y ≠ 0 do
    y := y − 1;
    x := x + 1
od
```

et au §5.1.3, nous avons associé aux instructions $y := y - 1$ et $x := x + 1$ de ce programme les relations

$$Dec_2 = \{(x, y) \mapsto (x, y - 1) \mid (x, y) \in \mathbb{N} \times \mathbb{N} \text{ et } y \neq 0\};$$
$$Inc_1 = \{(x, y) \mapsto (x + 1, y) \mid (x, y) \in \mathbb{N} \times \mathbb{N}\}$$

dans l'ensemble $\mathbb{N} \times \mathbb{N}$. Nous dirons naturellement que ces relations dans $\mathbb{N} \times \mathbb{N}$ sont des *relations d'affectation*.

On peut convenir, si l'on veut, que les expressions $y := y - 1$ et $x := x + 1$ elles-mêmes désignent ces relations. Cependant, en tant que notations d'une relation, ces expressions sont incomplètes. Si l'on changeait par exemple l'ordre des variables dans la déclaration initiale du programme, en écrivant Var $y, x : \mathbb{N}$, alors les relations correspondant aux instructions $y := y - 1$, $x := x + 1$ seraient

$$Dec_1 = \{(y, x) \mapsto (y - 1, x) \mid (y, x) \in \mathbb{N} \times \mathbb{N} \text{ et } y \neq 0\};$$
$$Inc_2 = \{(y, x) \mapsto (y, x + 1) \mid (y, x) \in \mathbb{N} \times \mathbb{N}\}.$$

Par ailleurs, les instructions $y := y-1, x := x+1$ pourraient figurer dans un programme ayant d'autres variables que x et y, commençant par exemple par la déclaration Var $x, y, z : \mathbb{N}$. Dans ce cas, l'espace des états de mémoire serait l'ensemble $\mathbb{N}^3 = \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ et la relation associée à $y := y - 1$ serait l'ensemble

$$\{(x, y, z) \mapsto (x, y - 1, z) \mid (x, y, z) \in \mathbb{N}^3 \text{ et } y \neq 0\}.$$

La relation associée à une instruction d'affectation dépend donc du programme dans lequel elle se trouve, et plus exactement de la déclaration de variables par laquelle le programme commence. C'est pourquoi nous utiliserons la notation

$$\left[\!\!\left[\begin{array}{l} y := y - 1 \\ (x, y) \in \mathbb{N} \times \mathbb{N} \end{array} \right]\!\!\right]$$

comme notation complète de la relation correspondant à l'instruction $y := y - 1$ dans le cadre de la déclaration de variables $x, y : \mathbb{N}$.

La forme générale d'une déclaration de n variables est

$$\text{Var } x_1 : E_1;\ \ldots\ ;\ \text{Var } x_n : E_n;$$

où $x_1, \ldots, x_n$ sont des variables distinctes et $E_1, \ldots, E_n$ sont des ensembles. L'espace des états de mémoire correspondant est le produit cartésien $E = E_1 \times \cdots \times E_n$, ensemble des n-uples de valeurs $(x_1, \ldots, x_n)$ tels que $x_i \in E_i$ pour $i = 1, \ldots, n$. Dans le cadre d'une telle déclaration de variables, une instruction d'affectation est de la forme générale

$$x_i := \mathbf{T},$$

où x_i est l'une des variables $x_1, \ldots, x_n$ et $\mathbf{T}$ est un terme. En général, ce terme peut dépendre de $x_1, \ldots, x_n$, à savoir que certaines de ces variables, voir toutes, peuvent figurer librement dans $\mathbf{T}$. C'est le cas par exemple dans l'instruction $x_i := x_i + x_j$, où $\mathbf{T}$ est le terme $x_i + x_j$.

Définition. La relation d'affectation

$$R = \left[\!\!\left[\begin{array}{l} x_i := \mathbf{T} \\ (x_1, \ldots, x_n) \in E_1 \times \cdots \times E_n \end{array} \right]\!\!\right]$$

dans l'ensemble d'états de mémoire $E = E_1 \times \cdots \times E_n$ est l'ensemble des couples d'états de mémoire de la forme $(x_1, \ldots, x_i, \ldots, x_n) \mapsto (x_1, \ldots, \mathbf{T}, \ldots, x_n)$. Formellement, en désignant par $(\mathbf{T}|x_i)(x_1, \ldots, x_n)$ le terme $(x_1, \ldots, \mathbf{T}, \ldots, x_n)$, obtenu en remplaçant x_i par $\mathbf{T}$ dans le terme $(x_1, \ldots, x_n)$, nous avons par définition

$$(1) \qquad \left[\!\!\left[\begin{array}{l} x_i := \mathbf{T} \\ (x_1, \ldots, x_n) \in E \end{array} \right]\!\!\right] = \left\{ (x_1, \ldots, x_n) \mapsto (\mathbf{T}|x_i)(x_1, \ldots, x_n) \;\middle|\; (x_1, \ldots, x_n) \in E \text{ et } \mathbf{T} \in E_i \right\}.$$

Exemple. Si $E = E_1 \times E_2 = \mathbb{N} \times \mathbb{N}$,

$$\left[\!\!\left[\begin{array}{l} y := y - 1 \\ (x, y) \in E \end{array} \right]\!\!\right] = \left\{ (x, y) \mapsto (x, y - 1) \;\middle|\; (x, y) \in E \text{ et } y - 1 \in \mathbb{N} \right\}.$$

Le terme $\mathbf{T}$ est dans ce cas le terme $y - 1$. La condition $(x, y) \in E$ et $y - 1 \in \mathbb{N}$, à droite de la barre verticale, correspond à la condition $(x_1, \ldots, x_n) \in E$ et $\mathbf{T} \in E_i$ de (1), à savoir, dans ce cas, $(x, y) \in E$ et $y - 1 \in E_2 = \mathbb{N}$. Elle équivaut à $(x, y) \in \mathbb{N} \times \mathbb{N}$ et $y \neq 0$. Cet exemple montre bien la nécessité de la condition $\mathbf{T} \in E_i$ à droite de la barre verticale dans la définition générale (1). La valeur que prend le terme $\mathbf{T}$, dans un état de mémoire $(x_1, \ldots, x_n)$ quelconque, n'appartient pas nécessairement à l'ensemble E_i. Par exemple, la valeur du terme $y - 1$, pour un état de mémoire $(x, y) \in \mathbb{N} \times \mathbb{N}$, n'appartient pas nécessairement à $\mathbb{N}$. Si la condition $\mathbf{T} \in E_i$ n'est pas satisfaite par un état de mémoire $(x_1, \ldots, x_n)$, l'opération d'affectation n'est pas exécutable. Cette condition est donc une restriction sur les états de mémoire à partir desquels l'opération d'affectation est exécutable.

Théorème. Soient $E_1, \ldots, E_n$ des ensembles, $E = E_1 \times \cdots \times E_n$, et soient

$$A = \{(x_1, \ldots, x_n) \in E \mid P(x_1, \ldots, x_n)\}$$
$$B = \{(x_1, \ldots, x_n) \in E \mid Q(x_1, \ldots, x_n)\}$$

deux sous-ensembles de E, définis au moyen de propositions $P(x_1, \ldots, x_n)$, $Q(x_1, \ldots, x_n)$, qui sont des conditions caractérisant les états de mémoire $(x_1, \ldots, x_n)$ appartenant à ces ensembles. Soit d'autre part

$$R = \left[\!\!\left[\begin{array}{l} x_i := \mathbf{T} \\ (x_1, \ldots, x_n) \in E \end{array} \right]\!\!\right]$$

une relation d'affectation (1) dans E. On a

$$(2)\qquad R\langle A\rangle \subset B \Leftrightarrow \left[\begin{array}{l}\text{pour tout } (x_1,\ldots,x_n)\in E : \\ \big(P(x_1,\ldots,x_n) \text{ et } \mathbf{T}\in E_i\big) \Rightarrow Q(x_1,\ldots,\mathbf{T},\ldots,x_n)\end{array}\right],$$

où la proposition $Q(x_1,\ldots,\mathbf{T},\ldots,x_n)$ est la proposition $(\mathbf{T}|x_i)Q$, obtenue en remplaçant toutes les occurrences libres de x_i dans Q par le terme $\mathbf{T}$ (que l'on suppose librement substituable à x_i dans Q).

Démonstration. Pour un état de mémoire $(x_1,\ldots,x_n)\in E$, la conjonction des propositions $P(x_1,\ldots,x_n)$ et $\mathbf{T}\in E_i$ équivaut, par définition de A et de R (1), à

$$(3)\qquad (x_1,\ldots,x_n)\in A \text{ et } (x_1,\ldots,x_n)\mapsto(\mathbf{T}|x_i)(x_1,\ldots,x_n)\in R.$$

D'autre part, la proposition $Q(x_1,\ldots,\mathbf{T},\ldots,x_n)$ signifie, par définition de B, que l'état de mémoire $(x_1,\ldots,\mathbf{T},\ldots,x_n)$ appartient à B, ce qui s'écrit :

$$(4)\qquad (\mathbf{T}|x_i)(x_1,\ldots,x_n)\in B.$$

L'implication $(P(x_1,\ldots,x_n)$ et $\mathbf{T}\in E_i)\Rightarrow Q(x_1,\ldots,\mathbf{T},\ldots,x_n)$, dans (2), équivaut donc à (3) $\Rightarrow$ (4). Dire qu'elle est vraie pour tout $(x_1,\ldots,x_n)\in E$ équivaut à $R\langle A\rangle\subset B$ (1.3.23(4)).

5.3.5. Exemple : machine de multiplication

La figure 5.11, dans laquelle la déclaration de variables $u, x, y, z : \mathbb{N}$ est sous-entendue, représente une machine (M, I, F) agissant dans l'ensemble $E = E_1\times\cdots\times E_4 = \mathbb{N}\times\mathbb{N}\times\mathbb{N}\times\mathbb{N} = \mathbb{N}^4$. On peut considérer cette machine comme correspondant au programme :

```
Var u, x, y, z : ℕ;
u := x;
z := 0;
while u ≠ 0 do
    z := z + y;
    u := u − 1
od.
```

L'expression $u := x$ représente la relation d'affectation

$$\left[\!\!\left[\begin{array}{l}u:=x\\(u,x,y,z)\in E\end{array}\right]\!\!\right] = \big\{(u,x,y,z)\mapsto(x,x,y,z)\;\big|\;(u,x,y,z)\in E\big\}.$$

La condition $\mathbf{T}\in E_i$ qui figure dans la définition générale 5.3.4(1) serait ici $x\in E_1$, c'est-à-dire $x\in\mathbb{N}$. Elle est superflue parce qu'elle est impliquée par $(u,x,y,z)\in E$. De même, les expressions $z:=0$, $z:=z+y$, $u:=u-1$ représentent les relations

$$\left[\!\!\left[\begin{array}{l}z:=0\\(u,x,y,z)\in E\end{array}\right]\!\!\right] = \big\{(u,x,y,z)\mapsto(u,x,y,0)\;\big|\;(u,x,y,z)\in E\big\}$$

$$\left[\!\!\left[\begin{array}{l}z:=z+y\\(u,x,y,z)\in E\end{array}\right]\!\!\right] = \big\{(u,x,y,z)\mapsto(u,x,y,z+y)\;\big|\;(u,x,y,z)\in E\big\}$$

$$\left[\!\!\left[\begin{array}{l}u:=u-1\\(u,x,y,z)\in E\end{array}\right]\!\!\right] = \big\{(u,x,y,z)\mapsto(u-1,x,y,z)\;\big|\;(u,x,y,z)\in E \text{ et } u\neq 0\big\}.$$

La condition $u\neq 0$, dans cette dernière, correspond à la condition $\mathbf{T}\in E_i$ de 5.3.4(1), qui s'exprime ici $u-1\in E_1$, c'est-à-dire $u-1\in\mathbb{N}$. Nous désignerons ces quatre relations

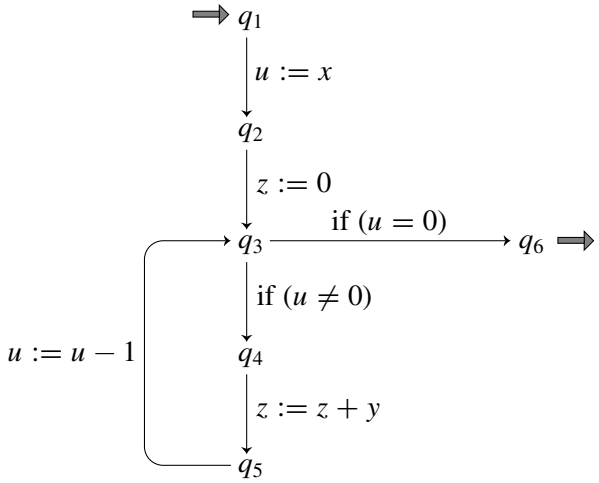


Figure 5.11
Machine de multiplication.
Variables $u, x, y, z : \mathbb{N}$.

d'affectation dans $\mathbb{N}^4$ par les notations abrégées

$$[\![u := x]\!], \quad [\![z := 0]\!], \quad [\![z := z + y]\!], \quad [\![u := u - 1]\!].$$

Les expressions if $(u = 0)$, if $(u \neq 0)$ représentent les relations de test

$$\left[\!\!\left[\begin{matrix}\text{if } (u = 0)\\ (u, x, y, z) \in E\end{matrix}\right]\!\!\right] = \big\{(u, x, y, z) \mapsto (u, x, y, z) \;\big|\; (u, x, y, z) \in E \text{ et } u = 0\big\}$$
$$= \mathrm{Id}_X \quad \text{où} \quad X = \{(u, x, y, z) \in E \mid u = 0\}$$
$$\left[\!\!\left[\begin{matrix}\text{if } (u \neq 0)\\ (u, x, y, z) \in E\end{matrix}\right]\!\!\right] = \big\{(u, x, y, z) \mapsto (u, x, y, z) \;\big|\; (u, x, y, z) \in E \text{ et } u \neq 0\big\}$$
$$= \mathrm{Id}_{\overline{X}} \quad \text{où} \quad \overline{X} = \{(u, x, y, z) \in E \mid u \neq 0\}.$$

Une assertation de cette machine M est donnée dans la figure 5.12. Les ensembles A_I, A_F, A_{q_i} $(i = 1, \ldots, 5)$ sont tous représentés seulement par une assertion (proposition) $P(u, x, y, z)$ et c'est chaque fois l'ensemble

$$A = \{(u, z, x, y) \in E \mid P(u, z, x, y)\}$$

qui est représenté ainsi (cf. figures 5.9 et 5.10). En particulier le symbole *true* représente une proposition vraie, par exemple la proposition $u = u$, de sorte que les ensembles A_I et A_{q_1} sont égaux à E puisqu'on a trivialement $\{(u, z, x, y) \in E \mid u = u\} = E$.

Il faut vérifier bien sûr que cette assertation de M en est bien une, à savoir que les conditions (1), (2), (3) de 5.3.2 sont satisfaites. Les deux premières le sont évidemment, puisque

$$A_I = A_{q_1} = E;$$
$$A_{q_6} = A_F = \{(u, x, y, z) \in E \mid z = xy\}.$$

Pour la troisième condition, 5.3.2(3), nous avons deux genres de transitions à considérer: celles qui comportent une instruction de test et celles qui comportent une instruction d'affectation. Il s'agit de vérifier par exemple que les inclusions

$$(1) \qquad [\![\text{if } (u = 0)]\!]\langle A_{q_3}\rangle \subset A_{q_6}, \qquad [\![u := u - 1]\!]\langle A_{q_5}\rangle \subset A_{q_3}$$

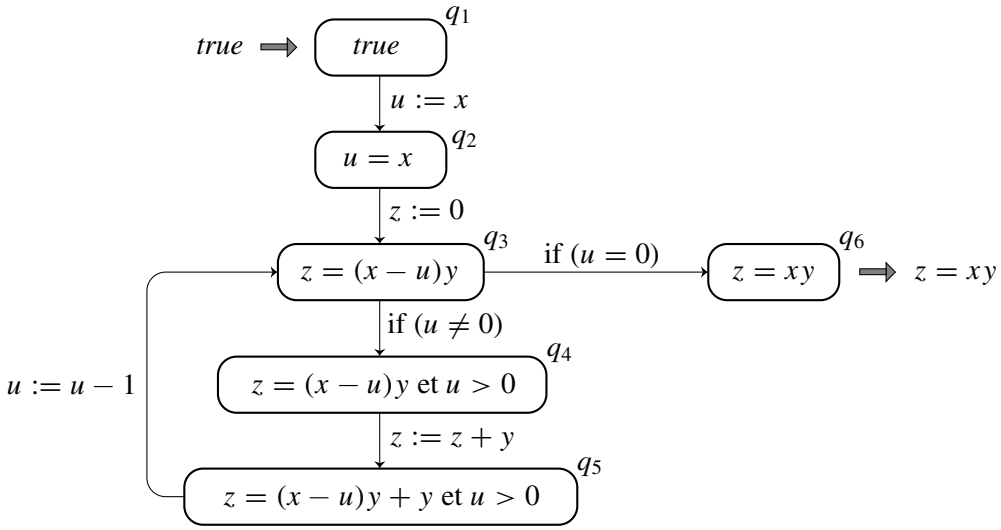


Figure 5.12

sont vraies pour les ensembles

$$A_{q_3} = \{(u, x, y, z) \in E \mid z = (x - u)y\};$$
$$A_{q_6} = \{(u, x, y, z) \in E \mid z = xy\};$$
$$A_{q_5} = \{(u, x, y, z) \in E \mid z = (x - u)y + y \text{ et } u > 0\}.$$

La relation $[\![\text{if } (u = 0)]\!]$ est la relation identique Id_X de l'ensemble

$$X = \{(u, x, y, z) \in E \mid u = 0\}.$$

D'après le théorème 6 de 1.3.23, l'inclusion $[\![\text{if } (u = 0)]\!]\langle A_{q_3}\rangle \subset A_{q_6}$ équivaut à

$$A_{q_3} \cap X \subset A_{q_6}.$$

Il est facile de montrer que cette inclusion est vraie. Soit en effet $a = (u, x, y, z)$ un état de mémoire quelconque ($a \in E$). Si $a \in A_{q_3} \cap X$, c'est-à-dire si l'on a $z = (x - u)y$ et $u = 0$, alors $z = xy$, donc $a \in A_{q_6}$. En résumé, la première des inclusions (1) est vérifiée simplement parce que, pour tout $(u, x, y, z) \in E$:

$$\big(z = (x - u)y \text{ et } u = 0\big) \Rightarrow z = xy.$$

Toutes les transitions de test se vérifient de la même manière. Par exemple, on a

$$[\![\text{if } (u \neq 0)]\!]\langle A_{q_3}\rangle \subset A_{q_4}$$

parce que $\big(z = (x - u)y \text{ et } u \neq 0\big) \Rightarrow \big(z = (x - u)y \text{ et } u > 0\big)$.

Pour les transitions comportant une affectation, nous appliquons le théorème 5.3.4(2). En vertu de ce théorème, on a par exemple

$[\![u := x]\!]\langle A_{q_1}\rangle \subset A_{q_2}$
$\Leftrightarrow$
pour tout $(u, x, y, z) \in \mathbb{N}^4$: (*true* et $x \in \mathbb{N}$) $\Rightarrow (x|u)(u = x)$.

Le second membre de cette équivalence est vrai d'après le principe de logique élémentaire « tout implique le vrai », puisque la proposition $(x|u)(u = x)$ est la proposition vraie $x = x$. Donc $[\![u := x]\!]\langle A_{q_1}\rangle \subset A_{q_2}$ est vraie. On a de même

$[\![z := 0]\!]\langle A_{q_2}\rangle \subset A_{q_3}$
$\Leftrightarrow$
pour tout $(u, x, y, z) \in \mathbb{N}^4$: $(u = x \text{ et } 0 \in \mathbb{N}) \Rightarrow (0|z)\big(z = (x-u)y\big)$.

L'implication $(u = x \text{ et } 0 \in \mathbb{N}) \Rightarrow (0|z)\big(z = (x-u)y\big)$ équivaut à

$$u = x \;\Rightarrow\; 0 = (x-u)y.$$

Elle est évidemment vraie pour tout $(u, x, y, z) \in \mathbb{N}^4$. Donc $[\![z := 0]\!]\langle A_{q_2}\rangle \subset A_{q_3}$ est vraie. Finalement, on a

$[\![u := u-1]\!]\langle A_5\rangle \subset A_3$
$\Leftrightarrow$
pour tout $(u, x, y, z) \in \mathbb{N}^4$:
$$\underbrace{\big(z = (x-u)y + y \text{ et } u > 0 \text{ et } u - 1 \in \mathbb{N}\big) \;\Rightarrow\; (u-1|u)\big(z = (x-u)y\big)}_{\mathbf{P}}.$$

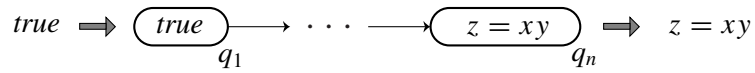
L'implication **P** peut s'écrire plus simplement

$$\big(z = (x-u)y + y \text{ et } u > 0\big) \;\Rightarrow\; z = (x - (u-1))y.$$

Elle est vraie pour tout $(u, x, y, z) \in \mathbb{N}^4$. Donc $[\![u := u-1]\!]\langle A_5\rangle \subset A_3$ est vraie.

5.3.6. Exercice

Construire une machine assertée



agissant dans $\mathbb{N}^5$, correspondant au programme suivant :

```
Var x, y, z, u, v : ℕ;
u := x;
v := y;
z := 0;
while u ≠ 0 do
    if odd(u) then z := z + v;
    u := u div 2;
    v := v * 2
od.
```

La machine aura un seul état initial (q_1) et un seul état final q_n. Elle sera de type $(\mathbb{N}^5, \Omega)$ où Ω est l'ensemble des relations d'affectation et de test

$$[\![u := x]\!] = \left[\!\!\left[\begin{array}{l} u := x \\ (x, y, z, u, v) \in \mathbb{N}^5 \end{array}\right]\!\!\right], \quad \text{etc.}$$

$$[\![\text{if } (u = 0)]\!] = \left[\!\!\left[\begin{array}{l} \text{if } (u = 0) \\ (x, y, z, u, v) \in \mathbb{N}^5 \end{array}\right]\!\!\right], \quad [\![\text{if odd}(u)]\!] = \ldots$$

5.4 Machines déterministes

5.4.1. Produit de deux types de machines

Soient E_1 et E_2 deux ensembles, R_1 une relation dans E_1 et R_2 une relation dans E_2.

On définit la *composition parallèle* de R_1 et R_2, notée $R_1 /\!/ R_2$, comme étant la relation suivante dans l'ensemble $E = E_1 \times E_2$:

$$R_1 /\!/ R_2 = \{(x_1, x_2) \mapsto (y_1, y_2) \mid x_1 R_1 y_1 \text{ et } x_2 R_2 y_2\}. \tag{1}$$

Exécuter « l'instruction » $R_1 /\!/ R_2$ à partir d'un couple $(x_1, x_2) \in E_1 \times E_2$ revient à exécuter simultanément ou « parallèlement » R_1 à partir de x_1 dans l'espace E_1 et R_2 à partir de x_2 dans l'espace E_2. Il faut bien voir que les *deux* exécutions doivent être possibles pour que l'exécution de $R_1 /\!/ R_2$ le soit. Autrement dit, on a

$$\mathrm{Dom}(R_1 /\!/ R_2) = \mathrm{Dom}(R_1) \times \mathrm{Dom}(R_2). \tag{2}$$

Un couple (x_1, x_2) tel que $x_1 \in \mathrm{Dom}(R_1)$ et $x_2 \notin \mathrm{Dom}(R_2)$, par exemple, n'appartient pas au domaine de $R_1 /\!/ R_2$.

Un cas particulier important est celui où la relation R_1 (resp. R_2) est la relation identique de E_1 (resp. de E_2). On a

$$R_1 /\!/ \mathrm{Id}_{E_2} = \{(x_1, x_2) \mapsto (y_1, x_2) \mid x_1 R_1 y_1 \text{ et } x_2 \in E_2\}$$
$$\mathrm{Id}_{E_1} /\!/ R_2 = \{(x_1, x_2) \mapsto (x_1, y_2) \mid x_1 \in E_1 \text{ et } x_2 R_2 y_2\}.$$

On parle souvent de $R_1 /\!/ \mathrm{Id}_{E_2}$ comme étant l'instruction « R_1 *executée dans* $E_1 \times E_2$ », et l'on écrit abusivement R_1 au lieu de $R_1 /\!/ \mathrm{Id}_{E_2}$. On écrit de même R_2, abusivement, pour la relation $\mathrm{Id}_{E_1} /\!/ R_2$, et l'on parle de l'instruction « R_2 exécutée dans $E_1 \times E_2$ ».

Le *produit de deux types de machines* (E_1, Ω_1), (E_2, Ω_2) est un type de machine noté $(E_1, \Omega_1) \otimes (E_2, \Omega_2)$. Il est défini comme suit :

$$(E_1, \Omega_1) \otimes (E_2, \Omega_2) = (E_1 \times E_2, \Omega)$$

où
$$\Omega = \{\alpha_1 /\!/ \mathrm{Id}_{E_2} \mid \alpha_1 \in \Omega_1\} \cup \{\mathrm{Id}_{E_1} /\!/ \alpha_2 \mid \alpha_2 \in \Omega_2\}.$$

On peut dire que les instructions du type produit sont celles de l'un et l'autre des deux types facteurs, mais effectuées dans $E_1 \times E_2$.

5.4.2. Exemple : machine à registre de lecture et pile

Soit X un ensemble. Nous allons définir deux types de machines (E_1, Ω_1), (E_2, Ω_2), avec

$$E_1 = E_2 = \mathrm{W}(X).$$

Le premier sera appelé le type *registre de lecture d'alphabet* X et le deuxième le type *pile d'alphabet* X. On peut considérer un registre d'alphabet X comme étant un organe de mémoire pouvant contenir un mot quelconque sur X, ce mot lui-même étant l'état de mémoire du registre. De même pour une pile d'alphabet X.

Le répertoire d'instructions Ω_1 du type *registre de lecture* d'alphabet X — nous le désignerons plutôt par Ω_r (r pour registre) — se compose des relations suivantes :

Premièrement, pour chaque $a \in X$, la relation

$$\mathtt{LIRE}_a = \{\langle\!\langle\, a \,\rangle\!\rangle x \mapsto x \mid x \in \mathrm{W}(X)\}.$$

Deuxièmement, la relation de test « registre vide » :

$$\mathtt{rVIDE} = \{\Lambda \mapsto \Lambda\} = \mathrm{Id}_{\{\Lambda\}}.$$

Troisièmement la relation neutre $\mathtt{NOP}_\mathrm{r} = \mathrm{Id}_{E_1}$.

Le domaine de la relation $\texttt{LIRE}_a$ (pour un $a \in X$) est l'ensemble des séquences sur X qui commencent par a, donc de la forme $\langle\!\langle\, a \,\rangle\!\rangle x$. L'exécution de cette instruction « consomme » le premier élément de la séquence. L'instruction ne peut pas être exécutée à partir de la séquence vide ou d'une séquence non vide qui ne commence pas par a.

Le répertoire d'instructions Ω_2 du type *pile* d'alphabet X — nous le désignerons par Ω_p (p pour pile) — se compose des relations suivantes :

Premièrement, pour chaque $a \in X$, les relations

$$\begin{aligned} \texttt{PUSH}_a &= \{x \mapsto \langle\!\langle\, a \,\rangle\!\rangle x \mid x \in \mathrm{W}(X)\}; \\ \texttt{POP}_a &= \{\langle\!\langle\, a \,\rangle\!\rangle x \mapsto x \mid x \in \mathrm{W}(X)\}. \end{aligned}$$

Deuxièmement, la relation de test « pile vide » :

$$\texttt{pVIDE} = \{\Lambda \mapsto \Lambda\} = \mathrm{Id}_{\{\Lambda\}}.$$

Troisièmement la relation neutre $\texttt{NOP}_\mathrm{p} = \mathrm{Id}_{E_2}$.

Soient $\texttt{a}$, $\texttt{b}$ deux objets distincts et $X = \{\texttt{a}, \texttt{b}\}$.

L'accepteur fini (M, I, F) représenté dans la figure 5.13 est un exemple de machine du type produit :

$$\text{(registre de lecture d'alphabet } X) \otimes \text{(pile d'alphabet } X).$$

Autrement dit, cette machine agit dans l'ensemble

$$E = E_1 \times E_2 = \mathrm{W}(X) \times \mathrm{W}(X),$$

et son répertoire d'opérations se compose de toutes les opérations de l'un ou l'autre des ensembles Ω_r, Ω_p, *effectuées dans* $E_1 \times E_2$ au sens de 5.4.1, c'est-à-dire que $\texttt{LIRE}_a$ est mis pour $\texttt{LIRE}_a \mathbin{/\!/} \mathrm{Id}_{E_2}$, $\texttt{PUSH}_a$ est mis pour $\mathrm{Id}_{E_1} \mathbin{/\!/} \texttt{PUSH}_a$, etc. L'opération $\texttt{NOP}$ est la fonction identique $\mathrm{Id}_E = \mathrm{Id}_{(E_1 \times E_2)} = \mathrm{Id}_{E_1} \mathbin{/\!/} \mathrm{Id}_{E_2} = \texttt{NOP}_\mathrm{r} \mathbin{/\!/} \texttt{NOP}_\mathrm{p}$. La relation $|M|$ dans E définie par cette machine est la relation

$$\big(\texttt{LIRE}_\texttt{a}\texttt{PUSH}_\texttt{a} \cup \texttt{LIRE}_\texttt{b}\texttt{PUSH}_\texttt{b}\big)^*\texttt{NOP}\big(\texttt{LIRE}_\texttt{a}\texttt{POP}_\texttt{a} \cup \texttt{LIRE}_\texttt{b}\texttt{POP}_\texttt{b}\big)^*\texttt{rVIDE}\texttt{pVIDE}.$$

Le facteur neutre $\texttt{NOP}$ peut être évidemment omis dans ce produit.

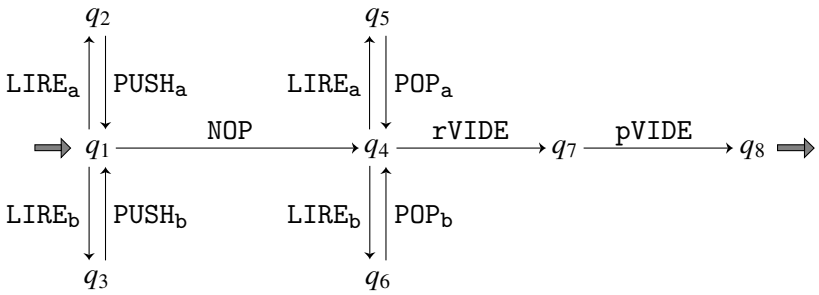


Figure 5.13

5.4.3. Exercice

On rappelle qu'un palindrome sur un alphabet X est une séquence $x \in \mathrm{W}(X)$ telle que $\rho(x) = x$, où ρ est la fonction de renversement des séquences sur X. En particulier, les palindromes de *longueur paire* sur X sont les séquences de la forme $x\rho(x) = \langle\!\langle\, x[1], \ldots, x[n], x[n], \ldots, x[1] \,\rangle\!\rangle$.

Montrer que la relation $|M|$ définie par la machine M de la figure 5.13 dans l'ensemble $E = \mathrm{W}(\{\mathtt{a}, \mathtt{b}\}) \times \mathrm{W}(\{\mathtt{a}, \mathtt{b}\})$ est l'ensemble

$$\big\{(x, y) \mapsto (\Lambda, \Lambda) \;\big|\; (x, y) \in E \text{ et } \exists u \in \mathrm{W}(X) : x = u\rho(u)y\big\}.$$

La relation $|M|$ est donc une fonction constante. Son domaine est l'ensemble des couples de séquences $(x, y) \in E$ tels que x est une séquence de la forme wy, où w est un palindrome de longueur paire. En particulier, on a $(x, \Lambda) \in \mathrm{Dom}(|M|)$ si et seulement si x est un palindrome de longueur paire. On dit que M *accepte* ainsi les palindromes de longueur paire.

5.4.4. Exercice

Modifier la machine M de la figure 5.13 de telle manière qu'elle accepte tous les palindromes sur $\{\mathtt{a}, \mathtt{b}\}$, de longueur paire ou impaire, c'est-à-dire que l'on ait $(x, \Lambda) \in \mathrm{Dom}(|M|)$ si et seulement si x est un palindrome quelconque sur $\{\mathtt{a}, \mathtt{b}\}$.

5.4.5. Machines déterministes

Une machine (M, I, F) de type (E, Ω) est dite *déterministe* — en tant que machine — si elle est déterministe en tant qu'accepteur fini sur Ω, au sens de 4.4.1, et si elle satisfait en outre aux trois conditions suivantes :

(1) Pour chaque transition (p, α, q) de M, la relation α est une fonction.

(2) Pour deux transitions (p, α, q) et (p, α', q') de M partant du même état p, on a toujours $\mathrm{Dom}\alpha \cap \mathrm{Dom}\alpha' = \varnothing$. Au plus une des deux instructions α, α' peut être exécutée à partir d'un état de mémoire quelconque.

(3) Pour toute transition (p, α, q) de M, on a $p \notin F$. Autrement dit, il n'y pas de transition partant d'un état final.

Exemples. La machine de multiplication de 5.3.5 (figure 5.11) est déterministe. C'est en effet un accepteur déterministe au sens de 4.4.1. Les relations d'affectation et de test sont toujours des fonctions. Les relations $[\![\text{if } (u = 0)]\!]$, $[\![\text{if } (u \neq 0)]\!]$ ont des domaines disjoints. Il n'y a pas de transition partant de l'état final q_4.

La machine de la figure 5.13 n'est pas déterministe. C'est un accepteur déterministe au sens de 4.4.1, et elle satisfait aux conditions (1) et (3). Par contre, elle ne satisfait pas à la condition (2). Les relations $\mathtt{LIRE}_{\mathtt{a}}$ et $\mathtt{NOP}$ des transitions

$$(q_1, \mathtt{LIRE}_{\mathtt{a}}, q_2), \quad (q_1, \mathtt{NOP}, q_4)$$

n'ont pas des domaines disjoints. Le domaine de la relation $\mathtt{NOP}$ est l'ensemble $E = E_1 \times E_2$. Ce domaine contient donc celui de n'importe quelle autre relation dans E. En particulier, il contient celui de la relation $\mathtt{LIRE}_{\mathtt{a}}$ (exéctutée dans $E_1 \times E_2$), à savoir l'ensemble des couples de séquences de la forme $(\langle\!\langle\, \mathtt{a} \,\rangle\!\rangle x,\ y)$. À partir d'un tel état de mémoire, on peut exécuter aussi bien l'instruction $\mathtt{LIRE}_{\mathtt{a}}$ que l'instruction $\mathtt{NOP}$.

Bien que cette machine M ne soit pas déterministe, nous avons vu () que sa relation $|M|$ est une fonction. La réciproque du théorème suivant est donc fausse.

Théorème. Soit (E, Ω) un type de machine. Pour toute machine déterministe (M, I, F) de type (E, Ω), la relation $|M|$ est une fonction.

La démonstration se base sur la formule 5.2.5(2) et les propriétés qui définissent une machine déterministe. Il assez évident qu'en vertu de ces propriétés, étant donné un état de mémoire initial quelconque $a \in E$, il existe au plus une séquence $\sigma \in L^M_{iF}$ qui soit exécutable à partir de a, c'est-à-dire telle que a appartienne au domaine de la relation produit

$R = \sigma[1] \cdots \sigma[n]$ ($n = \lg(\sigma)$). Cette relation R est elle-même une fonction puisque toutes les relations $\sigma[k]$ sont des fonctions. Donc il existe au plus un $b \in E$ tel que $(a, b) \in |M|$ (5.2.5(2)).

5.4.6. Exercice

La relation $|M|$ dans $E = \mathrm{W}(\{\mathtt{a}, \mathtt{b}\}) \times \mathrm{W}(\{\mathtt{a}, \mathtt{b}\})$ définie par la machine M de la figure 5.13 est donnée par l'expression régulière

$$\big(\mathtt{LIRE_a PUSH_a} \cup \mathtt{LIRE_b PUSH_b}\big)^* \big(\mathtt{LIRE_a POP_a} \cup \mathtt{LIRE_b POP_b}\big)^* \mathtt{rVIDE pVIDE}.$$

Construire une machine équivalente (5.2.10) utilisant le répertoire de relations $\mathtt{LIRE_a}$, $\mathtt{LIRE_b}$, $\mathtt{PUSH_a}$, $\mathtt{PUSH_b}$, $\mathtt{POP_a}$, $\mathtt{POP_b}$, $\mathtt{rVIDE}$, $\mathtt{pVIDE}$, mais *sans la relation* $\mathtt{NOP}$. On demande que cette machine soit un *accepteur déterministe* au sens de 4.4.1 et satisfasse aux conditions 5.4.5(1) et 5.4.5(3). Montrer que la machine construite ne satisfait pas à la condition 5.4.5(2).

5.4.7. Exercice

Soit $X = \{\mathtt{a}, \mathtt{b}\}$ un alphabet à deux éléments et $X' = \{\mathtt{a}, \mathtt{b}, \mathtt{¢}\}$, où l'on suppose que l'élément $\mathtt{¢}$ est distinct de $\mathtt{a}$ et $\mathtt{b}$. Donner une machine déterministe M de type (registre de lecture d'alphabet X') $\otimes$ (pile d'alphabet X') telle que l'on ait

$$(x, \Lambda) \in \mathrm{Dom}(|M|) \Leftrightarrow \exists u \in \mathrm{W}(X) : x = u \langle\!\langle\, \mathtt{¢} \,\rangle\!\rangle \rho(u).$$

On dit qu'une séquence x de cette forme est un « palindrome avec marqueur central $\mathtt{¢}$ » sur l'alphabet $X' = \{\mathtt{a}, \mathtt{b}, \mathtt{¢}\}$.

5.5 Machines de Turing

5.5.1. Introduction : le concept de calculabilité

Alan Mathison Turing (1912–1954) était un mathématicien anglais. En 1937, à l'époque où l'ordinateur n'avait pas encore été inventé, il devint célèbre en publiant un article[10] dans lequel il fut le premier à donner un sens mathématique précis au mot « calculable », en inventant pour cela un type de machine abstrait qui fut rapidement reconnu comme étant *universel*, en ce sens que toute espèce de machine peut être simulée par une machine de ce type.

L'adjectif « calculable » peut s'appliquer en particulier aux fonctions. Il existe des fonctions calculables et des fonctions non-calculables — on le sait depuis de Turing. Pour préciser cela, repartons de la définition mathématique du concept de fonction (1.3.9). Une fonction F est un ensemble de couples tel que, pour tout objet x, il existe au plus un objet y tel que $(x, y) \in F$, et si cet objet y existe il est désigné par $F(x)$. Cette définition ne parle pas d'un quelconque *procédé de calcul* de $F(x)$ à partir de x. Certes, pour la plupart des fonctions auxquelles on a affaire dans les cours de mathématique, il existe un tel procédé — un algorithme. Nous voulons seulement signaler ici que cette propriété de *calculabilité* n'est pas contenue dans le concept de fonction.

Démontrer la non-calculabilité d'une fonction F c'est démontrer la non-existence d'une machine — au sens le plus général — capable d'effectuer une certaine tâche : le calcul de $F(x)$ pour tout $x \in \mathrm{Dom}\, F$. Mais qu'entend-on par « machine au sens le plus général » ?

---

[10] A. M. Turing : On Computable Numbers, with an Application to the Enscheidungsproblem, Proc. London Math. Soc. (2) **42**, 230–265 (1937).

On ne peut pas donner de réponse *définitive* à cette question, mais seulement une réponse relative aux types de machines que l'on connaît actuellement. Il se trouve cependant que depuis Turing, personne n'a réussi inventer un type de machine capable de calculer des fonctions qu'aucune machine de Turing ne puisse calculer. Dès la parution de l'article de Turing, les mathématiciens se convainquirent rapidement que l'on ne trouverait jamais un type de machine plus « puissant ». Il ne s'agit pas de puissance au sens de la vitesse de calcul — les machines de Turing sont au contraire d'une lenteur insupportable — mais seulement de la capacité ou de l'incapacité de résoudre un problème, peu importe en combien de temps. Cette conviction est acquise depuis plus de 50 ans.

Les exemples connus de fonctions non-calculables sont souvent des fonctions à réponse booléenne, c'est-à-dire des fonctions F telles que, pour toute donnée $x \in \mathrm{Dom}(F)$, on a $F(x) \in \{0, 1\}$, ou si l'on veut: $F(x) \in \{\texttt{true}, \texttt{false}\}$. D'autre part, ces fonctions F sont toujours définies au moyen d'expressions conditionnelles. Cela veut dire que l'on définit F en disant que, pour tout $x \in \mathrm{Dom}\,F$, $F(x) = 1$ si une certaine condition $P(x)$ est satisfaite, et que $F(x) = 0$ si $P(x)$ n'est pas satisfaite. Cela s'écrit d'habitude

$$F(x) = \begin{cases} 1 & \text{si } P(x) \\ 0 & \text{si non}\,P(x). \end{cases} \tag{1}$$

Le choix du domaine de F et de la propriété $P(x)$ correspondent en général à un problème qui est un « défi ». Un tel défi avait été posé en 1900 par le grand mathématicien allemand David Hilbert. Il s'agissait de trouver un procédé sytématique permettant de *décider* (en allemand *entscheiden*), pour une proposition arbitraire d'un langage du premier ordre, si cette proposition est un théorème de logique des prédicats ou non. Ce problème historique fut appelé en anglais « *the* Entscheidungsproblem » (sic), et Turing montra dans son article qu'il n'a pas de solution. En d'autres termes, si l'on prend pour domaine de F l'ensemble des propositions d'un langage du premier ordre, et pour propriété $P(x)$ la propriété « x est un théorème de logique des prédicats », alors la fonction F définie par (1) n'est pas calculable. Il n'existe pas d'algorithme permettant de répondre correctement « oui » ou « non » pour toute proposition donnée x. Plus exactement: il n'existe pas de machine de Turing capable de calculer $F(x)$ pour toute proposition x. Depuis lors, on est convaincu qu'il n'existe pas de machine *de type quelconque* capable de calculer cette fonction. Toutefois, certains chercheurs aimant à se mouvoir à la frontière de la science-fiction professent parfois une opinion contraire.

L'usage d'expressions conditionnelles de la forme (1) intervient directement ou indirectement dans toute définition de fonction non-calculable. Or, le fait que l'on *puisse* définir une fonction de cette manière, autrement dit qu'il *existe*, au sens mathématique, une fonction F de domaine donné satisfaisant à (1), est un théorème simple, généralement admis comme une évidence, mais dont la démonstration utilise de manière décisive le principe logique du *tiers exclu*, à savoir que, quel que soit x, la proposition $P(x)$ ou non$P(x)$ est vraie. Si l'on exclut ce principe de la logique, en se limitant à la logique dite intuitionniste[11], et si l'on en tire les conséquences en expurgeant les mathématiques de tout ce qui en dépend, alors on ne peut plus définir mathématiquement de fonctions non-calculables. Mais cela ne fait que changer le nom d'une distinction qui existe bel et bien. Dire, en mathématique classique, qu'une certaine expression de la forme (1) définit une fonction non-calculable revient à dire, en mathématique intuitionniste, que cette expression ne définit pas de fonction du tout.

---

[11] J. Zahnd, *Logique élémentaire*, §6.4, Presses Polytechniques et Universitaires Romandes, 1998.

5.5.2. Le concept « physique » de machine de Turing

Il est commode de se représenter d'abord une machine de Turing comme une machine physique (figure 5.14). Celle-ci se compose d'un organe de mémoire qui est un ruban infini (en anglais *tape*) divisé en cellules. Chaque cellule peut contenir un symbole, appartenant à un alphabet fini Σ déterminé, appelé *l'alphabet de ruban* de la machine et dont la donnée fait partie de celle de la machine. Les cellules vides sont considérées comme contenant un symbole particulier dit « blanc », appartenant à l'alphabet Σ. Le contenu de l'ensemble des cellules du ruban est soumis à la restriction suivante, qui est d'une grande importance : le ruban comporte une infinité de cellules, mais il y toujours seulement un nombre fini de cellules non-vides, c'est-à-dire contenant un symbole autre que le symbole blanc.

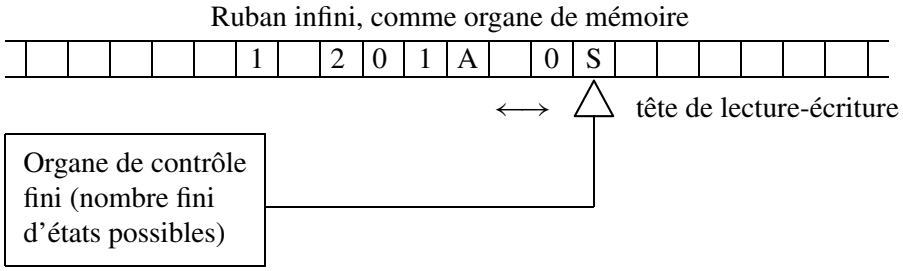


Figure 5.14

Le ruban est muni d'une tête de lecture-écriture qui peut se déplacer à gauche ou à droite, d'une cellule à la fois. Cette tête est commandée par un organe de contrôle qui est un dispositif fini, c'est-à-dire possédant un nombre fini d'états. La tête de lecture-écriture peut exécuter les instructions suivantes :

- se déplacer d'une cellule à gauche
- se déplacer d'une cellule à droite
- tester si le symbole inscrit dans la cellule est égal à un symbole donné
- remplacer le symbole de la cellule par un symbole donné.

5.5.3. Le type d'une machine de Turing

Nous allons définir maintenant un type de machine (E, Ω) au sens de 5.2.3, correspondant au type « physique » que nous venons de décrire.

L'ensemble E des états de mémoire d'une telle machine est l'ensemble des états du ruban — un état du ruban étant caractérisé par le contenu de toutes les cellules et par la position de la tête de lecture/écriture — et Ω est l'ensemble des instructions décrites ci-dessus. Nous les définirons mathématiquement plus loin. L'ensemble E dépend de l'alphabet de ruban Σ de la machine *et* de l'élément de cet alphabet Σ considéré comme « blanc ». Nous désignerons cet élément par $\flat$. L'ensemble E dépend du choix de $\flat$ parce que les états de ruban doivent respecter la condition que seul un nombre fini de cellules contient un élément de Σ distinct de $\flat$. De même, le répertoire d'instructions dépend de l'alphabet Σ, puisqu'il est question de lire et d'écrire dans les cellules des éléments de Σ.

On ne peut donc pas définir un type de machine unique (E, Ω) qui soit celui de toutes les machines de Turing, mais, pour chaque couple $(\Sigma, \flat)$ où Σ est un ensemble fini et $\flat$ est un élément choisi de Σ, il y a un type de machine

$$\big(\mathsf{T}(\Sigma, \flat),\ \Omega(\Sigma)\big) \tag{1}$$

appelé le type *Turing d'alphabet* Σ *avec symbole blanc* $\flat$, qui se compose de :

$\mathsf{T}(\Sigma, \flat)$	l'ensemble des états d'un ruban de machine de Turing d'alphabet Σ avec symbole blanc $\flat$;
$\Omega(\Sigma)$	le répertoire d'opérations standard exécutées par la tête de lecture-écriture d'un tel ruban : lecture, écriture, déplacements.

Dans ce paragraphe, nous allons définir formellement l'ensemble $\mathsf{T}(\Sigma, \flat)$ et au paragraphe suivant nous ferons de même pour les instructions du répertoire $\Omega(\Sigma)$ correspondant. Pour simplifier l'écriture, nous utiliserons le symbole $\mathsf{T}(\Sigma, \flat)$ aussi bien pour désigner l'ensemble des états de ruban que pour désigner le type (1), à savoir l'ensemble des états de ruban « muni » de son répertoire d'opérations standard.

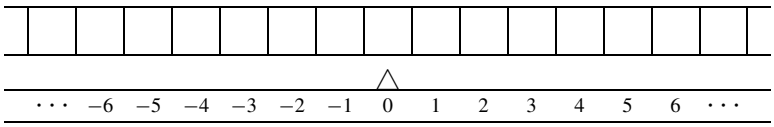


Figure 5.15

Nous considérons la tête de lecture-écriture comme faisant partie de l'organe de mémoire appelé ruban, de sorte que sa position par rapport au ruban fait partie de l'état de cette mémoire. Un déplacement de la tête est un changement de l'état de mémoire. Pour formaliser cela, nous imaginons un « système de coordonnées » lié à la tête de lecture, (figure 5.15), dans lequel chaque cellule est repérée par son abscisse, qui est un entier $k \in \mathbb{Z}$, où $\mathbb{Z}$ est l'ensemble des entiers $0, \pm 1, \pm 2, \pm 3, \ldots$ La cellule qui se trouve sous la tête de lecture a toujours l'abscisse 0.

Un état du ruban est déterminé par la donnée, pour chaque $k \in \mathbb{Z}$, du symbole contenu dans la cellule d'abscisse k. Autrement dit, un état peut être considéré comme une fonction de domaine $\mathbb{Z}$ associant à chaque $k \in \mathbb{Z}$ un élément de l'alphabet Σ. Un état de ruban α est donc une application de $\mathbb{Z}$ dans Σ (figure 5.16). Mais cette application est soumise à la condition que le nombre de cellules contenant un symbole non blanc doit être fini. Un état de ruban est donc une fonction

$$\alpha : \mathbb{Z} \to \Sigma$$

qui satisfait à la condition suivante : l'ensemble

$$\{k \in \mathbb{Z} \mid \alpha(k) \neq \flat\}$$

est fini. Il revient au même de dire que cet ensemble est *borné*, c'est-à-dire qu'il existe un $m \in \mathbb{N}$ tel que

$$\forall k \in \mathbb{Z} : \quad \alpha(k) \neq \flat \;\Rightarrow\; |k| \leq m.$$

Nous désignons par $\mathsf{T}(\Sigma, \flat)$ l'ensemble des états de rubans ainsi définis. La lettre T se rapporte au mot « tape » (ruban) ou à Turing si l'on préfère. En résumé, nous posons la définition suivante :

Définition. Soient Σ un alphabet fini et $\flat \in \Sigma$. Nous appellons *ensemble des états de ruban d'alphabet* Σ *avec symbole blanc* $\flat$ et nous désignons par $\mathsf{T}(\Sigma, \flat)$ l'ensemble des fonctions $\alpha : \mathbb{Z} \to \Sigma$ qui satisfont à la condition

$$\exists m \in \mathbb{N} : \forall k \in \mathbb{Z} : \quad \alpha(k) \neq \flat \;\Rightarrow\; |k| \leq m.$$

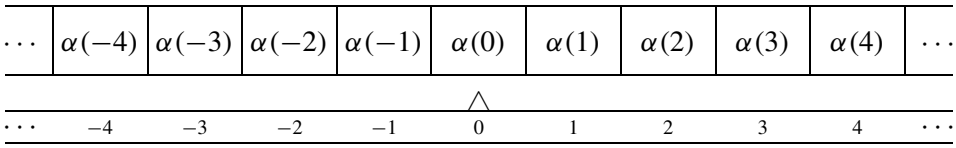


Figure 5.16

État de ruban $\alpha : \mathbb{Z} \to \Sigma$

Exemple. Soient $\Sigma = \{0, 1\}$ et $\flat = 0$. La figure 5.17 représente un état de ruban α de type $\mathsf{T}(\Sigma, \flat)$, c'est-à-dire une fonction $\alpha : \mathbb{Z} \to \{0, 1\}$ telle que $\alpha(k) \neq 0$ seulement pour un nombre fini de valeurs de k. La figure sous-entend en effet que toutes les cellules d'abscisse $k < -6$ ou $k > 6$ contiennent le symbole 0.

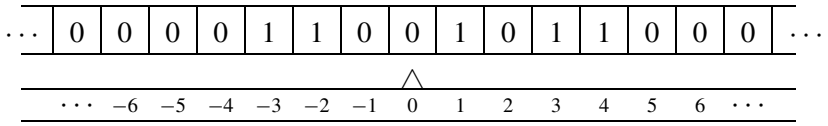


Figure 5.17

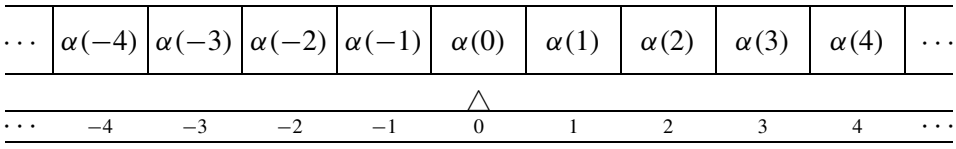
5.5.4. Répertoire d'instructions d'une machine de Turing

Soient Σ un alphabet fini et $\flat \in \Sigma$. Nous définissons dans ce paragraphe le répertoire d'instructions d'une machine de Turing de type $\mathsf{T}(\Sigma, \flat)$. Ces instructions sont bien entendu des relations dans l'ensemble des états de mémoire $\mathsf{T}(\Sigma, \flat)$, conformément à la notion générale de type de machine. Ces relations seront toutes des fonctions, mais pas toutes des fonctions de domaine $\mathsf{T}(\Sigma, \flat)$.

L'instruction `L`. Il s'agit de l'instruction « déplacer la tête de lecture d'une cellule à gauche » (Left). L'exécution de cette instruction à partir d'un état de ruban α est dépeinte dans la figure 5.18. L'exécution aboutit à l'état de ruban β tel que $\beta(k) = \alpha(k - 1)$ pour tout $k \in \mathbb{Z}$. Par définition, `L` est une application de l'ensemble $\mathsf{T}(\Sigma, \flat)$ dans lui-même et, pour tout $\alpha \in \mathsf{T}(\Sigma, \flat)$, on a :

$$\forall k \in \mathbb{Z} : \quad \big(\mathtt{L}(\alpha)\big)(k) = \alpha(k - 1).$$

État de ruban α



État de ruban $\mathtt{L}(\alpha)$

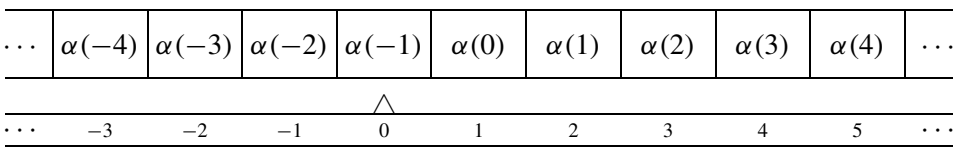


Figure 5.18

Exécution de l'opération `L`

L'instruction R. Il s'agit de l'instruction « déplacer la tête de lecture d'une cellule à droite » (Right). Le lecteur n'a pas besoin d'encore un dessin pour voir que l'exécution de cette instruction à partir d'un état de ruban α aboutit à l'état de ruban β tel que $\beta(k) = \alpha(k+1)$ pour tout $k \in \mathbb{Z}$. Par définition, R est une application de l'ensemble $\mathsf{T}(\Sigma, \flat)$ dans lui-même et, pour tout $\alpha \in \mathsf{T}(\Sigma, \flat)$, on a:

$$\forall k \in \mathbb{Z} : \quad \big(\mathtt{R}(\alpha)\big)(k) = \alpha(k+1).$$

Les instructions $\mathtt{S}_a$ ($a \in \Sigma$). Pour chaque élément a de l'alphabet Σ, l'instruction $\mathtt{S}_a$ est l'instruction de test de la condition $\alpha(0) = a$, la notion d'instruction de test étant prise au sens de 5.1.8. Autrement dit, $\mathtt{S}_a$ est la fonction identique de l'ensemble des états de ruban α tels que $\alpha(0) = a$.

$$S_a = \{\alpha \mapsto \alpha \mid \alpha \in \mathsf{T}(\Sigma, \flat) \text{ et } \alpha(0) = a\}.$$

La notation $\mathtt{S}_a$ peut se lire « See a » ou « Scan a », ce qui veut dire: constater qu'il y a a sous la tête de lecture. Cela ne peut se faire évidemment que s'il y a effectivement a sous la tête de lecture.

Les instructions $\mathtt{P}_a$ ($a \in \Sigma$). Pour chaque élément a de l'alphabet Σ, l'instruction $\mathtt{P}_a$ est l'instruction « imprimer a », ou « Print a », dans la cellule qui se trouve sous la tête de lecture-écriture. L'exécution de cette instruction à partir d'un état de ruban α est dépeinte dans la figure 5.19. Par définition, $\mathtt{P}_a$ est une application de l'ensemble $\mathsf{T}(\Sigma, \flat)$ dans lui-même et, pour tout $\alpha \in \mathsf{T}(\Sigma, \flat)$, on a:

$$\forall k \in \mathbb{Z} : \quad \big(\mathtt{P}_a(\alpha)\big)(k) = \begin{cases} a & \text{si } k = 0; \\ \alpha(k) & \text{si } k \neq 0. \end{cases}$$

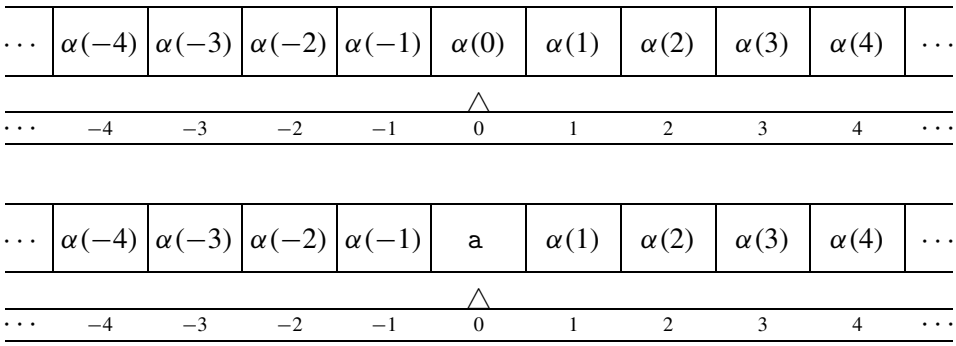


Figure 5.19

Exécution de l'instruction $\mathtt{P}_a$

La définition du répertoire d'instructions du type de machine $\mathsf{T}(\Sigma, \flat)$ est achevée. Comme l'alphabet de ruban Σ est fini — cela fait partie de la définition de ce type de machine — on voit que ce répertoire d'instructions est fini. Il comporte exactement $2n + 2$ instructions distinctes si n est le nombre d'éléments de Σ.

5.5.5. Représentation des états de rubans

Soient Σ un ensemble fini et $\flat \in \Sigma$. Étant donné deux séquences $x, y \in \mathrm{W}(\Sigma)$, nous désignons par

$$x \rhd y$$

l'état de ruban de type $\mathsf{T}(\Sigma, \natural)$ dépeint dans la figure 5.20, où l'on suppose que $\lg(x) = 3$ et $\lg(y) = 4$ pour fixer les idées. L'expression $x \rhd y$ suggère l'image de la tête de lecture-écriture, symbolisée par $\rhd$, pointant sur le premier élément de la séquence y. Nous appellerons le symbole $\rhd$ le *curseur*. La définition générale exacte est donnée ci-dessous.

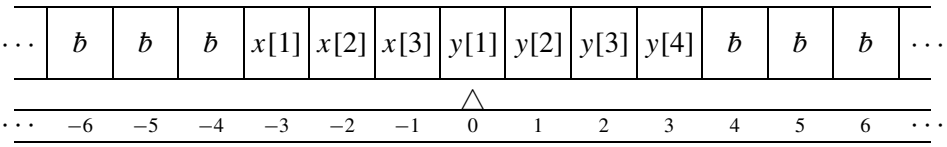


Figure 5.20
État de ruban $\alpha = x \rhd y$

Définition. Soient Σ un ensemble fini et $\natural \in \Sigma$. Pour chaque couple $(x, y) \in \mathrm{W}(\Sigma) \times \mathrm{W}(\Sigma)$, l'état de ruban de type $\mathsf{T}(\Sigma, \natural)$ désigné par $x \rhd y$ est la fonction α de domaine $\mathbb{Z}$ telle que, pour tout $k \in \mathbb{Z}$:

$$\alpha(k) = \begin{cases} (\rho(x))[-k] & \text{si } -\lg(x) \leq k \leq -1; \\ y[k+1] & \text{si } 0 \leq k \leq \lg(y) - 1; \\ \natural & \text{si } k < -\lg(x) \text{ ou } k \geq \lg(y). \end{cases} \tag{1}$$

La fonction ρ est évidemment la fonction de renversement des séquences sur Σ.

Le cas particulier des états de ruban $x \rhd y$ qui correspondent à un couple de séquences $(x, y) \in \mathrm{W}(\Sigma) \times \mathrm{W}(\Sigma)$ dans lequel $x = \Lambda$ ou $y = \Lambda$ est illustré par la figure 5.21. Le lecteur vérifiera que ces états de rubans vérifient bien la formule (1).

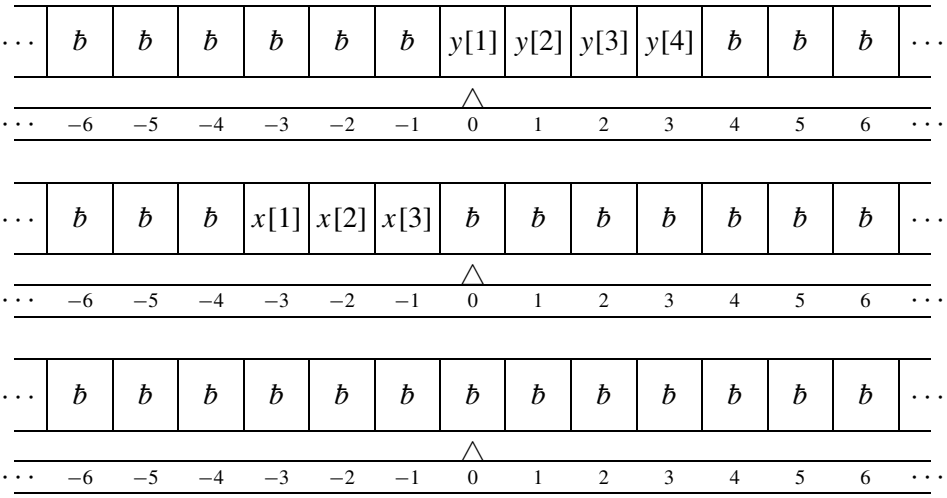


Figure 5.21
États de ruban $\Lambda \rhd y$, $x \rhd \Lambda$, $\Lambda \rhd \Lambda$.

On convient d'abréger

$$\Lambda \rhd y \quad \text{par} \quad \rhd y, \qquad x \rhd \Lambda \quad \text{par} \quad x \rhd, \qquad \Lambda \rhd \Lambda \quad \text{par} \quad \rhd .$$

La fonction de représentation des états de ruban. Étant donné un alphabet de ruban Σ et un élément $\natural \in \Sigma$, nous considérons la fonction

$$F = \big\{(x, y) \mapsto x \rhd y \,\big|\, (x, y) \in \mathrm{W}(\Sigma) \times \mathrm{W}(\Sigma)\big\}, \tag{2}$$

qui fait correspondre à chaque couple (x, y) de séquences sur Σ l'état de ruban $x \rhd y$ de type $\mathsf{T}(\Sigma, \flat)$ défini par (1).

Cette fonction n'est *pas injective*. Les états de rubans $x \rhd y$ et $x' \rhd y'$ correspondant à des couples de séquences (x, y) et (x', y') distincts peuvent être identiques: $x \rhd y = x' \rhd y'$. C'est notamment le cas si $(x', y') = (\langle\!\langle\, \flat \,\rangle\!\rangle x, y)$ ou $(x', y') = (x, y \langle\!\langle\, \flat \,\rangle\!\rangle)$. On a en effet les égalités d'états de rubans

$$\langle\!\langle\, \flat \,\rangle\!\rangle x \rhd y = x \rhd y, \qquad x \rhd y \langle\!\langle\, \flat \,\rangle\!\rangle = x \rhd y. \tag{3}$$

En particulier, pour toute séquence $y \in \mathrm{W}(\Sigma)$, on a

$$\Lambda \rhd y = \langle\!\langle\, \flat \,\rangle\!\rangle \rhd y = \langle\!\langle\, \flat, \flat \,\rangle\!\rangle \rhd y = \langle\!\langle\, \flat, \flat, \flat \,\rangle\!\rangle \rhd y = \ldots$$

La fonction (2) est une application *surjective* de l'ensemble $\mathrm{W}(\Sigma) \times \mathrm{W}(\Sigma)$ dans l'ensemble d'états de ruban $\mathsf{T}(\Sigma, \flat)$. En effet, pour tout état de ruban $\alpha \in \mathsf{T}(\Sigma, \flat)$ il existe un couple (x, y) de séquences sur Σ tel que $\alpha = x \rhd y$, car il existe un $m \in \mathbb{N}$ tel que $\alpha(k) = \flat$ pour tout $k \leq m$ et pour tout $k \geq m$ et l'on a $\alpha = x \rhd y$ si l'on prend pour x la séquence $\langle\!\langle\, \alpha(-m), \ldots, \alpha(-1) \,\rangle\!\rangle$ et pour y la séquence $\langle\!\langle\, \alpha(0), \ldots, \alpha(m) \,\rangle\!\rangle$.

On exprime cette propriété de la fonction (2) en disant que tout état de ruban $\alpha \in \mathsf{T}(\Sigma, \flat)$ peut être représenté par un couple (x, y) de séquences sur Σ. Mais cette représentation de α n'est jamais unique, en raison des égalités (3). La fonction (2) peut être appelée la fonction de représentation des états de rubans de type $\mathsf{T}(\Sigma, \flat)$ par des couples de séquences sur Σ.

Les opérations $\mathtt{L}$, $\mathtt{R}$, $\mathtt{S}_a$ $(a \in \Sigma)$, $\mathtt{P}_a$ $(a \in \Sigma)$, en tant que relations dans l'ensemble des états de ruban $\mathsf{T}(\Sigma, \flat)$, s'expriment commodément en utilisant cette représentation des états de ruban. Ainsi, la relation $\mathtt{L}$ dans $\mathsf{T}(\Sigma, \flat)$ est l'ensemble des couples d'états de ruban de la forme $x\langle\!\langle\, a \,\rangle\!\rangle \rhd y \mapsto x \rhd \langle\!\langle\, a \,\rangle\!\rangle y$ tels que $a \in \Sigma$ et $x, y \in \mathrm{W}(\Sigma)$. On voit le curseur se déplacer à gauche dans un tel couple d'états de ruban. Formellement, cela s'écrit

$$\mathtt{L} = \left\{ x\langle\!\langle\, a \,\rangle\!\rangle \rhd y \mapsto x \rhd \langle\!\langle\, a \,\rangle\!\rangle y \;\middle|\; a \in \Sigma \text{ et } x \in \mathrm{W}(\Sigma) \text{ et } y \in \mathrm{W}(\Sigma) \right\}.$$

Symétriquement, on a

$$\mathtt{R} = \left\{ x \rhd \langle\!\langle\, a \,\rangle\!\rangle y \mapsto x\langle\!\langle\, a \,\rangle\!\rangle \rhd y \;\middle|\; a \in \Sigma \text{ et } x \in \mathrm{W}(\Sigma) \text{ et } y \in \mathrm{W}(\Sigma) \right\}.$$

Pour chaque $a \in \Sigma$, l'opération « See a » est la relation

$$\mathtt{S}_a = \left\{ x \rhd \langle\!\langle\, a \,\rangle\!\rangle y \mapsto x \rhd \langle\!\langle\, a \,\rangle\!\rangle y \;\middle|\; x \in \mathrm{W}(\Sigma) \text{ et } y \in \mathrm{W}(\Sigma) \right\}.$$

C'est la fonction identique de l'ensemble des états de ruban de la forme $x \rhd \langle\!\langle\, a \,\rangle\!\rangle y$. Enfin, pour chaque $a \in \Sigma$, on a

$$\mathtt{P}_a = \left\{ x \rhd \langle\!\langle\, b \,\rangle\!\rangle y \mapsto x \rhd \langle\!\langle\, a \,\rangle\!\rangle y \;\middle|\; b \in \Sigma \text{ et } x \in \mathrm{W}(\Sigma) \text{ et } y \in \mathrm{W}(\Sigma) \right\}.$$

5.5.6. Exemple

Un type de ruban particulièrement simple et important est le type $\mathsf{T}(\{0, 1\}, 0)$, d'alphabet de ruban binaire $\Sigma = \{0, 1\}$, avec 0 comme élément blanc ($\flat = 0$). Il se trouve que toute fonction numérique calculable peut l'être au moyen d'une machine de ce type, d'une manière qui sera précisée plus loin. Une machine M de ce type est représentée dans la figure 5.22.

Cette machine M est équivalente au programme suivant:

Var $\alpha : \mathsf{T}(\{0, 1\}, 0)$;
while $\alpha(0) = 0$ **do**
    $\alpha := \mathtt{R}(\alpha)$
od;
$\alpha(0) := 0$.

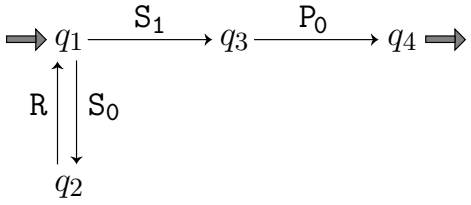


Figure 5.22

Désignons par E l'ensemble des états de ruban $\mathsf{T}(\{0, 1\}, 0)$. Soient x et y des séquences quelconques sur l'alphabet $\{0, 1\}$ et $n \in \mathbb{N}$. Partant de l'état de ruban $x \rhd 0^n 1y$, l'exécution de ce programme aboutit à l'état $x0^n \rhd 0y$. Cela peut être prouvé en vérifiant l'assertation de cette machine donnée dans la figure 5.23, avec les ensembles initiaux à un seul élément

$$A_I = \{w \in E \mid w = x \rhd 0^n 1y\};$$
$$A_F = \{w \in E \mid w = x0^n \rhd 0y\}.$$

D'après le théorème 5.3.3, on a $|M|\langle A_I\rangle \subset A_F$. D'après la formule 1.3.23(4), cela signifie que quels que soient les états de rubans $w, w' \in E$, on a

$$\big(w = x \rhd 0^n 1y \ \text{ et } \ w \mapsto w' \in |M|\big) \ \Rightarrow \ w' = x0^n \rhd 0y.$$

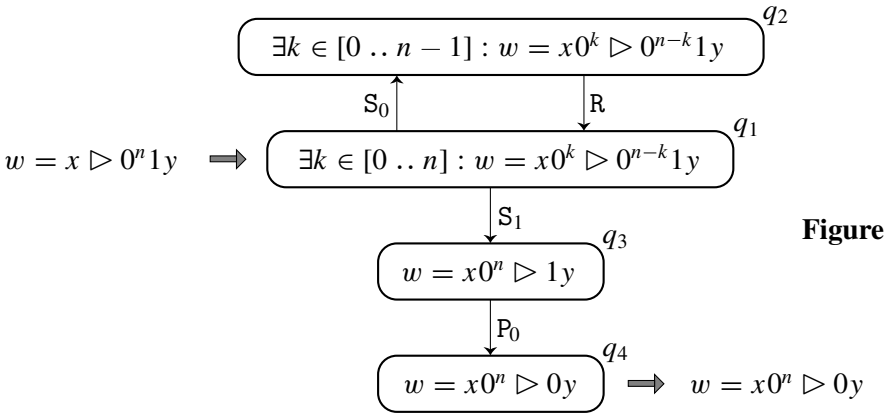


Figure 5.23

Vérification de l'assertation. Nous devons vérifier d'abord que $A_I \subset A_{q_1}$, c'est-à-dire que $w = x \rhd 0^n 1y$ implique $\exists k \in [0\,..\,n] : w = x0^k \rhd 0^{n-k}1y$. Cela est immédiat puisque $x \rhd 0^n 1y$ peut s'écrire $x0^0 \rhd 0^{n-0}1y$ (0^0 est la séquence $\langle\!\langle\, 0 \,\rangle\!\rangle^0 = \Lambda$). Si un état de ruban de la forme $x0^k \rhd 0^{n-k}1y$ appartient au domaine de l'opération $\mathtt{S}_0$, alors $k < n$ (il y a un zéro sous la tête de lecture), donc cet état w appartient à l'ensemble A_{q_2}. Comme $\mathtt{S}_0(w) = w$, cela montre que $\mathtt{S}_0\langle A_{q_1}\rangle \subset A_{q_2}$. Pour un état de ruban $w \in A_{q_2}$, c'est-à-dire de la forme $w = x0^k \rhd 0^{n-k}1y$ avec $k \in [0\,..\,n-1]$, on a $\mathtt{R}(w) = x0^{k+1} \rhd 0^{n-(k+1)}1y$ avec $k+1 \in [0\,..\,n]$, donc $\mathtt{R}(w)$ appartient à A_{q_1}. Cela montre que $\mathtt{R}\langle A_{q_2}\rangle \subset A_{q_1}$. Si $w \in A_{q_1}$, c'est-à-dire si w est de la forme $x0^k \rhd 0^{n-k}1y$ avec $k \in [0\,..\,n]$, et si $w \in \mathrm{Dom}\,\mathtt{S}_1$, alors $k = n$ (il y a un 1 sous la tête de lecture), donc $\mathtt{S}_1(w) = w = x0^n \rhd 1y$. Cela montre que $\mathtt{S}_1\langle A_{q_1}\rangle \subset A_{q_3}$. Les inclusions $\mathtt{P}_0\langle A_{q_3}\rangle \subset A_{q_4}$ et $A_{q_4} \subset A_F$ sont évidentes.

5.5.7. Fonctions de codage et de décodage

Une machine de Turing M de type $\mathsf{T}(\Sigma, \natural)$ peut être utilisée pour calculer une fonction $F : A \to B$ où A et B sont des ensembles quelconques. Pour cela, il faut représenter les éléments de A et de B par des états de rubans, c'est-à-dire introduire un codage de ces éléments sous la forme d'états de ruban de type $\mathsf{T}(\Sigma, \natural)$.

Pour préciser cette idée, admettons d'abord que la relation $|M|$ dans l'ensemble des états de ruban $\mathsf{T}(\Sigma, \flat)$ soit une fonction. Le calcul de $F(a)$, pour un élément a de A, se déroule alors comme suit: on applique à a une fonction de codage $\alpha : A \to \mathsf{T}(\Sigma, \flat)$, puis on applique à l'état de ruban $\alpha(a)$ la fonction $|M|$. En d'autres termes, on exécute le « programme » $|M|$ à partir de l'état de mémoire initial $\alpha(a)$. En admettant que cet état de mémoire initial apartienne au domaine de la fonction $|M|$, l'état de mémoire final correspondant est $|M|(\alpha(a))$. On applique alors à cet état de ruban une fonction de décodage $\beta : \mathsf{T}(\Sigma, \flat) \to B$. La machine M et les fonctions de codage et de décodage α et β doivent être conçues de telle manière que l'on ait $\beta\big(|M|(\alpha(a))\big) = F(a)$ pour tout $a \in A$, autrement dit que la fonction F soit la composition de fonctions

$$F = \beta \circ |M| \circ \alpha = \alpha |M| \beta. \tag{1}$$

Si cette égalité est vérifiée, on dit que F est la fonction calculée par la machine M *moyennant les fonctions de codage et de décodage* α *et* β.

La définition générale de cette notion ne fait pas certaines hypothèses que nous venons de faire. Notamment, le domaine de la fonction de décodage β ne doit pas être nécessairement tout l'ensemble $\mathsf{T}(\Sigma, \flat)$, mais seulement un sous-ensemble de celui-ci. La relation $|M|$ ne doit pas être nécessairement une fonction. La relation composée $\alpha|M|\beta$ peut donc ne pas être une fonction et l'on parle seulement de la *relation* définie par M moyennant les fonctions de codage et de décodage considérées.

Définition. Soient Σ un alphabet fini, $\flat \in \Sigma$ et M une machine de type $\mathsf{T}(\Sigma, \flat)$. Soient d'autre part α et β deux fonctions telles que

$$\mathrm{Prt}\alpha \subset \mathsf{T}(\Sigma, \flat) \text{ et } \mathrm{Dom}\beta \subset \mathsf{T}(\Sigma, \flat).$$

La relation composée $\alpha|M|\beta$ est appelée la *relation calculée par* M *moyennant la fonction de codage* α *et la fonction de décodage* β.

5.5.8. Exemple: machine d'addition

La machine M représentée dans la figure 5.24, de type $\mathsf{T}(\{0, 1\}, 0)$, calcule la fonction d'addition $F : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, telle que $F(m, n) = m + n$ quels que soient $m, n \in \mathbb{N}$, moyennant la fonction de codage

$$\alpha = \{(m, n) \mapsto \rhd 01^{m+1}01^{n+1}0 \mid (m, n) \in \mathbb{N} \times \mathbb{N}\}$$

et la fonction de décodage

$$\beta = \{x \rhd 01^{p+1}0y \mapsto p \mid x, y \in (0 \cup 1)^* \text{ et } p \in \mathbb{N}\}.$$

Par exemple, le couple d'entiers $(m, n) = (3, 2)$ est codé par l'état de ruban

$$\alpha(3, 2) = \rhd 0111101110.$$

Nous écrivons naturellement 0111101110 pour $\langle\!\langle\, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0 \,\rangle\!\rangle$. L'exécution de M à partir de cet état de ruban initial doit aboutir à un état de ruban final de la forme

$$x \rhd 01111110y = x \rhd 01^{m+n+1}0y.$$

Cet état de ruban, dans lequel x et y peuvent être des séquences quelconques, appartient au domaine de la fonction de décodage β et celle-ci en « extrait » le nombre $m + n = 5$. L'exécution de M à partir de l'état de ruban initial $\rhd 0111101110$ consiste à: déplacer la tête de lecture jusqu'au 0 situé entre les deux blocs de 1s; remplacer ce 0 par 1; ramener la tête de lecture à gauche; effacer deux 1s (les remplacer par des 0s).

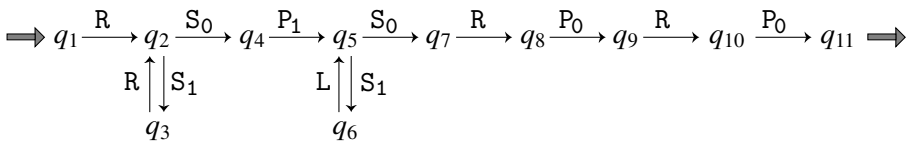


Figure 5.24

Une machine équivalente est représentée dans la figure 5.25. Elle est de type $(E, \mathrm{Reg}(\Omega))$ (5.2.10), où E est l'ensemble des états de rubans $\mathsf{T}(\{0, 1\}, 0)$ et Ω est le répertoire de relations $\{\mathtt{L}, \mathtt{R}, \mathtt{S}_0, \mathtt{S}_1, \mathtt{P}_0, \mathtt{P}_1\}$ dans E.

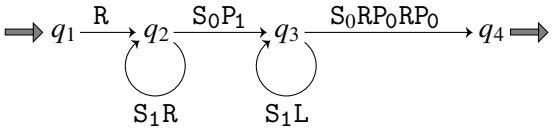


Figure 5.25

5.5.9. Exercice

Soient M la machine de la figure 5.24 et $m, n \in \mathbb{N}$. On désigne par E l'ensemble des états de ruban $\mathsf{T}(\{0, 1\}, 0)$. Donner une assertation de M avec les ensembles initiaux et finaux A_I, A_F suivants, qui sont des sous-ensembles à un seul élément de E (cf. 5.5.6):

$$A_I = \{w \in E \mid w = \rhd 01^{m+1}01^{n+1}0\};$$
$$A_F = \{w \in E \mid w = \rhd 01^{m+n+1}0\}.$$

D'après le théorème 5.3.3 et la formule 1.3.23(4), une telle assertation prouve que quels que soient les états de ruban $w, w' \in E$, on a:

$$\big(w = \rhd 01^{m+1}01^{n+1}0 \text{ et } w \mapsto w' \in |M|\big) \Rightarrow w' = \rhd 01^{m+n+1}0.$$

5.5.10. Exercice

Construire une machine M de type $\mathsf{T}(\{0, 1\}, 0)$ qui calcule la fonction F définie ci-dessous, moyennant les fonctions de codage et de décodage α et β de 5.5.8.

$$F : \mathbb{N} \times \mathbb{N} \to \mathbb{N}.$$
$$\forall a, b \in \mathbb{N} : \quad F(a, b) = a \dot{-} b = \begin{cases} 0 & \text{si } a < b; \\ a - b & \text{si } a \geq b. \end{cases}$$

Le nombre noté $a \dot{-} b$ est appelé la *différence positive* de a et b.

5.5.11. Exercice

Construire une machine de Turing M de type $\mathsf{T}(\{0, 1\}, 0)$ qui calcule la fonction de multiplication $F : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ telle que $F(m, n) = mn$ pour tout $(m, n) \in \mathbb{N} \times \mathbb{N}$, moyennant les fonctions de codage et de décodage α, β de 5.5.8.

Indications. Partant d'un état de ruban initial de la forme

$$\rhd 0\overbrace{11\cdots1}^{m+1}0\overbrace{11\cdots1}^{n+1}0,$$

la méthode consiste à décaler $m - 1$ fois, de n positions vers la droite, le deuxième bloc de 1s de cet état de ruban et à effacer les m premiers 1s du premier bloc, pour parvenir à l'état

de ruban

$$\triangleright\, 01\underbrace{\overbrace{0\cdots0}^{n}\cdots\overbrace{0\cdots0}^{n}}_{(m-1)n}\overbrace{11\cdots1}^{n+1}.$$

Cela revient à insérer $m-1$ fois la séquence 0^n juste avant le deuxième bloc de 1s, en décalant celui-ci. Ce faisant, on consomme les 1s du premier bloc pour savoir quand cette opération est finie. Ensuite, il n'y a plus qu'à remplacer tous les 0s insérés par des 1s et effacer encore un 1.

L'opération de décalage à droite d'un bloc 1^{n+1} de n positions (ou insertion de n zéros à gauche de ce bloc) s'effectue de la manière suivante. Partant d'un état de ruban de la forme

$$x\,\triangleright\, 0\overbrace{1111111111111}^{n+1}0$$

on commence par effacer le n-ième 1 et écrire un 1 à droite du $(n+1)$-ième, parvenant ainsi à l'état de ruban

$$x\,\triangleright\, 0\overbrace{1111111111}^{n-1}0110$$

dans lequel le bloc de 1s a été scindé en deux blocs séparés par un seul 0. Ensuite on répète l'opération: effacer le premier 1 du premier de ces deux blocs et écrire un 1 à droite du deuxième bloc jusqu'à ce que le premier bloc soit complètement effacé. Il l'est lorsque, en revenant de la droite, on rencontre deux zéros consécutifs à gauche du deuxième bloc de 1s.

5.5.12. Exercice

Soient Γ_1 et Γ_2 deux alphabets finis, $\flat_1 \in \Gamma_1$ et $\flat_2 \in \Gamma_2$. Une machine M du type produit $\mathsf{T}(\Gamma_1, \flat_1) \otimes \mathsf{T}(\Gamma_2, \flat_2)$ (5.4.1) est appelée une « machine de Turing à deux rubans ». Pour fixer les idées, nous considérons les alphabets $\Gamma_1 = \Gamma_2 = \{\mathtt{0}, \mathtt{a}, \mathtt{b}, \mathtt{c}\}$ avec $\flat_1 = \flat_2 = \mathtt{0}$. Un état de mémoire d'une telle machine est un couple (α_1, α_2) d'états de rubans tels que $\alpha_1 \in \mathsf{T}(\Gamma_1, \flat_1)$ et $\alpha_2 \in \mathsf{T}(\Gamma_2, \flat_2)$, par exemple le couple (α_1, α_2) représenté dans la figure 5.26, où toutes les cellules vides contiennent 0.

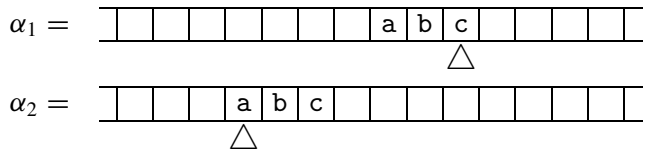


Figure 5.26

Posons $E_1 = \mathsf{T}(\Gamma_1, \flat_1)$, $E_2 = \mathsf{T}(\Gamma_2, \flat_2)$ et désignons par $\mathtt{R}_1$ (resp. $\mathtt{R}_2$), $\mathtt{L}_1$ (resp $\mathtt{L}_2$) les opérations R, L dans l'ensemble E_1 (resp. E_2). Désignons de même par $\mathtt{S}_{1x}$ et $\mathtt{P}_{1x}$ (resp. par $\mathtt{S}_{2x}$ et $\mathtt{P}_{2x}$), pour chaque $x \in \Gamma_1$ (resp. pour chaque $x \in \Gamma_2$), les opérations de lecture $\mathtt{S}_x$ et d'écriture $\mathtt{P}_x$ dans E_1 (resp. dans E_2). Par définition du type produit (5.4.1), les opérations de M sont les compositions parallèles d'opérations $\mathtt{R}_1 /\!/ \mathrm{Id}_{E_2}$, $\mathtt{L}_1 /\!/ \mathrm{Id}_{E_2}$, $\mathtt{S}_{1x} /\!/ \mathrm{Id}_{E_2}$ et $\mathtt{P}_{1x} /\!/ \mathrm{Id}_{E_2}$ (pour chaque $x \in \Gamma_1$), ainsi que $\mathrm{Id}_{E_1} /\!/ \mathtt{R}_2$, $\mathrm{Id}_{E_1} /\!/ \mathtt{S}_{2x}$ (pour chaque $x \in \Gamma_2$), etc. Nous les désignons simplement par $\mathtt{R}_1$, $\mathtt{R}_2$, etc. selon la convention usuelle (5.4.1).

On demande de montrer qu'une telle machine M peut être simulée par une machine de Turing M' à un seul ruban. Le ruban de M' sera un ruban à 5 « pistes », mais avec *une seule* tête de lecture-écriture qui « voit » les 5 pistes en même temps (figure 5.27). Le contenu des pistes 1 et 3 correspond à celui des deux rubans de la machine M. Les pistes 2 et 4 contiennent chacune un seul 1, dont la position correspond à celle des têtes de lecture/écriture des rubans

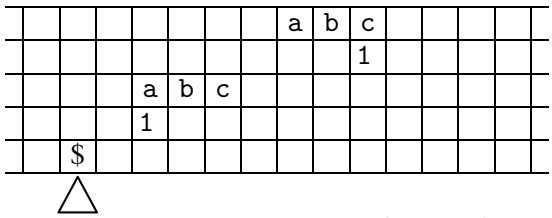


Figure 5.27

de M. La piste 5 contient un seul symbole \$. Une cellule du ruban de M' se compose des 5 cellules de pistes d'une même colonne, et leur contenu forme un seul symbole pour M'. Un symbole de ruban de M' est donc un 5-uple tel que

$$\begin{pmatrix} 0 \\ 0 \\ \mathtt{a} \\ 1 \\ 0 \end{pmatrix} \quad = \quad (0, 0, \mathtt{a}, 1, 0) \quad \text{en notation horizontale,}$$

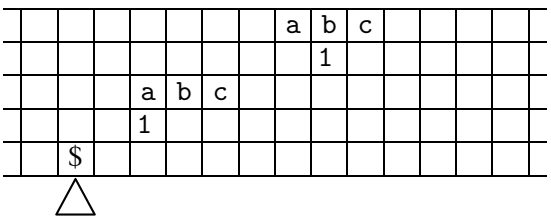
appartenant au produit cartésien

$$\Sigma = \Gamma_1 \times \{0, 1\} \times \Gamma_2 \times \{0, 1\} \times \{0, \$\}.$$

Cet ensemble Σ est l'alphabet de ruban de M'. Les opérations de lecture et d'écriture de M' sont donc de la forme $\mathtt{S}_{(0,0,\mathtt{a},1,0)}$ et $\mathtt{P}_{(0,0,\mathtt{a},1,0)}$, par exemple. L'élément blanc de Σ est le vecteur $\flat = (0, 0, 0, 0, 0)$.

Un état des deux rubans de M est simulé par un état du ruban de M' dans lequel la tête de lecture est sous la cellule contenant le symbole \$, celui-ci étant toujours *à gauche* de tous les 1s des pistes 2 et 4. Sa position exacte, à gauche, n'a pas d'importance par ailleurs.

Pour construire M', on commence par construire, pour chacune des opérations $\mathtt{L}_1, \mathtt{L}_2, \mathtt{R}_1, \mathtt{R}_2, \mathtt{S}_{1x}, \mathtt{S}_{2x}, \mathtt{P}_{1x}, \mathtt{P}_{2x}$ du répertoire de M, une machine $M'(\mathtt{L}_1)$, $M'(\mathtt{L}_2)$, etc. de type $\mathsf{T}(\Sigma, \flat)$ qui simule cette opération. Par exemple, partant de l'état de ruban initial de la figure 5.27, la machine $M'(\mathtt{L}_1)$ simulant l'opération $\mathtt{L}_1$ de M aboutira à l'état de ruban final



Ces machines $M'(\mathtt{L}_1)$, $M'(\mathtt{L}_2)$, etc. auront chacune un seul état initial q_0 et un seul état final q_n. Pour construire la machine M' simulant M, il suffit de remplacer chaque transition

$$p \xrightarrow{\sigma} q \qquad \big(\text{par exemple } p \xrightarrow{\mathtt{L}_1} q\big)$$

du diagramme de M par une copie

$$p \to \cdots \to q$$

du diagramme complet de la machine $M'(\sigma)$ qui simule l'opération σ, dans lequel on a renommé les états, p étant pris comme état initial et q comme état final.

On demande de construire les machines $M'(\mathtt{S}_{1\mathtt{a}})$, $M'(\mathtt{L}_2)$, $M'(\mathtt{P}_{2\mathtt{b}})$, $M'(\mathtt{R}_1)$.

Chapitre 6 Calculabilité

6.1 Fonctions calculables

6.1.1. Fonctions numériques

La notion de calculabilité peut être définie pour différents types de fonctions, où, par « type » d'une fonction, nous entendons le genre d'arguments qu'elle prend et le genre de valeurs qu'elle retourne. Nous ne parlons dans ce chapitre que des fonctions dont les arguments et les valeurs sont des entiers naturels. Nous les appelons les *fonctions numériques*, en prenant le mot « numérique » dans le sens restreint des nombres entiers naturels.

De façon précise, si $n \in \mathbb{N}$, nous appelons *fonction numérique* n-aire toute fonction F qui est une application d'un sous-ensemble de l'ensemble $\mathbb{N}^n = \mathbb{N} \times \cdots \times \mathbb{N}$ (n facteurs) dans $\mathbb{N}$.

L'ensemble $\mathbb{N}^n$ est l'ensemble des n-uples $(x_1, \ldots, x_n)$ d'entiers naturels. Il est convenu que $\mathbb{N}^1 = \mathbb{N}$ et que $\mathbb{N}^0$ est un ensemble à un seul élément, $\mathbb{N}^0 = \{()\}$, cet élément étant le 0-*uple* $()$.

Nous utilisons la notation $F : A \rightarrowtail\!\!\!\!\!\rightarrow B$ pour exprimer que F est une application d'un sous-ensemble de A dans B (on parle parfois d'une application partielle de A dans B). Cela signifie que F est une fonction telle que $\mathrm{Dom}\, F \subset A$ et $\mathrm{Prt}\, F \subset B$ (cf. 1.3.14). Une fonction numérique n-aire est donc une fonction F telle que

$$F : \mathbb{N}^n \rightarrowtail\!\!\!\!\!\rightarrow \mathbb{N}.$$

Dans le cas particulier où $\mathrm{Dom}\, F = \mathbb{N}^n$, donc où $F : \mathbb{N}^n \to \mathbb{N}$, on dit que F est une fonction numérique n-aire *totale*. Dans le cas contraire, c'est-à-dire si $\mathrm{Dom}\, F$ et un sous-ensemble propre de $\mathbb{N}^n$, on dit que F est une fonction numérique n-aire *partielle*. Par exemple, la fonction « soustraction »

$$F = \{(a, b) \mapsto a - b \mid (a, b) \in \mathbb{N}^2 \text{ et } a \geq b\}$$

est une fonction numérique binaire partielle.

6.1.2. Fonctions de codage et de décodage

Il y a plusieurs manières de définir les fonctions numériques calculables. Notre définition utilise les machines de Turing de type $\mathsf{T}(\{0, 1\}, 0)$ et les fonctions de codage et décodage que nous avons déjà introduites au §5.5.8. Pour tout $n \in \mathbb{N}$, notre *fonction de codage standard*

$$\alpha_n : \mathbb{N}^n \to \mathsf{T}(\{0, 1\}, 0)$$

est telle que, pour tout $(x_1, \ldots, x_n) \in \mathbb{N}^n$:

$$\begin{aligned} \alpha_n(x_1, \ldots, x_n) &= \rhd\, 01^{x_1+1}0 \cdots 01^{x_n+1} \\ &= \rhd \operatorname*{\mathsf{CAT}}_{k=1}^{n} \big(\langle\!\langle\, 0 \,\rangle\!\rangle \langle\!\langle\, 1 \,\rangle\!\rangle^{x_n+1} \big). \end{aligned}$$

La fonction de codage α du §5.5.8 est donc notre fonction de codage α_2. La fonction $\alpha_0 : \mathbb{N}^0 \to \mathsf{T}(\{0, 1\}, 0)$ fait correspondre à l'unique élément $()$ de $\mathbb{N}^0$ l'état de ruban $\alpha_0() = \rhd\, \Lambda$, totalement « blanc » (5.5.5).

Notre fonction de décodage standard (5.5.8)

$$\beta : \mathsf{T}(\{0, 1\}, 0) \twoheadrightarrow \mathbb{N}$$

est l'ensemble des couples (état de ruban, entier naturel) de la forme

$$u \rhd 01^{p+1}0v \mapsto p$$

où p est un entier naturel et u, v sont des séquences quelconques sur l'alphabet $\{0, 1\}$. Le domaine de cette fonction est l'ensemble des états de ruban de la forme $u \rhd 01^{p+1}0v$, ou plus exactement

$$u \rhd \langle\!\langle\, 0 \,\rangle\!\rangle \langle\!\langle\, 1 \,\rangle\!\rangle^{p+1} \langle\!\langle\, 0 \,\rangle\!\rangle v,$$

et, pour un tel état de ruban, β retourne le nombre p.

6.1.3. Machines de Turing numériques. L'ensemble MTN

Nous appelons *machine de Turing numérique* toute machine de Turing (M, I, F) de type $\mathsf{T}(\{0, 1\}, 0)$ qui satisfait aux trois conditions suivantes :

(1) M est déterministe.

(2) L'ensemble d'états Q^M est un intervalle $[0 \,..\, n]$ de $\mathbb{N}$ et l'on a

$$I = \{0\} \text{ et } F = \{n\}.$$

(3) Chaque état $k \in Q^M$ est l'état de départ ou l'état d'arrivée d'au moins une transition de M. On dit que M n'a pas d'état isolé.

Nous désignons par MTN l'ensemble des machines de Turing numériques.

Exemple. La machine M de la figure 5.24 est une machine de Turing numérique, si nous précisons que ses états $q_1, \ldots, q_{11}$ sont les entiers $0, \ldots, 10$ ($q_k = k - 1$). Nous n'avons pas précisé quels objets étaient ces états au §5.5.8, car cela n'avait pas d'importance. Il nous suffisait d'admettre qu'il s'agissait de 11 objets distincts. Maintenant nous précisons lesquels. Cette machine M est de type $\mathsf{T}(\{0, 1\}, 0)$. Elle est déterministe. Elle satisfait à la condition (2) et n'a pas d'état isolé.

6.1.4. Définition : fonctions numériques calculables

Soit $n \in \mathbb{N}$. Une fonction numérique $F : \mathbb{N}^n \twoheadrightarrow \mathbb{N}$ est dite *calculable* s'il existe une machine de Turing numérique M (6.1.3) qui calcule F moyennant les fonctions de codage et de décodage α_n et β, c'est-à-dire telle que :

$$F = \alpha_n |M| \beta.$$

Exemple. La fonction d'addition $F : \mathbb{N}^2 \to \mathbb{N}$ telle que $F(x_1, x_2) = x_1 + x_2$ est calculable. Nous avons vu en effet (5.5.8) que la machine M de la figure 5.24 calcule cette fonction moyennant les fonctions de codage et de décodage $\alpha = \alpha_2$ et β. Cette machine M appartient à l'ensemble MTN, en admettant que $q_1, \ldots, q_{11}$ sont les entiers $0, \ldots, 10$.

Remarques. On peut se demander si la notion de calculabilité, telle qu'elle vient d'être définie pour une fonction $F : \mathbb{N}^n \twoheadrightarrow \mathbb{N}$, n'est pas trop restrictive, du fait qu'elle n'exige pas seulement l'existence d'une machine de Turing M de type $\mathsf{T}(\{0, 1\}, 0)$ telle que $F = \alpha_n |M| \beta$, mais qu'elle exige l'existence d'une machine M appartenant une *classe particulière* de machines de Turing, à savoir la classe MTN, soumise aux restrictions 6.1.3(1) à 6.1.3(3).

On peut se demander notamment si la classe des fonctions calculables ne serait pas plus large si l'on admettait l'emploi de machines de Turing non déterministes. En fait, il n'en est

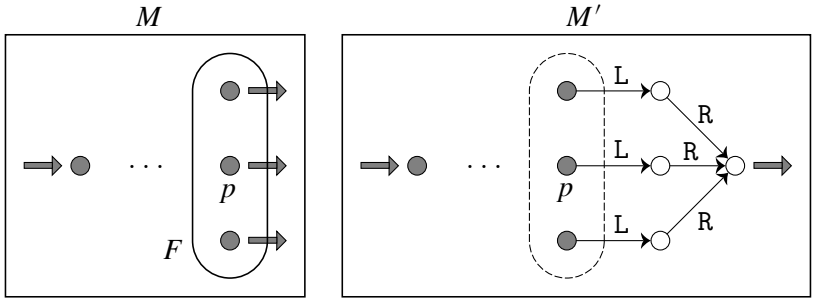


Figure 6.1

rien, mais cela est un théorème qui n'est pas évident et qui ne sera pas démontré dans ce cours (Voir Automates et calculabilité II).

Par contre il est évident que les restrictions 6.1.3(2) et 6.1.3(3) ne sont pas essentielles. Premièrement, on peut toujours choisir comme ensemble d'états d'un accepteur fini un ensemble d'entiers naturels $[0 \,..\, p]$. En outre, s'il s'agit d'un accepteur fini déterministe (4.4.1), possédant donc un état initial et un seul, on peut toujours prendre 0 comme état initial. Deuxièmement, la figure 6.1 montre que pour toute machine de Turing déterministe M on peut construire une machine déterministe équivalente M' ayant un seul état final. La machine M' est déterministe en vertu du fait que M est déterministe, donc qu'il n'y a pas de transition partant d'un état $p \in F$ dans M (5.4.5(3)). Cette construction utilise le fait que la relation composée $\mathtt{LR}$ est la relation identique de l'ensemble des états de ruban, donc que $|M'| = |M|\mathtt{LR} = |M|$.

6.1.5. Machines de Turing déterministes

Puisque la définition des fonctions numériques calculables fait intervenir les machines de Turing *déterministes* de type $\mathsf{T}(\{\mathtt{0}, \mathtt{1}\}, \mathtt{0})$, il y a lieu de noter ce qui caractérise les machines de Turing déterministes en général. Soient donc Σ un alphabet fini et $b \in \Sigma$. Une machine (M, I, F) de type $\mathsf{T}(\Sigma, b)$ est déterministe si elle satisfait aux conditions du §5.4.5, définissant une machine déterministe en général. Comme toutes les relations du répertoire ($\mathtt{R}$, $\mathtt{L}$, $\mathtt{S}_a$ $(a \in \Sigma)$, $\mathtt{P}_a$ $(a \in \Sigma)$) de ce type de machine sont des fonctions, la condition 5.4.5(1) est satisfaite pour toute machine de ce type, déterministe ou non.

La condition (2) est que pour deux transitions (p, σ, q), (p', σ', q') partant du même état p, les domaines des fonctions σ et σ' sont disjoints. Ni l'une ni l'autre de ces deux fonctions σ, σ' ne peut donc être la fonction $\mathtt{L}$ ou la fonction $\mathtt{R}$ ou une fonction $\mathtt{P}_a$ $(a \in \Sigma)$ car toutes ces fonctions ont pour domaine l'ensemble $E = \mathsf{T}(\Sigma, b)$ de tous les états de ruban (5.5.4). Pour une machine M de type $\mathsf{T}(\Sigma, b)$, la condition (2) de 5.4.5 peut donc s'exprimer comme suit: si (p, σ, q) et (p, σ', q') sont deux transitions de M partant du même état p, on a

$$\sigma = \mathtt{S}_a \text{ et } \sigma' = \mathtt{S}_{a'}$$

pour deux éléments $a \neq a'$ de Σ. Pour une machine de type $\mathsf{T}(\{\mathtt{0}, \mathtt{1}\}, \mathtt{0})$, l'une des relations σ, σ' est $\mathtt{S}_0$, l'autre est $\mathtt{S}_1$ (voir l'exemple de la figure 5.24).

6.1.6. Exercice

Montrer que la fonction numérique partielle (6.1.1) $F : \mathbb{N}^2 \dashrightarrow \mathbb{N}$ de soustraction

$$F = \{(a, b) \mapsto a - b \mid (a, b) \in \mathbb{N}^2 \text{ et } a \geq b\} \tag{1}$$

est calculable en construisant une machine de Turing appropriée selon la définition 6.1.4. On partira de la solution de l'exercice 5.5.10, car la fonction (1) est la restriction au sous-

ensemble $\{(a, b) \in \mathbb{N}^2 \mid a \geq b\}$ de $\mathbb{N}^2$ de la fonction « différence positive » considérée dans l'exercice mentionné, laquelle est une fonction numérique totale $\mathbb{N}^2 \to \mathbb{N}$.

6.2 Une fonction non-calculable

6.2.1. Résumé

Le but de cette section est de donner l'exemple d'une fonction $F : \mathbb{N} \to \mathbb{N}$ qui n'est pas calculable. Nous commencerons par définir cette fonction F, puis nous démontrerons qu'elle n'est pas calculable. Cela prouvera l'existence de fonctions non-calculables.

La définition de cette fonction ne se formule pas en une ligne. Elle nécessite quelques développements préalables, dont nous donnons ici un résumé.

Les restrictions auxquelles sont soumises les machines de la classe MTN (6.1.3) font que cet ensemble de machines est *dénombrable*. Cela veut dire qu'il existe une bijection $\mu : \mathbb{N} \to \text{MTN}$. La plus grande partie de notre travail préalable va être de définir une telle bijection. À chaque entier $n \in \mathbb{N}$, cette fonction μ associera une machine de Turing numérique bien déterminée $\mu(n)$, que nous noterons aussi μ_n et que appellerons la *n-ième machine de Turing numérique*. Par définition d'une application bijective, pour chaque machine $M \in \text{MTN}$, il existera un $n \in \mathbb{N}$ et un seul tel que $M = \mu_n$, et cet entier n sera appelé le *numéro* de la machine numérique M.

La fonction $F : \mathbb{N} \to \mathbb{N}$ dont nous montrerons qu'elle n'est pas calculable sera la fonction

$$F = \big(\lambda n \in \mathbb{N} : \textbf{\textit{Si}}\big(\rhd 01^{n+1}0 \in \text{Dom}|\mu_n|, 1, 0\big)\big),$$

autrement dit la fonction F de domaine $\mathbb{N}$ telle que, pour tout $n \in \mathbb{N}$:

$$F(n) = \begin{cases} 1 & \text{si } \rhd 01^{n+1}0 \in \text{Dom}|\mu_n|; \\ 0 & \text{si } \rhd 01^{n+1}0 \notin \text{Dom}|\mu_n|. \end{cases} \tag{1}$$

Voyons en quoi consiste le « calcul » de la valeur $F(n)$ correspondant à un $n \in \mathbb{N}$. Premièrement, on prend la n-ième machine de Turing numérique, μ_n, telle qu'elle est définie par la fonction $\mu : \mathbb{N} \to \text{MTN}$. La relation $|\mu_n|$ définie par cette machine est une relation dans l'ensemble $\mathsf{T}(\{0, 1\}, 0)$ des états de ruban d'alphabet $\{0, 1\}$ avec élément blanc 0. On considère l'état de ruban $\rhd 01^{n+1}0$. De deux choses l'une: ou bien cet état de ruban appartient au domaine de la relation $|\mu_n|$ ou bien il n'appartient pas à ce domaine (principe du tiers exclu). Dans le premier cas, on retourne la valeur 1. Dans le second, on retourne la valeur 0.

Cette description du « procédé de calcul $F(n)$ » ne doit pas faire illusion. Il ne s'agit pas d'un procédé algorithmique, exécutable par une machine, sinon la fonction F serait calculable — et nous montrerons qu'elle ne l'est pas. Notre description du calcul de $F(n)$ pour un $n \in \mathbb{N}$ n'est qu'un commentaire en français de la formule (1). Dans ce « procédé de calcul » de $F(n)$, il y a une opération qui est bel et bien algorithmique, à savoir la détermination de la machine de Turing μ_n. Cela apparaîtra clairement lorsque nous aurons défini la fonction μ. L'opération qui n'est pas mécanisable, dans le calcul de $F(n)$, est de déterminer si l'état de ruban $\rhd 01^{n+1}0$ appartient au domaine de la relation $|\mu_n|$ ou non. Le principe du tiers exclu nous dit que l'un des deux est vrai, et c'est pourquoi $F(n)$ a une valeur bien définie, mathématiquement, même si l'on ne connaît pas cette valeur. Un être humain, utilisant les ressources de son intelligence, sera peut-être capable de déterminer, pour un $n \in \mathbb{N}$ donné, si $\rhd 01^{n+1}0 \in \text{Dom}|\mu_n|$ ou non. Il découvrira peut-être un algorithme permettant de répondre à cette question pour *certains* entiers $n \in \mathbb{N}$. Ce qui n'existe pas, c'est un algorithme permettant de répondre à cette question pour *n'importe quel* $n \in \mathbb{N}$.

6.2.2. Définition d'une bijection $\mu : \mathbb{N} \to \text{MTN}$

Notre but est de définir une fonction μ qui associe à chaque entier naturel une machine de Turing numérique bien déterminée μ_n, de telle manière que les conditions suivantes soient vérifiées: (a) les machines μ_n, $\mu_{n'}$ correspondant à des entiers $n \neq n'$ sont différentes ($\mu_n \neq \mu_{n'}$), autrement dit μ est injective; (b) pour toute machine de Turing $M \in \text{MTN}$, il existe un $n \in \mathbb{N}$ tel que $M = \mu_n$, autrement dit μ est une application surjective de $\mathbb{N}$ dans MTN. Une telle fonction peut être appelée une *numérotation* des machines de Turing numériques. L'ensemble MTN peut être présenté comme une énumération

$$\text{MTN} = \{\mu_0, \mu_1, \mu_2, \ldots\}$$

Pour définir cette fonction μ, nous procéderons en plusieurs étapes, où nous définirons des fonctions auxiliaires utilisées dans la définition de μ. La définition de μ sera la dernière d'une série de définitions de fonctions. Nous résumons d'abord cette série, avant d'entrer dans les détails.

Un ensemble joue un rôle important dans ces définitions. C'est l'ensemble des transitions de toutes les machines de Turing numériques, à savoir l'ensemble de tous les triplets de la forme

$$(1) \qquad (p, \mathtt{L}, q), \quad (p, \mathtt{R}, q), \quad (p, \mathtt{S}_0, q), \quad (p, \mathtt{S}_1, q), \quad (p, \mathtt{P}_0, q), \quad (p, \mathtt{P}_1, q),$$

où p et q sont des entiers naturels quelconques. Nous désignons cet ensemble par Tr(MTN). Sa définition peut s'exprimer simplement par

$$\text{Tr(MTN)} = \mathbb{N} \times \{\mathtt{L}, \mathtt{R}, \mathtt{S}_0, \mathtt{S}_1, \mathtt{P}_0, \mathtt{P}_1\} \times \mathbb{N}.$$

Cet ensemble est évidemment un ensemble infini.

Pour toute machine $M \in \text{MTN}$, l'ensemble T^M des transitions de M est un sous-ensemble *fini* de l'ensemble infini Tr(MTN). Mais ce n'est pas n'importe que sous-ensemble fini, car il est soumis aux conditions de 6.1.3: tous les états p, q qui figurent dans les transitions (p, σ, q) de cet ensemble doivent former un intervalle $[0 \,..\, n]$ de $\mathbb{N}$ et, en prenant 0 comme état initial et n comme état final, on doit obtenir une machine déterministe.

L'étape principale, en vue de la définition de la fonction μ, sera la définition d'une fonction Δ qui associe à chaque sous-ensemble fini Z de l'ensemble Tr(MTN), un entier $\Delta(Z) \in \mathbb{N}$. Cette fonction sera une application bijective de l'ensemble des parties finies (sous-ensembles finis) de l'ensemble Tr(MTN) dans $\mathbb{N}$. En désignant par $\mathfrak{P}_\mathrm{f}(X)$ l'ensemble des parties finies d'un ensemble X, nous aurons donc

$$\Delta : \mathfrak{P}_\mathrm{f}(\text{Tr(MTN)}) \xrightarrow{bij} \mathbb{N}.$$

Nous utiliserons aussi la fonction réciproque, Δ^{-1}, qui sera naturellement une bijection de $\mathbb{N}$ dans $\mathfrak{P}_\mathrm{f}(\text{Tr(MTN)})$. Pour chaque $k \in \mathbb{N}$, $\Delta^{-1}(k)$ est un sous-ensemble fini de Tr(MTN).

Cette fonction Δ sera définie plus loin (6.2.5). Dans le présent paragraphe, nous définissons la fonction μ en supposant que la fonction Δ est donnée. La définition de μ procède comme suit. On considère l'ensemble des entiers naturels de la forme $\Delta(T^M)$, correspondant aux ensembles de transitions T^M des machines $M \in \text{MTN}$, autrement dit l'ensemble

$$(2) \qquad A = \{\Delta(T^M) \mid M \in \text{MTN}\}.$$

Par définition, A est un sous-ensemble de $\mathbb{N}$ et l'on a

$$a \in A \Leftrightarrow \exists M \in \text{MTN} : a = \Delta(T^M).$$

Cet ensemble A est évidemment un sous-ensemble infini de $\mathbb{N}$, car l'ensemble MTN est lui-même infini et Δ est injective.

Les éléments de n'importe quel sous-ensemble infini A de $\mathbb{N}$ peuvent être numérotés par ordre de grandeur :

$$A = \{a_0, a_1, a_2, \ldots\} \quad \text{avec} \quad a_0 < a_1 < a_2 < \ldots$$

En termes plus précis, étant donné un ensemble infini $A \subset \mathbb{N}$, il existe une suite $(a_n)_{n \in \mathbb{N}}$ et une seule telle que $A = \{a_n \mid n \in \mathbb{N}\}$ et $a_n < a_{n+1}$ pour tout n. Cette suite est définie simplement par récurrence comme suit : a_0 est le plus petit élément de A; pour tout $n \in \mathbb{N}$, a_{n+1} est le plus petit élément de $A - \{a_0, \ldots, a_n\}$.

Nous supposons donnée cette numérotation des éléments de l'ensemble A défini par (2). Les entiers $a_0, a_1, a_2, \ldots$ sont donc tous les entiers naturels, par ordre de grandeur croissante, qui sont l'image par la fonction Δ de l'ensemble des transitions T^M d'une machine de Turing numérique M. Inversement, les sous-ensembles $\Delta^{-1}(a_0)$, $\Delta^{-1}(a_1)$, $\Delta^{-1}(a_2)$, ... de l'ensemble Tr(MTN) sont les ensembles de transitions T^M de toutes les machines de Turing numériques M.

Pour définir la fonction $\mu : \mathbb{N} \to \text{MTN}$, il n'y a plus qu'à définir μ_n, pour tout $n \in \mathbb{N}$, comme étant la machine $M \in \text{MTN}$ telle que

$$T^M = \Delta^{-1}(a_n).$$

En posant cette définition, nous utilisons le fait qu'une machine de Turing numérique M est entièrement déterminée par la donnée de son ensemble de transitions T^M. En effet, en vertu de la condition 6.1.3(3) à laquelle une telle machine est soumise, son ensemble d'états Q^M est l'ensemble des états de départ ou d'arrivée des transitions de M. Il est donc déterminé par la donnée de T^M. En outre, cet ensemble d'états Q^M est un intervalle $[0 \,..\, n]$ et l'état initial de M est 0, l'état final est n. M est donc entièrement déterminée par T^M.

La fonction μ sera donc complètement définie lorsque nous aurons défini la fonction Δ. Cela sera fait dans les paragraphes qui suivent. Avant cela, admettons que Δ soit définie et, pour clarifier la définition de μ, précisons comment, pratiquement, on détermine la machine μ_n correspondant à un entier n donné.

Pour déterminer la machine μ_n, il faut en fait déterminer les $n+1$ machines $\mu_0, \ldots, \mu_n$. Il faut en effet parcourir les entiers 0, 1, 2, ..., en partant de 0, jusqu'à ce que l'on ait trouvé les $n+1$ plus petits entiers $a_0, a_1, \ldots, a_n$ tels que, pour $i = 0, \ldots, n$, l'ensemble $\Delta^{-1}(a_i)$ soit l'ensemble des transitions d'une machine de Turing numérique — cette machine est précisément la machine μ_i. Partant de $k = 0$, on teste donc l'ensemble de transitions $\Delta^{-1}(k)$, pour savoir s'il satisfait aux conditions qui caractérisent une machine de Turing numérique, et l'on incrémente k jusqu'à ce que l'on ait trouvé $n+1$ fois un ensemble $\Delta^{-1}(k)$ qui est l'ensemble des transitions d'une machine de Turing numérique.

Les paragraphes suivants sont consacrés à la définition de la fonction Δ.

6.2.3. Une bijection $\mathbb{N}^2 \to \mathbb{N}$

Pour définir la fonction Δ dont il est question au paragraphe précédent, qui associe à chaque sous-ensemble fini Z de l'ensemble Tr(MTN) un entier $\Delta(Z) \in \mathbb{N}$ nous définirons d'abord une bijection ν de l'ensemble

$$\text{Tr(MTN)} = \mathbb{N} \times \{\mathtt{L}, \mathtt{R}, \mathtt{S}_0, \mathtt{S}_1, \mathtt{P}_0, \mathtt{P}_1\} \times \mathbb{N}$$

dans l'ensemble $\mathbb{N}$, associant à toute transition $t = (p, \sigma, q) \in \text{Tr(MTN)}$ un entier $\nu(t) \in \mathbb{N}$. Pour cela, nous commençons par choisir une bijection

$$g : \mathbb{N} \times \mathbb{N} \to \mathbb{N},$$

puisque $\mathbb{N} \times \mathbb{N}$ est l'ensemble des couples (état de départ,état d'arrivée) des transitions $t \in \mathrm{Tr}(\mathrm{MTN})$. Il y a beaucoup de manières de définir une telle fonction g. La figure 6.2 en indique deux.

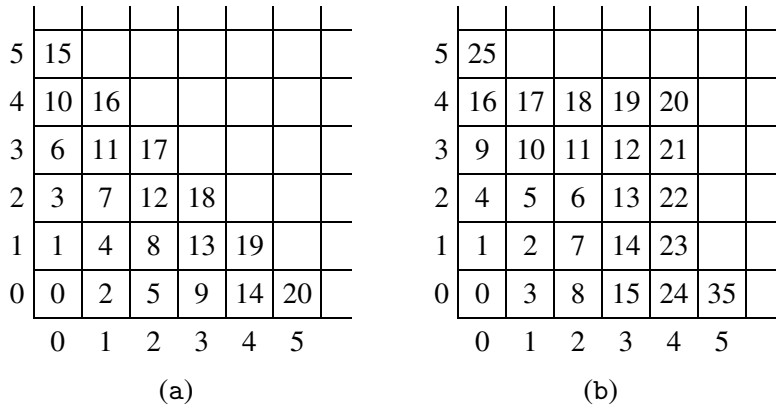


Figure 6.2
Bijections $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$

L'ensemble $\mathbb{N} \times \mathbb{N}$ est représenté par un quart de plan divisé en cellules repérées chacune par un couple de coordonnées entières. Pour chaque couple $(x, y) \in \mathbb{N} \times \mathbb{N}$, $g(x, y)$ est l'entier qui est noté dans la cellule de coordonnées (x, y), c'est-à-dire celle qui se trouve dans la colonne x et la ligne y. Dans le premier tableau, on a par exemple $g(3, 2) = 18$, dans le deuxième $g(3, 2) = 13$. Définir une telle fonction revient à donner une manière de parcourir les cellules du tableau de telle manière que toute cellule soit visitée une fois et une seule dans le parcours. Des parcours tels que ceux de la figure 6.3 ne conviennent évidemment pas, puisqu'ils ne visitent pas toutes les cellules du plan.

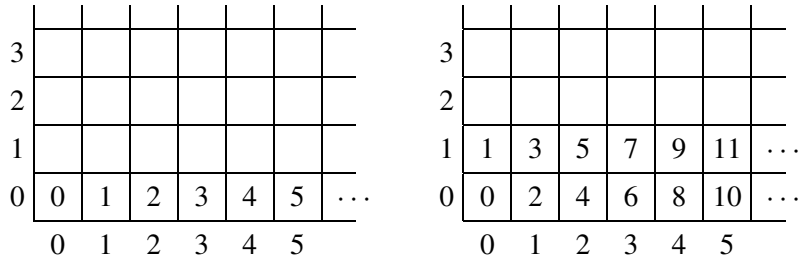


Figure 6.3

Nous choisissons la fonction $g : \mathbb{N}^2 \to \mathbb{N}$ décrite par le tableau (b) de la figure 6.2. Le principe de la construction de ce tableau est que, pour chaque $n \in \mathbb{N}$, les entiers naturels appartenant à l'intervalle $[n^2 \,..\, (n+1)^2 - 1] = [n^2 \,..\, n^2 + 2n]$ sont disposés dans la n-ième ligne et dans la n-ième colonne du tableau comme ceci :

n	n^2	n^2+1	n^2+2	$\ldots$	n^2+n
					n^2+n+1
$\vdots$					$\vdots$
0					n^2+2n
	0			$\ldots$	n

Le lecteur vérifiera que la fonction g peut être définie précisément par la formule suivante, valable quels que soient $x, y \in \mathbb{N}$:

$$g(x, y) = \begin{cases} y^2 + x & \text{si } x \leq y\ ; \\ x^2 + 2x - y & \text{si } y < x. \end{cases} \tag{1}$$

Il vérifiera aussi que les fonctions $\pi_1, \pi_2 : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ qui retournent, inversement, pour chaque $m \in \mathbb{N}$ le couple de coordonnées $\pi_1(m) = x$ et $\pi_2(m) = y$ de la cellule dans laquelle se trouve le nombre m, sont définies précisément par les formules suivantes, dans lesquelles $\mathsf{r}(m)$ désigne, pour tout $m \in \mathbb{N}$, la « racine carrée entière » de m, c'est-à-dire le plus grand entier $n \in \mathbb{N}$ tels que $n^2 \leq m$:

$$\begin{aligned} \pi_1(m) &= \begin{cases} m - \mathsf{r}(m)^2 & \text{si } m \leq \mathsf{r}(m)^2 + \mathsf{r}(m); \\ \mathsf{r}(m) & \text{si } m > \mathsf{r}(m)^2 + \mathsf{r}(m). \end{cases} \\ \pi_2(m) &= \begin{cases} \mathsf{r}(m) & \text{si } m \leq \mathsf{r}(m)^2 + \mathsf{r}(m); \\ \mathsf{r}(m)^2 + 2\mathsf{r}(m) - m & \text{si } m > \mathsf{r}(m)^2 + \mathsf{r}(m). \end{cases} \end{aligned} \tag{2}$$

Une vérification des formules (1) et (2) consiste à en démontrer que

$$\pi_1(g(x, y)) = x; \quad \pi_2(g(x, y)) = y; \quad g(\pi_1(m), \pi_2(m)) = m. \tag{3}$$

Ces égalités se démontrent par disjonction des cas qui interviennent dans (1) et (2).

6.2.4. Numérotation des transitions des machines numériques

Nous définissons maintenant une bijection $\nu : \mathrm{Tr}(\mathrm{MTN}) \to \mathbb{N}$ qui numérote toutes les transitions des machines des machines de Turing numériques. Pour cela, pour chaque couple (p, q) d'entiers naturels, nous imaginons les six transitions $(p, \mathtt{L}, q)$, $(p, \mathtt{R}, q)$, $(p, \mathtt{S}_0, q)$, $(p, \mathtt{S}_1, q)$, $(p, \mathtt{P}_0, q)$, $(p, \mathtt{P}_1, q)$ placées dans la cellule de coordonnées (p, q) de la figure 6.2 (b), comme cela est montré dans la figure 6.4.

2	**4** (0, L, 2) (0, R, 2) (0, $\mathtt{S}_0$, 2) (0, $\mathtt{S}_1$, 2) (0, $\mathtt{P}_0$, 2) (0, $\mathtt{P}_1$, 2)	**5** (1, L, 2) (1, R, 2) (1, $\mathtt{S}_0$, 2) (1, $\mathtt{S}_1$, 2) (1, $\mathtt{P}_0$, 2) (1, $\mathtt{P}_1$, 2)	**6** (2, L, 2) (2, R, 2) (2, $\mathtt{S}_0$, 2) (2, $\mathtt{S}_1$, 2) (2, $\mathtt{P}_0$, 2) (2, $\mathtt{P}_1$, 2)	**13** (3, L, 2) (3, R, 2) (3, $\mathtt{S}_0$, 2) (3, $\mathtt{S}_1$, 2) (3, $\mathtt{P}_0$, 2) (3, $\mathtt{P}_1$, 2)
1	**1** (0, L, 1) (0, R, 1) (0, $\mathtt{S}_0$, 1) (0, $\mathtt{S}_1$, 1) (0, $\mathtt{P}_0$, 1) (0, $\mathtt{P}_1$, 1)	**2** (1, L, 1) (1, R, 1) (1, $\mathtt{S}_0$, 1) (1, $\mathtt{S}_1$, 1) (1, $\mathtt{P}_0$, 1) (1, $\mathtt{P}_1$, 1)	**7** (2, L, 1) (2, R, 1) (2, $\mathtt{S}_0$, 1) (2, $\mathtt{S}_1$, 1) (2, $\mathtt{P}_0$, 1) (2, $\mathtt{P}_1$, 1)	**14** (3, L, 1) (3, R, 1) (3, $\mathtt{S}_0$, 1) (3, $\mathtt{S}_1$, 1) (3, $\mathtt{P}_0$, 1) (3, $\mathtt{P}_1$, 1)
0	**0** (0, L, 0) (0, R, 0) (0, $\mathtt{S}_0$, 0) (0, $\mathtt{S}_1$, 0) (0, $\mathtt{P}_0$, 0) (0, $\mathtt{P}_1$, 0)	**3** (1, L, 0) (1, R, 0) (1, $\mathtt{S}_0$, 0) (1, $\mathtt{S}_1$, 0) (1, $\mathtt{P}_0$, 0) (1, $\mathtt{P}_1$, 0)	**8** (2, L, 0) (2, R, 0) (2, $\mathtt{S}_0$, 0) (2, $\mathtt{S}_1$, 0) (2, $\mathtt{P}_0$, 0) (2, $\mathtt{P}_1$, 0)	**15** (3, L, 0) (3, R, 0) (3, $\mathtt{S}_0$, 0) (3, $\mathtt{S}_1$, 0) (3, $\mathtt{P}_0$, 0) (3, $\mathtt{P}_1$, 0)
	0	1	2	3

Figure 6.4

Les cellules de ce tableau de l'ensemble Tr(MTN) sont numérotées selon le schéma de la figure 6.2 (b), le numéro de chaque cellule étant noté en haut à gauche. Les six transitions de la cellule 0 ((0, L, 0) à (0, P_1, 0)) sont numérotées de 0 à 5. Les six transitions de la cellule 1 ((0, L, 1) à (0, P_1, 1)) sont numérotées de 6 à 11. Celles de la cellule 2 sont numérotées de 12 à 17, et ainsi de suite : les six transitions de la cellule k reçoivent les numéros $6k$ à $6k + 5$. Chaque transition t de l'ensemble Tr(MTN) reçoit ainsi un numéro $\nu(t) \in \mathbb{N}$. Par exemple, le numéro de la transition $t = (3, S_1, 1)$ est $\nu(t) = 6 \cdot 14 + 3 = 87$.

L'application ν : Tr(MTN) $\to \mathbb{N}$ ainsi définie est bijective, autrement dit, pour tout entier $n \in \mathbb{N}$, il existe une transition $t \in$ Tr(MTN) et une seule telle que $\nu(t) = n$. Cela provient de ce que tout entier $n \in \mathbb{N}$ peut s'écrire d'une manière et d'une seule sous la forme $n = 6k + r$ avec $k \in \mathbb{N}$ et $r \in [0 \,..\, 5]$: k est le quotient ($k = n$ div 6), r est le reste ($r = n$ mod 6) de la division de n par 6.

Nous pouvons donner une formule pour cette fonction ν, en désignant les opérations L, R, S_0, S_1, P_0, P_1 respectivement par op_0, op_1, op_2, op_3, op_4, op_5. Alors, quels que soient $p, q \in \mathbb{N}$ et $k \in [0 \,..\, 5]$, le numéro de la transition (p, op_k, q) est donné par la formule

$$\nu(p, op_k, q) = 6g(p, q) + k,$$

où g est la bijection $\mathbb{N}^2 \to \mathbb{N}$ définie au §6.2.3.

La fonction inverse de ν est la fonction $\tau : \mathbb{N} \to$ Tr(MTN) qui retourne, pour chaque $n \in \mathbb{N}$, la transition t telle que $\nu(t) = n$. Cette fonction τ peut être définie par la formule

$$\tau(n) = \big(\pi_1(n \text{ div } 6), op_{n \bmod 6}, \pi_2(n \text{ div } 6)\big),$$

où π_1 et π_2 sont les fonctions $\mathbb{N} \to \mathbb{N}$ définies au §6.2.3.

6.2.5. Une bijection $\Delta : \mathfrak{P}_f(\text{Tr(MTN)}) \to \mathbb{N}$

Pour définir une application bijective Δ de l'ensemble des parties finies de Tr(MTN) dans $\mathbb{N}$, nous posons, pour tout ensemble fini $Z \subset$ Tr(MTN) :

$$\Delta(Z) = \sum_{t \in Z} 2^{\nu(t)}, \tag{1}$$

Par exemple, pour l'ensemble de transitions

$$Z = \{(0, S_0, 0), (0, S_1, 1), (1, P_0, 2)\}, \tag{2}$$

on a

$$\Delta(Z) = 2^{\nu(0,S_0,0)} + 2^{\nu(0,S_1,1)} + 2^{\nu(1,P_0,2)} = 2^2 + 2^9 + 2^{34}.$$

Dans le système de numération binaire, ce nombre s'écrit

$$10000000000000000000000000001000000100$$

avec 34 chiffres après le premier 1 et 9 chiffres après le deuxième.

Cet exemple montre immédiatement que la fonction Δ définie par (1) est une bijection de l'ensemble des parties finies de Tr(MTN) dans $\mathbb{N}$. Premièrement, cette fonction est injective. Pour deux sous-ensembles finis $Z \neq Z'$ de Tr(MTN), les nombres $\Delta(Z)$ et $\Delta(Z')$ sont différents car leurs développements en base 2 donc différents. Il y a en effet une transition t qui appartient à l'un des ensembles Z, Z' et qui n'appartient pas à l'autre. Dans l'un des développements en base 2 des nombres $\Delta(Z)$, $\Delta(Z')$, et dans l'un des deux seulement, il y a un chiffre 1 en position $\nu(t) + 1$.

Montrons encore que cette fonction est une application surjective de l'ensemble des parties finies de Tr(MTN) dans $\mathbb{N}$, à savoir que, pour tout $n \in \mathbb{N}$, il existe un ensemble fini

$Z \subset \mathrm{Tr(MTN)}$ tel que $n = \Delta(Z)$. Considérons un $n \in \mathbb{N}$ quelconque. Ce nombre possède un développement en base 2

$$n = \sum_{k=0}^{r} a_k 2^k,$$

avec $a_k = 0$ ou $a_k = 1$ pour $k = 0, \ldots, r$. Pour chacun des entiers $k = 0, \ldots, r$ il existe une transition $t \in \mathrm{Tr(MTN)}$ (et une seule) telle que $k = \nu(t)$. En prenant pour Z l'ensemble des transitions $t \in \mathrm{Tr(MTN)}$ telles que $\nu(t) \in [0\,..\,r]$ et $a_{\nu(t)} = 1$, on obtient pour Z un sous-ensemble fini de Tr(MTN) et l'on peut écrire

$$n = \sum_{k=0}^{r} a_k 2^k = \sum_{t \in Z} 2^{\nu(t)} = \Delta(Z).$$

6.2.6. Démonstration de la non-calculabilité de F

Nous allons démontrer maintenant que la fonction $F : \mathbb{N} \to \mathbb{N}$ (6.2.1) définie par

$$\forall n \in \mathbb{N}: \quad F(n) = \begin{cases} 1 & \text{si } \rhd 01^{n+1}0 \in \mathrm{Dom}|\mu_n|; \\ 0 & \text{si } \rhd 01^{n+1}0 \notin \mathrm{Dom}|\mu_n| \end{cases} \tag{1}$$

n'est pas calculable. Nous le ferons par réduction à l'absurde, en utilisant le théorème auxiliaire suivant.

Théorème 1. Pour toute fonction calculable $F : \mathbb{N} \to \mathbb{N}$ telle que $\mathrm{Prt}\,F = \{0, 1\}$, il existe une machine $M' \in \mathrm{MTN}$ telle que, pour tout $n \in \mathbb{N}$:

$$F(n) = \begin{cases} 0 & \text{si } \rhd 01^{n+1}0 \in \mathrm{Dom}|M'|; \\ 1 & \text{si } \rhd 01^{n+1}0 \notin \mathrm{Dom}|M'|. \end{cases} \tag{3}$$

Démonstration. L'idée de la démonstration est donnée dans la figure 6.5. On suppose que $F : \mathbb{N} \to \mathbb{N}$ est une fonction calculable quelconque telle que $\mathrm{Prt}\,F = \{0, 1\}$. Il existe donc une machine $M \in \mathrm{MTN}$ telle que $F = \alpha_1|M|\beta$. Une telle machine M, munie d'une assertation, est représentée dans la figure 6.5. L'ensemble d'états Q^M est un intervalle $[0\,..\,p]$ de $\mathbb{N}$, 0 est l'état initial, p l'état final. La machine M' est construite simplement en ajoutant à M les transitions $(p, \mathtt{R}, p + 1)$, $(p + 1, \mathtt{R}, p + 2)$, $(p + 2, \mathtt{S}_0, p + 3)$ et en prenant $p + 3$ comme état final. Il est clair que $M' \in \mathrm{MTN}$. Un état de ruban w qui satisfait l'assertion de $p + 2$ appartient au domaine de l'opération $\mathtt{S}_0$ si et seulement si $F(n) = 0$. Donc un état de ruban initial $w = \rhd 01^{n+1}0$ appartient au domaine de $|M'|$ si et seulement si $F(n) = 0$. Cette machine M' vérifie donc (3). Une démonstration plus détaillée de ce théorème est donnée dans [A].

Théorème 2. La fonction $F : \mathbb{N} \to \mathbb{N}$ définie par (1) n'est pas calculable.

Démonstration. Supposons que F soit calculable. D'après le théorème 1, il existe une machine $M' \in \mathrm{MTN}$ qui vérifie (3) pour cette même fonction F. Soit k le numéro de cette machine M' au sens de la bijection $\mu : \mathbb{N} \to \mathrm{MTN}$, autrement dit soit $M' = \mu_k$. On peut remplacer M' par μ_k dans (3). Donc, pour tout $n \in \mathbb{N}$, on a

$$F(n) = \begin{cases} 0 & \text{si } \rhd 01^{n+1}0 \in \mathrm{Dom}|\mu_k|; \\ 1 & \text{si } \rhd 01^{n+1}0 \notin \mathrm{Dom}|\mu_k|. \end{cases} \tag{4}$$

Les égalités (1) et (4) sont vraies pour tout entier n. Or pour $n = k$ elles se contredisent.

6.2.7. Exercice

Déterminer manuellement les machines $\mu_0, \mu_1, \mu_2, \mu_3$ ainsi que les valeurs $F(0)$, $F(1)$, $F(2)$, $F(3)$ de la fonction F définie par 6.2.6(1).

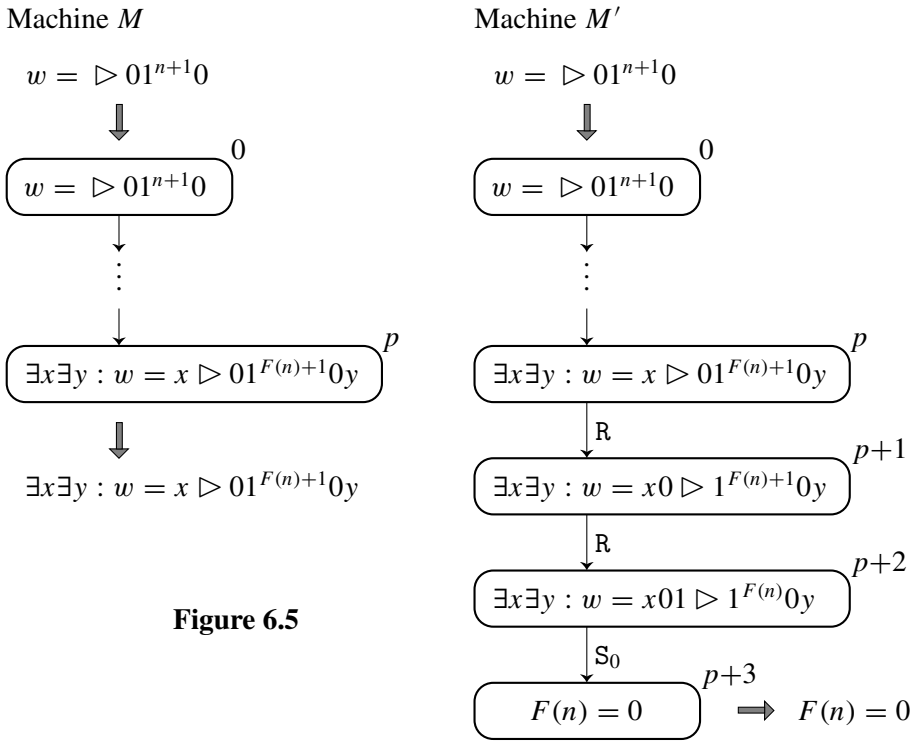


Figure 6.5

6.2.8. Exercice

Écrire un programme qui, étant donné un entier $n \in \mathbb{N}$ quelconque, fournit l'ensemble des transitions de la machine μ_n. Le choix du langage de programmation est libre.