Informatique théorique III
(Automates, langages & calculabilité)

Formulaire du Cours 2004-2005
EPFL – I&C

Uwe Nestmann
Sébastien Briais
Daniel C. Bünzli

14 février 2005

Bibliographie

[HMU03] John Hopcroft, Rajeev Motwani, and Jeffrey Ullman. *Introduction to Automata Theory, Languages and Computation*. Pearson Education International, 2003. ISBN 0321210298.

[Koz97] Dexter Kozen. *Automata and Computability*. Springer Verlag New York Inc., 1997.

[Sch95] Uwe Schöning. *Theoretische Informatik — kurzgefaßt*. Spektrum Lehrbuch. Spektrum Akademischer Verlag, 1995. 2. Auflage.

[Sch01] Carol Schumacher. *Chapter Zero — Fundamental Notions of Abstract Mathematics*. Addison Wesley, 2001.

[Sip97] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.

[Wol01] Pierre Wolper. *Introduction à la calculabilité — Cours et exercices corrigés*. Dunod, Paris, 2001. 2^e édition.

Table des matières

Plan du cours

$Id: notes-prelim.tex,v 1.40 2004/11/22 15:15:40 uwe Exp $

La plupart du matériel est tiré des livres mentionnés dans la bibliographie. Cependant, le livre principal est [HMU03] qui est recommandé à tous les participants du cours.

La répartition planifiée du matériel sur les semaines se trouve sur la page web du cours qui se trouve à

`http://lamp.epfl.ch/teaching/it3/2004/html/`

mais les semaines sont aussi indiqués sur les marges de ce document-ci.

Chapitre 0

Préliminaires

0.1 Notations

Pour clarifier nos propositions mathématiques nous utilisons une notation logique informelle. Pour des propositions quelconques A et B, nous écrivons,

- $A \wedge B$ pour (A et B), la conjonction de A et de B.
- $A \vee B$ pour (A ou B), la disjonction de A et de B.
- $\neg A$ pour (non A), la négation de A.
- $A \Rightarrow B$ pour (A implique B), qui signifie (si A alors B)
- $A \Leftrightarrow B$ pour (A ssi B) qui abrège (A si et seulement si B) et qui exprime l'équivalence logique de A et de B.

Pour nier des relations entre des objets mathématiques, nous tracons le symbole relationnel. Par exemple pour la relation d'égalité nous écrivons $A \neq B$ au lieu $\neg(A = B)$.

Une proposition $P(x, y)$ avec variables x et y est appellée un *prédicat*. Elle devient vraie ou fausse lorsque x et y sont remplacés par des objets particuliers. Pour quantifier les variables nous utilisons,

- $\exists x.\, P(x)$ pour (il existe x tel que $P(x)$), la quantification existentielle d'une variable.
- $\forall x.\, P(x)$ pour (pour tout x tel que $P(x)$), la quantification universelle d'une variable.

Nous utilisons fréquement les abréviations suivantes,

- $\exists x, y, \ldots, z.\, P(x)$ abrège $\exists x \exists y \ldots \exists z.\, P(x)$
- $\forall x, y, \ldots, z.\, P(x)$ abrège $\forall x \forall y \ldots \forall z.\, P(x)$.

Parfois il est pratique de pouvoir spécifier l'ensemble X auquel appartient la variable quantifiée. Nous écrivons donc,

- $\exists x \in X.\, P(x)$ pour $\exists x.\, x \in X \Rightarrow P(x)$
- $\forall x \in X.\, P(x)$ pour $\forall x.\, x \in X \Rightarrow P(x)$.

Finalement, nous écrivons $\exists! x.\, P(x)$, le quantificateur d'unicité qui affirme l'existence *unique* d'un objet x satisfaisant le prédicat $P(x)$. Ce dernier quantificateur n'est que l'abréviation de

$$\big(\exists x.\, P(x)\big) \wedge \big(\forall y, z.\, P(y) \wedge P(z) \Rightarrow y = z\big).$$

Lorsque nous souhaitons nommer par un identificateur A un objet mathématique X nous écrivons,

$$A \triangleq X$$

Cette notation est à distinguer de $A = X$ qui exprime une égalité entre deux objets mathématiques A et B.

0.2 Ensembles

Un ensemble est une collection non ordonnée d'objets dont on dit qu'ils sont les *éléments* ou *membres* de l'ensemble. Par convention les noms d'ensembles sont notés en majuscule. Nous écrivons $a \in A$ l'appartenance de l'objet a à l'ensemble A et $\{a, b, c, \ldots\}$ l'ensemble dont les éléments sont a, b, c, ...

Les ensembles suivants, dont on suppose l'existence, sont particulièrement importants.

- $\emptyset$, l'ensemble vide qui ne contient aucun élément. Il est caractérisé par la proposition $\forall x. x \notin \emptyset$.
- $\mathbb{N} \triangleq \{ 0, 1, 2, 3, \ldots \}$, l'ensemble des nombres naturels.
- $\mathbb{N}^* \triangleq \mathbb{N} \setminus \{ 0 \}$, l'ensemble des nombre naturels positifs.
- $[n, m] \triangleq \{n \in \mathbb{N} \mid n \leq i \wedge i \leq m\}$, pour $n, m \in \mathbb{N}$.
- $\mathbb{Z} \triangleq \{ \ldots, -2, -1, 0, 1, 2, \ldots \}$, l'ensemble des nombres entiers (relatifs).

0.2.1 Définition (Sous-ensemble et égalité) A est un *sous-ensemble* de B, noté $A \subseteq B$ ssi tout élément de A est un élément de B,

$$A \subseteq B \Leftrightarrow \forall x \in A. x \in B$$

Deux ensemble A et B sont *égaux*, noté $A = B$, ssi A est inclus dans B et vice-versa,

$$A = B \Leftrightarrow A \subseteq B \wedge B \subseteq A$$

Un ensemble A est un *sous-ensemble strict* de B, noté $A \subset B$ ssi il est sous-ensemble de B mais qu'il n'est pas égal à B.

$$A \subset B \Leftrightarrow A \subseteq B \wedge B \nsubseteq A$$

□

Pour construire des ensembles à partir d'autres ensembles nous utilisons les opérations suivantes.

0.2.2 Définition (Opérations sur les ensembles)

Compréhension. Si X est un ensemble et $P(x)$ un prédicat alors nous pouvons former l'ensemble

$$\{x \in X \mid P(x)\}$$

que l'on écrit aussi parfois $\{x \mid x \in X \wedge P(x)\}$. Il caractérise le plus grand sous-ensemble de X dont les éléments satisfont $P(x)$.

Indexation. Si I est un ensemble et que pour tout $i \in I$ il y a un objet x_i alors on peut former l'ensemble

$$\{x_i \mid i \in I\}$$

dont les éléments x_i sont dits *indexés* par les éléments de I.

Union. L'union $A \cup B$ de deux ensembles A et B contient les éléments de chacun de ceux-ci.

$$A \cup B \triangleq \{x \mid x \in A \vee x \in B\}$$

Intersection. L'intersection $A \cap B$ de deux ensembles A et B contient les éléments présent dans chacun de ceux-ci.

$$A \cap B \triangleq \{x \mid x \in A \wedge x \in B\}$$

Produit. Le produit $A \times B$ de deux ensembles est l'ensemble des paires (x, y) ordonnées dont la première composante est dans A et la seconde dans B.

$$A \times B \triangleq \{(x, y) \mid x \in A \wedge y \in B\}$$

Complément. Le complément $A \backslash B$ d'un ensemble B relativement à un ensemble A, est l'ensemble A dont les éléments de B sont soustraits.

$$A \backslash B \triangleq \{x \mid x \in A \wedge x \notin B\}$$

Lorsque l'ensemble A est implicite ou évident dans le contexte, on écrit simplement $\overline{B}$ pour $A \backslash B$.

Ensemble des parties. L'ensemble des parties $\mathcal{P}(A)$ d'un ensemble A est l'ensemble des sous-ensembles de A.

$$\mathcal{P}(A) \triangleq \{B \mid B \subseteq A\}$$

0.3 Relations

Une *relation* n-aire sur une collection d'ensembles $A_1, \ldots, A_n$ est un ensemble $R \subseteq A_1 \times \ldots \times A_n$. Les éléments $x_1 \in A_1, \ldots, x_n \in A_n$ sont dit *reliés* par R si $(x_1, \ldots, x_n) \in R$. Une relation est un ensemble et l'ensemble des relations sur la collection $A_1, \ldots, A_n$ est l'ensemble d'ensembles $\mathcal{P}(A_1 \times \ldots A_n)$.

Une relation unaire P sur un ensemble A est appelée un *prédicat*. On dit que P est vraie pour un élément $x \in A$ ssi $x \in P$. Au lieu de noter $x \in P$ l'on note parfois $P(x)$, en interprétant P comme une fonction qui associe les éléments de A aux valeurs de vérité.

Pour une relation binaire $R \subseteq A \times B$ sur deux ensembles A et B nous écrivons parfois aRb au lieu de $(a, b) \in R$.

0.3.1 Définition (Identité) L'identité id_A sur un ensemble A est la relation binaire sur $A \times A$ définie par $\mathrm{id}_A \triangleq \{(x, x) \mid x \in A\}$. □

0.3.2 Définition (Propriétés des relations binaires) Une relation binaire $R \subseteq A \times A$ sur un ensemble A est,

réflexive si $\forall a \in A.\, aRa$,

irréflexive si $\forall a \in A.\, \neg aRa$,

symétrique si $\forall a, b \in A.\, aRb \Rightarrow bRa$,

antisymétrique si $\forall a, b \in A.\, (aRb \wedge bRa) \Rightarrow a = b$,

transitive si $\forall a, b, c \in A.\, (aRb \wedge bRc) \Rightarrow aRc$,

totale si $\forall a, b \in A.\, aRb \vee bRa$,

une trichotomie si $\forall a, b \in A.\, aRb \vee a = b \vee bRa$ □

Pour construire des relations à partir d'autres relations nous utilisons notamment les opérations suivantes.

0.3.3 Définition (Opérations sur les relations)

Raffinement. Soit R une relation, R' est un raffinement de R si $R' \subseteq R$.

Inverse. Soit $R \subseteq A \times B$ une relation, l'inverse (ou l'opposé) $R^{-1} \subseteq B \times A$ d'une relation R est défini par

$$R^{-1} \triangleq \{(b, a) \in B \times A \mid (a, b) \in R\}$$

Composition. Soit $R \subseteq A \times B$ et $R' \subseteq B \times C$ la composition $RR' \subseteq A \times C$ de R et de R' est définie par,

$$RR' \triangleq \{(a, c) \in A \times C \mid \exists b \in B.(a, b) \in R \wedge (b, c) \in R'\}$$

La composition RR' est aussi notée $R' \circ R$ (notamment pour les fonctions).

Fermeture réflexive. Soit R une relation, la fermeture réflexive est la plus petite relation réflexive R' qui contient R.

Fermeture transitive. Soit R une relation, la fermeture transitive R^+ de R est la plus petite relation transitive qui contient R. La fermeture transitive et réflexive d'une relation R est notée R^*. □

0.3.4 Lemme

1. Si R est symétrique alors $R = R^{-1}$.
2. Soit $R \subset A \times A$, la relation $R' \triangleq R \cup \{(a, a) \mid A \in A\}$ est la fermeture réflexive de R.
3. Soit $R \subset A \times A$, et la suite de relation R^i définie par induction de la manière suivante,

$$R^1 \triangleq R \qquad R^{i+1} \triangleq R\, R^i$$

alors l'ensemble $R^+ \triangleq \bigcup_{i \in \mathbb{N}} R_i$ est la fermeture transitive de R. □

0.3.5 Définition (Ordres) Soit A un ensemble et $R \subseteq A \times A$ une relation.

Pré-ordre. Le couple (A, R) est un pré-ordre si R est réflexive et transitive.

Ordre (partiel). Le pré-ordre (A, R) est un ordre (partiel) si R est antisymétrique.

Ordre total. L'ordre (A, R) est un ordre total si de plus R est totale.

Ordre strict. Le couple (A, R) est un ordre strict si R est irréflexive et transitive.

Ordre total strict. Le couple (A, R) est un ordre total strict si (A, R) est un ordre strict et R est une trichotomie.

En général, on utilise les symboles $\leq$, $\preceq$ et $\sqsubseteq$ pour les ordres, $<$, $\prec$ et $\sqsubset$ pour les ordres stricts. □

0.3.6 Définition (Relation d'équivalence) Une *relation d'équivalence* sur A est une relation $R \subseteq A \times A$ réflexive, symétrique et transitive.

La *classe d'équivalence* de $a \in A$ dans R est l'ensemble défini par

$$[a]_R \triangleq \{\, b \in A \mid aRb \,\}$$

Lorsqu'il n'y a pas d'ambiguité sur la relation R dont on parle, on écrit $[a]$ pour $[a]_R$.

Le *quotient* $A/_R$ est défini par :

$$A/_R \triangleq \{\, [a]_R \mid a \in A \,\}$$

c'est l'ensemble des classes d'équivalences de A.
L'*index* de R est le nombre de classes d'équivalence de R. Il est égal à $\#(A/_R)$ (cf. 0.5.1). □

0.3.7 Lemme Soit R une relation d'équivalence sur un ensemble A et $p, q \in A$, alors :

1. pRq ssi $[p] = [q]$.
2. Si $[p] \neq [q]$, alors $p \notin [q]$ et $q \notin [p]$.
3. $[p] = [q] \vee [p] \cap [q] = \emptyset$
4. $A = \bigcup_{r \in A} [r]$. □

0.3.8 Lemme Soit R, S deux équivalences sur A avec index fini et $R \subseteq S$.

1. $\forall p \in A \,.\, [p]_R \subseteq [p]_S$.
2. $\#(A/_R) \geq \#(A/_S)$. □

0.4 Fonctions

La notion de fonction est un cas particulier de la notion de relation binaire.

0.4.1 Définition (Fonction) Une *fonction partielle* de A dans B est un triplet $f = (A, B, R)$ où $R \subseteq A \times B$ est une relation, appelé le *graphe* de f, telle que

$$\forall a, b, b'. \big((a, b) \in R \wedge (a, b') \in R\big) \Rightarrow b = b'$$

On définit les ensembles suivants,

- $\mathrm{ED}(f) \triangleq A$, l'ensemble de départ de f,
- $\mathrm{EA}(f) \triangleq B$, l'ensemble d'arrivée de f,
- $\mathrm{dom}(f) \triangleq \{a \in A \mid \exists b \in B. (a, b) \in R\}$, le domaine (de définition) de f.
- $\mathrm{img}(f) \triangleq \{b \in B \mid \exists a \in A. (a, b) \in R\}$, l'image de f.

La fonction f est définie (resp. indéfinie) en $a \in A$ si $a \in \mathrm{dom}(f)$ (resp. $a \notin \mathrm{dom}(f)$).

La fonction f est totale si $\mathrm{dom}(f) = \mathrm{ED}(f)$. On dit aussi de f qu'elle est une application de A dans B.

L'ensemble des fonctions partielles de A dans B est noté $A \rightharpoonup B$. On note $A \to B$ l'ensemble des applications de A dans B. Remarquons que

$$(A \to B) \subseteq (A \rightharpoonup B) \subseteq \mathcal{P}(A \times B).$$

On écrit $f(a) = b$ ou $a \mapsto b$ si $(a, b) \in R$. □

Par abus de langage, on confond le graphe R d'une fonction $f = (A, B, R)$ avec f et on dit que f est une fonction, souvent sans préciser A ou B.

Souvent au lieu de noter $f \in A \to B$ nous écrivons $f : A \to B$. Pour définir les fonctions nous utilisons les notations suivantes toutes équivalentes.

$$f : x \mapsto x^2 \qquad f(x) \triangleq x^2 \qquad f \triangleq \lambda x.x^2$$

Pour spécifier l'ensemble de départ et d'arrivée de la fonction définie, nous écrivons,

$$\begin{array}{rcccc} f & : & A & \to & B \\ & & x & \mapsto & x^2 \end{array}$$

Une application $f : \mathbb{N} \to A$ est communément appelé *suite* sur A ou suite de A.

0.4.2 Définition (Égalité) Deux fonctions $f : A \rightharpoonup B$ et $g : A' \rightharpoonup B'$ sont égales (noté $f = g$) si :

1. $A = A'$ et $B = B'$,
2. $\mathrm{dom}(f) = \mathrm{dom}(g)$, et
3. $\forall a \in \mathrm{dom}(f)\,.\, f(a) = g(a)$.

0.4.3 Définition (Image d'un ensemble) Soit $f \in A \rightharpoonup B$.

Soit $X \subseteq A$, *l'image directe* de X par f est

$$f(X) \triangleq \{f(a) \mid a \in (X \cap \mathrm{dom}(f))\}$$

Soit $Y \subseteq B$, *l'image réciproque* de Y par f est

$$f^R(Y) \triangleq \{a \in \mathrm{dom}(f) \mid f(a) \in Y\}$$

□

0.4.4 Définition (Fonction injective, surjective, bijective) Une fonction f de A dans B est,

injective ssi $\forall a, a'.\, f(a) = f(a') \Rightarrow a = a'$

surjective ssi $\forall b \in B.\, \exists a \in A.\, f(a) = b$

bijective ssi f est injective et surjective. □

0.4.5 Lemme

Soit A et B deux ensembles non vides.

1. Si $f : A \to B$ est une application injective,
   alors il existe une application $g : B \to A$ qui est surjective.
2. Si $g : B \to A$ est une application surjective,
   alors il existe une application $f : A \to B$ qui est injective.
3. Soit $f : A \to B$ une application, f^{-1} est une application
   ssi f est injective. □

0.4.6 Définition (Application involutive, idempotente) Soit $f : A \to A$ une application.

- f est *involutive* ssi $f \circ f = \mathrm{id}_A$.
- f est *idempotente* ssi $f \circ f = f$. □

0.4.7 Définition (Fermeture) Soit $(X, \sqsubseteq)$ un ordre. Une application $f : X \to X$ est une *fermeture* (sur X) si pour tout $x, y \in X$:

1. $x \sqsubseteq f(x)$,
2. si $x \sqsubseteq y$ alors $f(x) \sqsubseteq f(y)$,
3. $f(f(x)) = f(x)$

Un élément x est un *point fixe* de f si $x = f(x)$. □

0.4.8 Définition (Stabilité) Soit $f : X^n \to X$ une application. Un sous-ensemble $Y \subseteq X$ est *stable par* f si :

$$\forall y_1 \ldots, y_n \in Y\,.\, f(y_1 \ldots, y_n) \in Y$$

0.5 Cardinalité

0.5.1 Définition (Taille d'un ensemble) Soit A un ensemble. On dit que A est *fini* s'il existe $n \in \mathbb{N}$ tel qu'il existe une application bijective de A dans $[1, n]$. Si un tel n existe, il est unique et est appelé la taille de l'ensemble A, que l'on notera $\#(A)$. Si A n'est pas fini, on dit que A est *infini* et on pose $\#(A) = \infty$. □

Si, pour les ensembles finis, il est facile de comparer leur taille, pour les ensembles infinis, on introduit la notion de *cardinal* pour les comparer.

0.5.2 Définition (Cardinalité) Soit A et B deux ensembles.

1. A et B ont la même cardinalité, $\mathrm{card}(A) = \mathrm{card}(B)$, s'il existe une application bijective de A dans B. On dit aussi que A et B sont *équipotents*.
2. A a une cardinalité plus petite que celle de B, $\mathrm{card}(A) \leq \mathrm{card}(B)$, s'il existe une application injective de A dans B.
3. A a une cardinalité strictement plus petite que celle de B, $\mathrm{card}(A) < \mathrm{card}(B)$, si $\mathrm{card}(A) \leq \mathrm{card}(B)$ mais $\mathrm{card}(A) \neq \mathrm{card}(B)$.

Il est possible de reformuler les comparaisons de cardinalité en termes d'applications surjectives.

0.5.3 Définition (Dénombrable) Soit A un ensemble.

1. A est *dénombrable* si A est fini ou équipotent à $\mathbb{N}$.
2. A est *infiniment dénombrable* si A est infini et dénombrable.
3. A est *non-dénombrable* si $\mathrm{card}(\mathbb{N}) < \mathrm{card}(A)$.

0.5.4 Théorème (Cantor) Soit A un ensemble. Alors $\mathrm{card}(A) < \mathrm{card}(\mathcal{P}(A))$.

0.6 Structures algébriques

0.6.1 Définition (Monoïde) Soit M un ensemble et op une application de $M \times M$ dans M (aussi appelé loi de composition interne). Le couple (M, op) est un *monoïde* si :

- op est *associative* :

$$\forall x, y, z \in M : \mathrm{op}(x, \mathrm{op}(y, z)) = \mathrm{op}(\mathrm{op}(x, y), z)$$

- op admet un *élément neutre* :

$$\exists e \in M : \forall x \in M : \mathrm{op}(e, x) = \mathrm{op}(x, e) = x$$

0.6.2 Lemme (Unicité de l'élément neutre) Soit (M, op) un monoïde et soit $e, e' \in M$ deux éléments neutres pour op. Alors $e = e'$.

On a donc que si (M, op) est un monoïde, il existe un unique élément neutre pour op appelé l'élément neutre de (M, op). Si e est cet élément neutre, on pourra aussi dire que (M, op) est un monoïde d'élément neutre e.

Si (M, op) est un monoïde, il est d'usage de noter de manière infixe la loi de composition interne : $\mathrm{op}(x, y)$ est plutôt noté $x \,\mathrm{op}\, y$. Ainsi, l'associativité de op s'écrit alors $\forall x, y, z \in M : x \,\mathrm{op}(y \,\mathrm{op}\, z) = (x \,\mathrm{op}\, y) \,\mathrm{op}\, z$.

0.6.3 Définition (Puissance d'un élément) Soit (M, op) un monoïde d'élément neutre e. On définit pour $x \in M$, la puissance n^{e} d'un élément, pour $n \in \mathbb{N}$, comme suit :

$$x^0 \triangleq e$$
$$x^{n+1} \triangleq x \,\mathrm{op}\, x^n$$

On a donc $x^n = \underbrace{x \,op \cdots \mathrm{op}\, x}_{n \text{ fois}}$.

0.6.4 Définition (Monoïde commutatif) Soit (M, op) un monoïde. Le couple (M, op) est dit *commutatif* si

– op est *commutative* :

$$\forall x, y \in M : x \,\mathrm{op}\, y = y \,\mathrm{op}\, x$$

La loi de composition interne op d'un monoïde commutatif est souvent noté $+$ et l'élément neutre $\mathbf{0}$. Si la loi est noté $+$, on note généralement nx la puissance n^{e} de x.

Un exemple de monoïde commutatif est $(\mathbb{N}, +)$ où $+$ est l'addition classique sur les entiers naturels. L'élément neutre de $(\mathbb{N}, +)$ est bien entendu 0.

0.6.5 Définition (Groupe) Soit G un ensemble muni d'une loi de composition interne $\mathrm{op} : G \times G \to G$. Le couple (G, op) est un groupe si :

– (G, op) est un monoïde d'élément neutre e,

– Chaque élément de G admet un *symétrique* à gauche et à droite :

$$\forall x \in G : \exists y \in G : x \,\mathrm{op}\, y = y \,\mathrm{op}\, x = e$$

De plus, le groupe (G, op) est dit *abélien* (ou commutatif) si op est commutative.

0.6.6 Lemme (Unicité du symétrique) Soit (G, op) un groupe d'élément neutre e et $x, y, y' \in G$ tels que $x \,\mathrm{op}\, y = y \,\mathrm{op}\, x = e$, $x \,\mathrm{op}\, y' = y' \,\mathrm{op}\, x = e$. Alors $y = y'$.

Le lemme précédent dit que chaque élément x d'un groupe admet un unique élément symétrique, que l'on notera généralement x^{-1} (ou $-x$ si on utilise $+$ pour dénoter la loi de composition interne du groupe abélien).

Notons aussi qu'il est possible de prendre une définition de groupe ne requérant seulement l'existence d'un symétrique à gauche pour chaque

élément. C'est un bon exercice de montrer que cette définition en apparence moins forte coïncide avec celle donnée ci-dessus.

Le couple $(\mathbb{Z}, +)$ est un exemple de groupe commutatif (où $+$ est l'addition classique sur les entiers relatifs) d'élément neutre 0. En revanche $(\mathbb{N}, +)$ n'est pas un groupe car tous les éléments distincts de 0 n'ont pas de symétrique.

Chapitre 1

Introduction

$Id: notes-1.tex,v 1.16 2005/01/27 10:18:00 jobo Exp $

semaine 1
lundi &
mercredi

1.1 Alphabets et mots

1.1.1 Définition (Alphabet) On appelle *alphabet* tout ensemble fini Σ. Les éléments d'un alphabet sont traditionnellement appelés *symboles*, *lettres* ou *caractères*. □

Pour un alphabet $\Sigma \triangleq \{ s_1, s_2, \ldots, s_k \}$, on considère souvent l'ordre total $\preceq$ défini par $s_i \preceq s_j$ si et seulement si $i \leq j$.

1.1.2 Définition (Mots) Un *mot* w sur un alphabet Σ est une séquence finie de lettres de Σ.

Autrement dit, un mot w est un n-uplet $(a_1, a_2, \ldots, a_n)$ de lettres de Σ où $n \in \mathbb{N}$ est appelé la *longueur* de w, notée $|w|$. Si $n = 0$, w est l'unique mot de longueur nulle appelé *mot vide* que l'on note ϵ . La i^e *projection* $(w)_i$ de w est la lettre a_i, pour $1 \leq i \leq n$.

L'ensemble des mots sur un alphabet Σ est noté Σ^*. L'ensemble des mots non vides, c'est à dire $\Sigma^* \setminus \{\epsilon\}$ est noté Σ^+. □

Par convention, nous utilisons les (meta-) variables $a, b, c, \ldots$ pour les lettres et $\ldots, u, v, w, x, y, z$ les mots.

1.1.3 Définition (Concaténation de mots) Soit $w \triangleq (a_1, \ldots, a_n) \in \Sigma^*$ et $w' \triangleq (b_1, \ldots, b_m) \in \Sigma^*$, la *concaténation* $w \cdot w'$ de w avec w' est définie par,

$$w \cdot w' \triangleq (a_1, \ldots, a_n, b_1, \ldots, b_m)$$

□

1.1.4 Théorème $(\Sigma^*, \cdot)$ est un monoïde d'élément neutre ϵ.

1.1.5 Notation Pour un alphabet $\Sigma \triangleq \{s_1, \ldots, s_n\}$ on identifie la lettre $s_i \in \Sigma$ avec le mot $(s_i) \in \Sigma^*$ de longueur 1. Ceci nous permet d'écrire pour $w \triangleq (a_1, a_2, \ldots, a_n) \in \Sigma^*$, $w = a_1 \cdot a_2 \cdot \ldots \cdot a_n = a_1 a_2 \ldots a_n$ □

1.1.6 Définition (Préfixe, suffixe, facteur) Soit $w \in \Sigma^*$ un mot sur un alphabet Σ.

- $u \in \Sigma^*$ est un *préfixe* de w s'il existe $v \in \Sigma^*$ tel que $w = u \cdot v$.
- $v \in \Sigma^*$ est un *suffixe* de w s'il existe $u \in \Sigma^*$ tel que $w = u \cdot v$.
- $w' \in \Sigma^*$ est un *facteur* de w s'il existe $u, v \in \Sigma^*$ tel que $w = u \cdot w' \cdot v$.

1.1.7 Lemme (Décomposition d'un mot) Soit w un mot sur un alphabet Σ , nous avons,

Soit $w = \epsilon$

ou alors $\exists a \in \Sigma, w' \in \Sigma^*. w = a \cdot w'$ avec $|w'| = |w| - 1$ □

1.1.8 Lemme (Principe d'induction sur les mots) Soit $P(w)$ un prédicat sur les mots d'un alphabet Σ.

Si $P(\epsilon)$

et $\forall w \in \Sigma^*. P(w) \Rightarrow \forall a \in \Sigma. P(a \cdot w)$

alors $\forall w \in \Sigma^*. P(w)$. □

Remarquons que les deux derniers lemmes peuvent se reformuler en décomposant les mots par la droite.
Si le dernier lemme nous permet de montrer qu'un prédicat est vrai sur l'ensemble des mots, il nous permet aussi de justifier les définitions par induction structurelle sur les mots en définissant

- le cas de base : pour le mot vide ϵ
- le cas inductif : supposant avoir défini pour le mot $w \in \Sigma^*$, on définit pour les mots $a \cdot w$ où $a \in \Sigma$.

Par exemple, on peut définir la longueur $\mathsf{len}(w)$ d'un mot w par induction structurelle en définissant :

- le cas de base. On définit $\mathsf{len}(\epsilon) \triangleq 0$
- le cas inductif. Soit $w \in \Sigma^*$ et supposons que $\mathsf{len}(w)$ soit défini et égal à n. On définit alors pour tout $a \in \Sigma$, $\mathsf{len}(a \cdot w) \triangleq n + 1 = \mathsf{len}(w) + 1$.

On peut alors montrer que $\mathsf{len}(w)$ et $|w|$ sont égaux pour tous mots $w \in \Sigma^*$ (par induction sur w !).

Ce genre de définition par induction sur les mots étant fréquent, on utilisera la présentation sous forme de *règles d'inférence*.

En continuant sur le même exemple, cela donnerait :

$$\frac{}{\mathsf{len}(\epsilon) \triangleq 0} \qquad \frac{\mathsf{len}(w) = n}{\mathsf{len}(a \cdot w) \triangleq n + 1}$$

Ces règles d'inférence se lisent : en supposant les prémisses (ce qui se trouve au-dessus de la barre), on définit la conclusion (ce qui se trouve en-dessous de la barre). S'il n'y a pas de prémisses, on parle de règle axiome (ou plus simplement axiome).

L'ordre total d'un alphabet Σ induit des ordres sur les mots de Σ^*.

1.1.9 Définition (Ordre sur les mots) Soit un alphabet Σ et un ordre total $(\Sigma, \preceq)$ (et l'ordre strict induit $\prec$).

Ordre alphabétique. L'ordre alphabétique $(\Sigma^*, \ll_{\mathrm{d}})$ est la plus petite relation sur Σ^* qui satisfait les règles suivantes.

$$\mathrm{D}_1\ \frac{}{\epsilon \ll_{\mathrm{d}} w} \qquad \mathrm{D}_2\ \frac{}{a \cdot w \ll_{\mathrm{d}} a' \cdot w'}\ a \prec a'$$

$$\mathrm{D}_3\ \frac{w \ll_{\mathrm{d}} w'}{a \cdot w \ll_{\mathrm{d}} a \cdot w'}$$

Ordre lexicographique. L'ordre lexicographique $(\Sigma^*, \ll_{\mathrm{l}})$ est la plus petite relation sur Σ^* qui satisfait les règles suivantes.

$$\mathrm{L}_1\ \frac{}{\epsilon \ll_{\mathrm{l}} w} \qquad \mathrm{L}_2\ \frac{}{w \cdot a \ll_{\mathrm{l}} w \cdot a'}\ a \preceq a'$$

$$\mathrm{L}_3\ \frac{w \ll_{\mathrm{l}} w'}{w \cdot a \ll_{\mathrm{l}} w' \cdot a'}\ w \neq w'$$

L'ordre lexicographique ordonne les mot d'abord par leur taille puis, si celle-ci est égale, alphabétiquement. □

1.1.10 Théorème

1. L'ordre alphabétique est total.
2. L'ordre lexicographique est total. □

1.2 Langages

1.2.1 Définition (Langage) On appelle *langage* sur un alphabet Σ tout sous-ensemble de Σ^*. Un langage n'est donc rien d'autre qu'un ensemble de mots. □

Nous utilisons les meta-variables $A, B, C \ldots, L, \ldots$ pour les langages. Les deux langages $\emptyset$ et $\{\epsilon\}$ sont des langages sur n'importe quel alphabet.

Pour construire des langages à partir d'autres langages, nous utilisons notamment les opérations suivantes.

1.2.2 Définition (Opération sur les langages) Soit Σ un alphabet fini, L et L' deux langages sur Σ^*.

Union. L'union $L \cup L'$ de L et L' correspond à leur union ensembliste.

Intersection. L'intersection $L \cap L'$ de L et L' correspond à leur intersection ensembliste.

Complément. Le complément $\overline{L}$ de L est le complément absolu de L par rapport à Σ^* (i.e. $\overline{L} = \Sigma^* \backslash L$) .

Concaténation. La concaténation LL' de L et L' est définie par $LL' \triangleq \{ww' \mid w \in L \wedge w' \in L'\}$.

Itération. La n^e itération L^n d'un language n est définie de manière inductive par,

$$L^0 \triangleq \{\epsilon\} \qquad L^{n+1} \triangleq LL^n$$

Fermeture itérative. La fermeture itérative (ou de Kleene) L^* de L est définie par

$$L^* \triangleq \bigcup_{n\in\mathbb{N}} L^n$$

On utilise aussi la notation suivante, $L^+ \triangleq LL^*$.

1.2.3 Lemme

1. Propriétés de la fermeture de Kleene.

$$L^*L^* = L^* \quad L^{**} = L^* \quad \emptyset^* = \{\,\epsilon\,\}$$

□

1.3 Fonctions versus programmes

Étant donné un langage de programmation quelconque **X** (exécuté de façon déterministe), nous démontrons qu'il existe plus de fonctions sur $\mathbb{N}$ que de programmes de **X** pouvant calculer des fonctions sur $\mathbb{N}$.

1.3.1 Définition L'ensemble $\mathbf{F}_\mathbb{N}$ de toutes les fonctions totales sur $\mathbb{N}$ est défini par :

$$\mathbf{F}_\mathbb{N} \triangleq \{\, f \mid f : \mathbb{N} \to \mathbb{N} \,\}$$

1.3.2 Définition Soit Σ l'alphabet d'un langage de programmation **X**. L'ensemble $\mathbf{P}_\mathbf{X}$ des mots/programmes sur Σ (voir §1.1 et §1.2) est défini par :

$$\mathbf{P}_\mathbf{X} \triangleq \Sigma^*$$

1.3.3 Définition Soit $\mathrm{exec}_\mathbf{X} : \mathbf{P}_\mathbf{X} \rightharpoonup (\mathbb{N} \to \mathbb{N})$ un interpréteur de **X** qui associe à un programme $\mathtt{p}$ une fonction $\mathrm{exec}_\mathbf{X}(\mathtt{p}) : \mathbb{N} \to \mathbb{N}$. Notons que $\mathrm{exec}_\mathbf{X}$ est partielle, tous les programmes ne sont pas associés à une fonction. Une fonction $f : \mathbb{N} \to \mathbb{N}$ est dite *calculable* dans le langage **X** si $f \in \mathrm{img}(\mathrm{exec}_\mathbf{X})$, c'est à dire si

$$\exists \mathtt{p} \in \mathbf{P}_\mathbf{X} \,.\, \mathrm{exec}_\mathbf{X}(\mathtt{p}) = f$$

Nous écrivons

$$\mathbf{F}_\mathbf{X} \triangleq \{\, f : \mathbb{N} \to \mathbb{N} \mid f \text{ calculable par } \mathbf{X} \,\} \qquad (\subseteq \mathbf{F}_\mathbb{N})$$

l'ensemble de fonctions calculables par (les programmes) X.

1.3.4 Théorème Pour tout **X** : $\mathrm{card}(\mathbf{F}_\mathbf{X}) < \mathrm{card}(\mathbf{F}_\mathbb{N})$.

Nous décomposons la preuve comme suit :

$$\mathrm{card}(\mathbf{F}_\mathbf{X}) \overset{(0)}{\leq} \mathrm{card}(\mathbf{P}_\mathbf{X}) \overset{(1)}{\leq} \mathrm{card}(\mathbb{N}) \overset{(2)}{<} \mathrm{card}(\mathcal{P}(\mathbb{N})) \overset{(3)}{\leq} \mathrm{card}(\mathbf{F}_\mathbb{N})$$

1.3.5 Lemme (Partie (0) de la preuve de 1.3.4) $\mathrm{card}(\mathbf{F_X}) \leq \mathrm{card}(\mathbf{P_X})$

1.3.6 Lemme (Partie (1) de la preuve de 1.3.4) $\mathrm{card}(\mathbf{P_X}) \leq \mathrm{card}(\mathbb{N})$

1.3.7 Lemme (Partie (2) de la preuve de 1.3.4) $\mathrm{card}(\mathbb{N}) < \mathrm{card}(\mathcal{P}(\mathbb{N}))$

1.3.8 Lemme (Partie (3) de la preuve de 1.3.4) $\mathrm{card}(\mathcal{P}(\mathbb{N})) \leq \mathrm{card}(\mathbf{F}_{\mathbb{N}})$

1.3.9 Théorème Supposons qu'il existe une énumeration $f_0, f_1, f_2, \ldots$ de toutes les fonctions sur les nombres naturels. Définissons :

$$\mathrm{diag}(i) \triangleq f_i(i)$$

Alors, la fonction définie par

$$g(i) \triangleq \mathrm{diag}(i) + 1$$

n'apparaît pas dans cette énumeration.

$Id: notes-2.tex,v 1.26 2005/01/28 10:25:22 uwe Exp $

semaine 2
lundi

1.4 La hiérarchie de Chomsky

1.4.1 Définition (Grammaire générative)
Une grammaire est un quadruplet $G \triangleq (V, \Sigma, P, S)$.

- V est un alphabet non vide de symboles *non-terminaux*.
- Σ est un alphabet non vide de symboles *terminaux* disjoint de V ($V \cap \Sigma = \emptyset$).
- $P \subseteq (\Gamma^+ \times \Gamma^*)$ (avec $\Gamma \triangleq V \cup \Sigma$) est un ensemble fini de *productions*.
- $S \in V$ est le *symbole de départ*. □

Nous utilisons des lettres grecques minuscules $\alpha, \beta, \ldots$ pour les éléments de $(V \cup \Sigma)^*$. Pour spécifier une production $(\alpha, \beta) \in P$, on note $\alpha \to \beta$ ou encore $\alpha \to_G \beta$.

1.4.2 Définition (Dérivation directe) Soit $G \triangleq (V, \Sigma, P, S)$ une grammaire. La relation de *dérivation directe* dans G entre deux mots $\alpha, \beta \in (V \cup \Sigma)^*$ est définie par,

$$(\alpha \Rightarrow_G \beta) \quad \Leftrightarrow \quad \exists \alpha', \beta', \gamma, \gamma' \in (V \cup \Sigma)^* : \begin{cases} \alpha' \to_G \beta' \\ \alpha = \gamma \alpha' \gamma' \\ \beta = \gamma \beta' \gamma' \end{cases}$$

On dit alors que β est *directement dérivable* de α dans G. □

1.4.3 Définition (Dérivation) Soit $G \triangleq (V, \Sigma, P, S)$ une grammaire. On définit la relation de dérivation $\Rightarrow_G^*$ comme étant la fermeture réflexive et transitive de $\Rightarrow_G$.

Une *dérivation* de α_0 en α_n dans G est une séquence finie $(\alpha_i)_{i \in [0,n]}$ d'éléments de $(V \cup \Sigma)^*$ telle que $\forall i \in [\![0, n-1]\!] : \alpha_i \Rightarrow_G \alpha_{i+1}$. On note généralement cette séquence $\alpha_o \Rightarrow_G \alpha_1 \Rightarrow_G \ldots \Rightarrow_G \alpha_n$. □

1.4.4 Définition (Langage d'une grammaire) Soit $G = (V, \Sigma, P, S)$ une grammaire.

1. Un mot $v \in \Sigma^*$ est généré par G ssi $S \Rightarrow_G^* v$. Dans ce cas on dit que v est un mot *terminal* de la grammaire.
2. Le langage généré par G est défini par :

$$\mathrm{L}(G) \triangleq \{ v \in \Sigma^* \mid S \Rightarrow_G^* v \}$$

1.4.5 Définition (Type d'une grammaire) Soit $G = (V, \Sigma, P, S)$ une grammaire. Un grammaire est dite de

Type 0 dans tous les cas.

Type 1 si pour toute production $\alpha \to_G \beta \in P$ on a,

1. $|\alpha| \leq |\beta|$
2. ou $\alpha = S \wedge \beta = \epsilon$.

La grammaire G est dite *contextuelle*.

Type 2 si pour toute production $\alpha \rightarrow_G \beta \in P$ on a $\alpha \in V$. La grammaire est dite *libre de contexte* ou *non-contextuelle*.

Type 3 si toutes les production $\alpha \rightarrow_G \beta \in P$ sont de la forme,

1. $A \rightarrow_G wB$
2. $A \rightarrow_G w$

avec $A, B \in V$ et $w \in \Sigma^*$. La grammaire est dite *régulière*.

Avec cette définition, on parle de grammaires *linéaires à droite*. Si l'on remplace ci-dessus wB par Bw, on parle de grammaires *linéaires à gauche*. Ces deux définitions sont équivalentes, un langage générable par une grammaire linéaire à droite l'est aussi par une grammaire linéaire à gauche, et vice-versa.

1.4.6 Définition L'ensemble des langages générés par les grammaires d'un type i est donné par :

$$\mathcal{L}_i = \{\mathrm{L}(G) \mid G \text{ est de type } i\}\,.$$

Un type de grammaire i est *inclus* dans un type j si $\mathcal{L}_i \subseteq \mathcal{L}_j$. □

1.4.7 Définition (Type d'un langage) Un langage L est de type i si $L \in \mathcal{L}_i$ et L est *strictement* de type i si $L \in \mathcal{L}_i \setminus \mathcal{L}_{i+1})$, c'est à dire si L est généré par une grammaire de type i et par aucune grammaire de type $i+1$.

1.4.8 Théorème (Hiérarchie de Chomsky)
La relation entre les types de langages est :

$$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$$

Les inclusions sont strictes.

Étant donné une grammaire $G = (V, \Sigma, P, S)$ de type 1, il existe un algorithme qui permet de décider en temps fini si un mot $x \in \Sigma^*$ appartient au langage $\mathrm{L}(G)$ ou non.

Chapitre 2

Langages réguliers

2.1 Automates finis déterministes

semaine 2
mercredi

2.1.1 Définition Un *automate fini déterministe (AFD)* est un quintuplet

$$M = (Q, \Sigma, \delta, s, F)$$

où

- Q est un ensemble fini d'*états*,
- Σ est un ensemble fini de symboles, *l'alphabet (d'entrée)*,
- $\delta : Q \times \Sigma \to Q$ est la *fonction (totale) de transition*,
- $s \in Q$ est l'*état initial* (ou de *départ*),
- $F \subseteq Q$ est un sous-ensemble de Q, les *états accepteurs* (ou *finaux*).

2.1.2 Notation Un automate $(Q, \Sigma, \delta, s, F)$ peut aussi être défini en spécifiant son alphabet $\Sigma = \{ s_1, \ldots, s_n \}$, son ensemble d'état $Q = \{ q_1, \ldots, q_m \}$ *et* l'un des éléments suivants :

- un *graphe* :
  1. Pour tout état $q \in Q$ on dessine un noeud dans le graphe (q entouré d'un cercle).
  2. Pour toute transition $\delta(q, s) = q'$ on dessine une flèche étiquettée par s allant du noeud q à q'
  3. L'état initial s est distingué par une flèche dont la queue n'est attachée à aucun état et dont la tête pointe sur s.
  4. Pour tout état accepteur $q \in F$, on dessine un second cercle autour de son noeud correspondant.

- un *tableau* de la forme :

δ	s_1	$\cdots$	s_i	$\cdots$	s_n
q_1	$\delta(q_1, s_1)$	$\cdots$	$\delta(q_1, s_i)$	$\cdots$	$\delta(q_1, s_n)$
$\vdots$	$\vdots$		$\vdots$		$\vdots$
$F\, q_j$	$\delta(q_j, s_1)$	$\cdots$	$\delta(q_j, s_i)$	$\cdots$	$\delta(q_j, s_n)$
$\vdots$	$\vdots$		$\vdots$		$\vdots$
$SF\, q_m$	$\delta(q_m, s_1)$	$\cdots$	$\delta(q_m, s_i)$	$\cdots$	$\delta(q_m, s_n)$

où S et F sont utilisés pour marquer les états initiaux (ici : q_m) et finaux (ici : $\{ q_j, q_m \}$).

2.1.3 Définition (Fonction de transition sur les mots) Étant donné un AFD $(Q, \Sigma, \delta, s, F)$, on étend la fonction de transition δ en une fonction de transition $\widehat{\delta} : Q \times \Sigma^* \to Q$ sur les mots définie par :

$$\begin{array}{rcl} \widehat{\delta}(q, \epsilon) & \triangleq & q \\ \widehat{\delta}(q, wa) & \triangleq & \delta(\widehat{\delta}(q, w), a) \end{array}$$

2.1.4 Définition (Langage accepté par un automate) Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD.

1. M *accepte* le mot $w \in \Sigma^*$ ssi $\widehat{\delta}(s, w) \in F$. Dans le cas contraire on dit que M *rejette* w.
2. Le *langage accepté par* M est défini par :

$$\mathrm{L}(M) \triangleq \{ w \in \Sigma^* \mid \widehat{\delta}(s, w) \in F \}$$

2.1.5 Définition (Langage régulier) Un langage $L \subseteq \Sigma^*$ est dit *régulier* s'il existe un automate fini M tel que $\mathrm{L}(M) = L$.

2.1.6 Théorème Soit L un langage sur Σ. Les deux propositions suivantes sont équivalentes :

1. Il existe un AFD $M = (Q, \Sigma, \delta, s, F)$ avec $\mathrm{L}(M) = L$.
2. Il existe une grammaire régulière $G = (V, \Sigma, P, S)$ avec $\mathrm{L}(G) = L$.

$Id: notes-3.tex,v 1.15 2004/11/17 09:49:58 dbuenzli Exp $

semaine 3
lundi

Voir [HMU03], §2.3–§2.5.

2.2 Automates finis non déterministes

2.2.1 Définition (AFN)
Un automate fini non déterministe (AFN) est un quintuplet

$$M = (Q, \Sigma, \Delta, s, F)$$

où
- Q est un ensemble fini d'*états*,
- Σ est un ensemble fini de symboles, *l'alphabet (d'entrée)*,
- $\Delta : Q \times \Sigma \to \mathcal{P}(Q)$ est la *fonction (totale) de transition*,
- $s \in Q$ est l'*état initial* (ou de *départ*),
- $F \subseteq Q$ est l'ensemble des *états accepteurs* (ou *finaux*).

2.2.2 Définition (Fonction de transition sur les mots) Étant donné un AFN $(Q, \Sigma, \Delta, s, F)$ on étend la fonction de transition Δ en une fonction de transition $\widehat{\Delta} : Q \times \Sigma^* \to \mathcal{P}(Q)$ sur les mots définie par :

$$\widehat{\Delta}(p, \epsilon) \triangleq \{\, p \,\} \qquad \widehat{\Delta}(q, wa) \triangleq \bigcup\nolimits_{p \in \widehat{\Delta}(q,w)} \Delta(p, a)$$

2.2.3 Définition (Langage accepté par un AFN)
Soit $M = (Q, \Sigma, \Delta, s, F)$ un AFN.

1. M *accepte* le mot $w \in \Sigma^*$ si $\widehat{\Delta}(s, w) \cap F \neq \emptyset$.
   Dans le cas contraire on dit que M *rejette* w.
2. Le *langage accepté par* M est défini par :

$$\mathrm{L}(M) \triangleq \{\, w \in \Sigma^* \mid \widehat{\Delta}(s, w) \cap F \neq \emptyset \,\}$$

Tout automate fini non déterministe peut être converti en un automate fini déterministe en utilisant la *construction des sous-ensembles*.

2.2.4 Définition (Conversion d'un AFN en AFD) On définit la fonction D de *déterminisation* des AFN de la manière suivante :

$$\begin{aligned} \mathrm{D} : \quad & \text{AFN} \to \text{AFD} \\ & (Q, \Sigma, \Delta, s, F) \mapsto (Q_\mathrm{D}, \Sigma, \delta, s_\mathrm{D}, F_\mathrm{D}) \end{aligned}$$

avec

$$\begin{aligned} Q_\mathrm{D} &\triangleq \mathcal{P}(Q) \\ \delta(P, a) &\triangleq \textstyle\bigcup_{q \in P} \Delta(q, a) \\ s_\mathrm{D} &\triangleq \{\, s \,\} \\ F_\mathrm{D} &\triangleq \{\, P \subseteq Q \mid P \cap F \neq \emptyset \,\} \end{aligned}$$

2.2.5 Lemme Soit $M = (Q, \Sigma, \Delta, s, F)$ un AFN et $\mathrm{D}(M) = (\ldots, \delta, \ldots)$ son AFD associé. Alors, on a $\forall w \in \Sigma^* : \widehat{\delta}(\{s\}, w) = \widehat{\Delta}(s, w)$.

2.2.6 Théorème Soit M un AFN, on a $\mathrm{L}(M) = \mathrm{L}(\mathrm{D}(M)))$.

En pratique la fonction de déterminisation des automates D génère un AFD avec un grand nombre d'états inaccesibles depuis l'état de départ et donc inutiles. C'est pourquoi on utilise plutôt l'algorithme suivant qui, étant donné un AFN, construit le tableau d'un AFD en ne générant que les états accessibles.

2.2.7 Définition (Algorithme de déterminisation)
Soit $M = (Q, \Sigma, \Delta, s, F)$ un AFN. On calcule le tableau de la fonction de transition δ d'un AFD M' tel que $\mathrm{L}(M) = \mathrm{L}(M')$ de la façon suivante.

1. Dans le tableau, insérer une ligne pour l'état inital $\{s\}$. Pour tout $a \in \Sigma$, calculer les valeurs du tableau $\delta(\{s\}, a) = \Delta(s, a)$. Chacune de ces valeurs définit un état P pour lequel on rajoute une ligne au tableau (s'il n'est pas déjà défini).
2. Pour chacun des états P rajoutés, pour tout $a \in \Sigma$, il faut calculer les valeurs du tableau $\delta(P, a) = \bigcup_{q \in P} \Delta(q, a)$. Chacune de ces valeurs définit un état P' pour lequel on rajoute une ligne au tableau (s'il n'est pas déjà défini) dont on calcule les valeurs $\delta(P', a)$ pour tout $a \in \Sigma$ comme indiqué ci-dessus. On répète ce processus jusqu'à ce que le tableau se stabilise, c.-à-d., jusqu'à ce qu'il n'y ait plus de nouvel état (ligne) à insérer dans le tableau.
3. Marquer l'état $\{s\}$ par un S. Et les états P tels que $P \cap F \neq \emptyset$ par un F.

2.3 Transitions instantanées

semaine 3
mercredi

2.3.1 Notation (Transition instantanée) On utilise le symbole distingué **e** pour représenter une transition instantanée. Une *transition instantanée* (ou *transition epsilon*) est une transition d'automate qui ne consomme pas de symbole.

2.3.2 Définition (AFN$_{\mathbf{e}}$) Un *automate fini non déterministe avec transitions instantanées* (AFN$_{\mathbf{e}}$) est un quintuplet

$$M = (Q, \Sigma, \Delta, s, F)$$

où

- Q est un ensemble fini d'*états*,
- Σ est un ensemble fini de symboles, *l'alphabet (d'entrée)*,
- $\Delta : Q \times (\Sigma \cup \{\mathbf{e}\}) \to \mathcal{P}(Q)$ est la *fonction (totale) de transition*,
- $s \in Q$ est l'*état initial* (ou de *départ*),
- $F \subseteq Q$ est l'ensemble des *états accepteurs* (ou *finaux*).

2.3.3 Définition (e-fermeture) Soit un AFN$_{\mathbf{e}}$ $(Q, \Sigma, \Delta, s, F)$.
On définit l'**e**-*fermeture* $\mathrm{C}_{\mathbf{e}}(q)$ d'un état $q \in Q$, comme étant le plus petit ensemble satisfaisant les règles :

$$\frac{}{q \in \mathrm{C}_{\mathbf{e}}(q)} \qquad \frac{p \in \mathrm{C}_{\mathbf{e}}(q) \qquad p' \in \Delta(p, \mathbf{e})}{p' \in \mathrm{C}_{\mathbf{e}}(q)}$$

L' **e**-fermeture d'un ensemble d'états est définie par :

$$\mathrm{C}_{\mathbf{e}}(P) = \bigcup_{q \in P} \mathrm{C}_{\mathbf{e}}(q)$$

Noter que $\mathrm{C}_{\mathbf{e}}(q) \subseteq Q$ et $\mathrm{C}_{\mathbf{e}}(P) \subseteq Q$. □

Intuitivement, l'**e**-fermeture d'un état q donne l'ensemble de tous les états atteignables par zéro, une, ou plusieurs **e**-transitions à partir de q.

2.3.4 Définition (Fonction de transition sur les mots) Étant donné un AFN$_{\mathbf{e}}$ $M = (Q, \Sigma, \Delta, s, F)$ on étend la fonction de transition Δ en une fonction de transition $\widehat{\Delta}_{\mathbf{e}} : Q \times \Sigma^* \to \mathcal{P}(Q)$ sur les mots définie par :

$$\widehat{\Delta}_{\mathbf{e}}(q, \epsilon) \triangleq \mathrm{C}_{\mathbf{e}}(q) \qquad \widehat{\Delta}_{\mathbf{e}}(q, xa) \triangleq \bigcup\nolimits_{p \in \widehat{\Delta}_{\mathbf{e}}(q,x)} \mathrm{C}_{\mathbf{e}}\left(\Delta(p, a)\right)$$

2.3.5 Définition (Langage accepté par un AFN$_{\mathbf{e}}$)
Soit $M = (Q, \Sigma, \Delta, s, F)$ un AFN$_{\mathbf{e}}$.

1. L'automate M *accepte* le mot $w \in \Sigma^*$ si $\widehat{\Delta}_{\mathbf{e}}(s, w) \cap F \neq \emptyset$. Dans le cas contraire on dit que M *rejette* w.
2. Le *langage accepté par* M est défini par :

$$\mathrm{L}_{\mathbf{e}}(M) \triangleq \{\, w \in \Sigma^* \mid \widehat{\Delta}_{\mathbf{e}}(s, w) \cap F \neq \emptyset \,\}$$

2.3.8 Définition (Conversion de AFN$_{\mathbf{e}}$ en AFD) La fonction $\mathrm{D}_{\mathbf{e}}$ d'*élimination des* **e**-*transitions et de déterminisation* est définie par :

$$\begin{aligned} \mathrm{D}_{\mathbf{e}} : \quad & \mathrm{AFN}_{\mathbf{e}} \to \mathrm{AFD} \\ & (Q, \Sigma, \Delta, s, F) \mapsto (Q_{\mathrm{D}_{\mathbf{e}}}, \Sigma, \delta, s_{\mathrm{D}_{\mathbf{e}}}, F_{\mathrm{D}_{\mathbf{e}}}) \end{aligned}$$

avec :

$$\begin{aligned} Q_{\mathrm{D}_{\mathbf{e}}} &\triangleq \mathcal{P}(Q) \\ \delta(P, a) &\triangleq \textstyle\bigcup_{q \in P} \mathrm{C}_{\mathbf{e}}(\Delta(q, a)) \\ s_{\mathrm{D}_{\mathbf{e}}} &\triangleq \mathrm{C}_{\mathbf{e}}(s) \\ F_{\mathrm{D}_{\mathbf{e}}} &\triangleq \{\, P \subseteq Q \mid P \cap F \neq \emptyset \,\} \end{aligned}$$

On peut raffiner cette définition par $Q_{\mathrm{D}_{\mathbf{e}}} \triangleq \{\, P \in \mathcal{P}(Q) \mid P = \mathrm{C}_{\mathbf{e}}(P) \,\}$ et une adaptation correspondante de $F_{\mathrm{D}_{\mathbf{e}}}$.

2.3.9 Théorème Soit M un AFN$_{\mathbf{e}}$, on a $\mathrm{L}_{\mathbf{e}}(M) = \mathrm{L}(\mathrm{D}_{\mathbf{e}}(M))$.

$Id: notes-4.tex,v 1.23 2005/01/13 09:28:28 uwe Exp $

semaine 4
lundi

Voir [HMU03] §3.1–§3.4, [Sip97] §1.3, et [Koz97] §9–10.

2.4 Expressions régulières

2.4.1 Définition (Expressions régulières) Soit Σ un alphabet. Pour chaque $a \in \Sigma$, on se donne un nouveau symbole que l'on note $\mathbf{a}$. On se donne également les symboles $\boldsymbol{\epsilon}$, $\boldsymbol{\emptyset}$, l'opérateur unaire *, les opérateurs binaires $+, \cdot$. On définit alors l'ensemble $\mathbf{RE}_\Sigma$ des *expressions régulières sur* Σ comme étant le plus petit ensemble satisfaisant les règles :

$$\frac{a \in \Sigma}{\mathbf{a} \in \mathbf{RE}_\Sigma} \qquad \frac{}{\boldsymbol{\epsilon} \in \mathbf{RE}_\Sigma} \qquad \frac{}{\boldsymbol{\emptyset} \in \mathbf{RE}_\Sigma}$$

$$\frac{\mathbf{x} \in \mathbf{RE}_\Sigma}{(\mathbf{x}^*) \in \mathbf{RE}_\Sigma} \qquad \frac{\mathbf{x} \in \mathbf{RE}_\Sigma \quad \mathbf{y} \in \mathbf{RE}_\Sigma}{(\mathbf{x} + \mathbf{y}) \in \mathbf{RE}_\Sigma} \qquad \frac{\mathbf{x} \in \mathbf{RE}_\Sigma \quad \mathbf{y} \in \mathbf{RE}_\Sigma}{(\mathbf{x} \cdot \mathbf{y}) \in \mathbf{RE}_\Sigma}$$

2.4.2 Notation Pour alléger la notation, on introduit un ordre de priorité sur les opérateurs qui est, du plus faible au plus fort, $+$, $\cdot$, *. Cela permet de supprimer certaines parenthèses. Par exemple l'expression $((\mathbf{x}^*)+(\mathbf{y}\cdot\mathbf{z}))$ peut s'écrire sans ambiguïté $\mathbf{x}^* + \mathbf{y} \cdot \mathbf{z}$. L'opérateur $\cdot$ est souvent omis, l'expression ci-dessus peut donc se noter $\mathbf{x}^* + \mathbf{yz}$.

Lorsqu'il n'y a pas d'ambiguïté sur l'alphabet Σ utilisé, on écrit simplement $\mathbf{RE}$ l'ensemble des expression régulières sur Σ.

2.4.3 Définition (Langage dénoté par une expression régulière) Soit Σ un alphabet. Le *langage* $\mathrm{L}(\mathbf{x})$ *denoté par une expression régulière* $\mathbf{x} \in \mathbf{RE}_\Sigma$ est défini par induction structurelle sur l'expression $\mathbf{x}$ comme suit :

$$\begin{aligned} \mathrm{L}(\mathbf{a}) &\triangleq \{\, a \,\} & \mathrm{L}(\mathbf{x}+\mathbf{y}) &\triangleq \mathrm{L}(\mathbf{x}) \cup \mathrm{L}(\mathbf{y}) \\ \mathrm{L}(\boldsymbol{\epsilon}) &\triangleq \{\, \epsilon \,\} & \mathrm{L}(\mathbf{x}\cdot\mathbf{y}) &\triangleq \mathrm{L}(\mathbf{x})\,\mathrm{L}(\mathbf{y}) \\ \mathrm{L}(\boldsymbol{\emptyset}) &\triangleq \emptyset & \mathrm{L}(\mathbf{x}^*) &\triangleq \mathrm{L}(\mathbf{x})^* \end{aligned}$$

Noter que l'on confond parfois l'expression régulière avec le langage qu'elle dénote. Dès lors il arrive que l'on écrive $w \in \mathbf{x}$ au lieu de $w \in \mathrm{L}(\mathbf{x})$.

2.4.4 Remarque Les symboles d'une expression régulière dénotent un langage, c'est pour cela que l'on utilise une fonte grasse pour distinguer le langage contenant un symbole du symbole lui-même. Cependant en pratique on omet souvent la graisse, tout en restant conscient de cette différence.

2.4.5 Théorème Soit $\mathbf{x}$ une expression régulière sur un alphabet Σ.
Alors, il existe un $\mathrm{AFN}_{\boldsymbol{\epsilon}}$ M sur Σ tel que $\mathrm{L}_{\boldsymbol{\epsilon}}(M) = \mathrm{L}(\mathbf{x})$.

2.4.6 Définition (Expressions régulières équivalentes et ordonnées) Deux expressions régulières $\mathbf{x}$ et $\mathbf{y}$ sont équivalentes, noté $\mathbf{x} \eqsim \mathbf{y}$, si les langages qu'elles dénotent sont égaux :

$$\mathbf{x} \eqsim \mathbf{y} \Leftrightarrow \mathrm{L}(\mathbf{x}) = \mathrm{L}(\mathbf{y})$$

Similairement, $\mathbf{x}$ et $\mathbf{y}$ sont ordonnées, noté $\mathbf{x} \lesssim \mathbf{y}$, si le langage dénoté par $\mathbf{x}$ est inclus dans le langage dénoté par $\mathbf{y}$, i.e.

$$\mathbf{x} \lesssim \mathbf{y} \Leftrightarrow \mathrm{L}(\mathbf{x}) \subseteq \mathrm{L}(\mathbf{y})$$

2.4.7 Lemme Soit $\mathbf{x}$ et $\mathbf{y}$ deux expressions régulières. Alors :

1. $\eqsim$ est une équivalence.
2. $\mathbf{x} \lesssim \mathbf{y} \Leftrightarrow \mathbf{x} + \mathbf{y} \eqsim \mathbf{y}$
3. $\mathbf{x} \lesssim \mathbf{y} \wedge \mathbf{y} \lesssim \mathbf{x} \Rightarrow \mathbf{x} \eqsim \mathbf{y}$

2.4.8 Définition (Contextes des expressions régulières) Soit Σ un alphabet. On définit l'ensemble $\mathbf{CRE}_\Sigma$ des *contextes des expressions régulières sur* Σ comme étant le plus petit ensemble satisfaisant les règles :

$$\frac{}{[\cdot] \in \mathbf{CRE}_\Sigma} \qquad \frac{\mathbf{c} \in \mathbf{CRE}_\Sigma}{(\mathbf{c}^*) \in \mathbf{CRE}_\Sigma}$$

$$\frac{\mathbf{x} \in \mathbf{RE}_\Sigma \qquad \mathbf{c} \in \mathbf{CRE}_\Sigma}{(\mathbf{x} + \mathbf{c}) \in \mathbf{CRE}_\Sigma} \qquad \frac{\mathbf{x} \in \mathbf{RE}_\Sigma \qquad \mathbf{c} \in \mathbf{CRE}_\Sigma}{(\mathbf{c} + \mathbf{x}) \in \mathbf{CRE}_\Sigma}$$

$$\frac{\mathbf{x} \in \mathbf{CRE}_\Sigma \qquad \mathbf{c} \in \mathbf{CRE}_\Sigma}{(\mathbf{x} \cdot \mathbf{c}) \in \mathbf{CRE}_\Sigma} \qquad \frac{\mathbf{x} \in \mathbf{CRE}_\Sigma \qquad \mathbf{c} \in \mathbf{CRE}_\Sigma}{(\mathbf{c} \cdot \mathbf{x}) \in \mathbf{CRE}_\Sigma}$$

$[\cdot]$ est appelé le *trou* d'un contexte. Si $\mathbf{c} \in \mathbf{CRE}_\Sigma$ et $\mathbf{x} \in \mathbf{RE}_\Sigma$, alors $\mathbf{c}[\mathbf{x}]$ denote l'expression régulière que l'on obtient comme résultat de l'*application* de $\mathbf{c}$ à $\mathbf{x}$, par laquelle le trou est remplacée par l'expression $\mathbf{x}$.

2.4.9 Définition (Congruence) Soit R une équivalence sur $\mathbf{RE}_\Sigma$. R est une *congruence* si elle est préservée par tous les contextes $\mathbf{c} \in \mathbf{CRE}_\Sigma$, donc, si elle satisfait la régle :

$$\frac{\mathbf{x}\ R\ \mathbf{y} \qquad \mathbf{c} \in \mathbf{CRE}_\Sigma}{\mathbf{c}[\mathbf{x}]\ R\ \mathbf{c}[\mathbf{y}]}$$

2.4.10 Lemme La relation $\eqsim$ est une congruence.

2.4.11 Proposition (Simplification d'expressions régulières) On peut simplifier les expressions régulières avec les équations et inéquations suivantes :

$$\mathbf{x} + (\mathbf{y} + \mathbf{z}) \approx (\mathbf{x} + \mathbf{y}) + \mathbf{z} \tag{2.1}$$
$$\mathbf{x} + \mathbf{y} \approx \mathbf{y} + \mathbf{x} \tag{2.2}$$
$$\mathbf{x} + \boldsymbol{\emptyset} \approx \mathbf{x} \tag{2.3}$$
$$\mathbf{x} + \mathbf{x} \approx \mathbf{x} \tag{2.4}$$
$$\mathbf{x}(\mathbf{yz}) \approx (\mathbf{xy})\mathbf{z} \tag{2.5}$$
$$\mathbf{x}\boldsymbol{\epsilon} \approx \mathbf{x} \tag{2.6}$$
$$\boldsymbol{\epsilon}\mathbf{x} \approx \mathbf{x} \tag{2.7}$$
$$\mathbf{x}\boldsymbol{\emptyset} \approx \boldsymbol{\emptyset} \tag{2.8}$$
$$\boldsymbol{\emptyset}\mathbf{x} \approx \boldsymbol{\emptyset} \tag{2.9}$$
$$\mathbf{x}(\mathbf{y} + \mathbf{z}) \approx \mathbf{xy} + \mathbf{xz} \tag{2.10}$$
$$(\mathbf{x} + \mathbf{y})\mathbf{z} \approx \mathbf{xz} + \mathbf{yz} \tag{2.11}$$
$$\boldsymbol{\epsilon} + \mathbf{x}\mathbf{x}^* \approx \mathbf{x}^* \tag{2.12}$$
$$\boldsymbol{\epsilon} + \mathbf{x}^*\mathbf{x} \approx \mathbf{x}^* \tag{2.13}$$
$$\mathbf{y} + \mathbf{xz} \lesssim \mathbf{z} \Rightarrow \mathbf{x}^*\mathbf{y} \lesssim \mathbf{z} \tag{2.14}$$
$$\mathbf{y} + \mathbf{zx} \lesssim \mathbf{z} \Rightarrow \mathbf{y}\mathbf{x}^* \lesssim \mathbf{z} \tag{2.15}$$

2.4.12 Définition Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD.
Pour tout $q \in Q$, on définit le langage L_q par :

$$L_q \triangleq \mathrm{L}(\,(Q, \Sigma, \delta, q, F)\,)$$

2.4.13 Lemme Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD. Alors, pour tout $q \in Q$:

$$L_q = \Big(\bigcup\nolimits_{a \in \Sigma} \{a\} \cdot L_{\delta(q,a)} \Big) \cup \{\, \epsilon \mid q \in F \,\} \tag{LIN}$$

2.4.14 Lemme (Arden) Soit Σ un alphabet. Soit $X, A, B \subseteq \Sigma^*$ des langages sur Σ tels que $\epsilon \notin A$ et satisfaisant l'équation :

$$X = AX + B$$

Alors $X = A^*B$.

2.4.15 Définition (Conversion de AFD en RE) Soit $(Q, \Sigma, \delta, s, F)$.

1. Établir un système de $\#(Q)$ équations linéaires (du format (LIN)) qui définissent les « variables » L_q correspondants aux états $q \in Q$.
2. Par abus de notation, nous écrivons $\mathbf{a}$ à la place de $\{a\}$ et $+$ à la place de $\cup$; ceci est justifié par Définition 2.4.3.
3. Résoudre ce système de manière récursive par élimination des variables (sauf L_s) l'une après l'autre en utilisant le Lemme 2.4.14 d'Arden et la Proposition 2.4.11.
4. On résout ce système afin de trouver une expression régulière pour L_s.

2.4.16 Théorème Soit M un AFD et $\mathbf{x}$ le résultat du système d'équations de la Définition 2.4.15 appliqué à M. Alors, $\mathrm{L}(\mathbf{x}) = \mathrm{L}(M)$.

semaine 4
mercredi

2.4.17 Définition (AFNG) Un *automate fini non déterministe généralisé* (AFNG) est un quintuplet

$$M = (Q, \Sigma, \mathbf{\Delta}, s, f)$$

où

- Q est un ensemble fini d'*états*,
- Σ est un ensemble fini de symboles, *l'alphabet (d'entrée)*,
- $\mathbf{\Delta} : (Q \setminus \{f\}) \times (Q \setminus \{s\}) \to \mathbf{RE}_\Sigma$ est la *fonction (totale) de transition*,
- $s \in Q$ est l'*état initial* (ou de *départ*),
- $f \in Q$ est l'*état accepteur* (ou *final*).

2.4.18 Définition Soit $M = (Q, \Sigma, \mathbf{\Delta}, s, f)$ un AFNG.

1. M *accepte* le mot $w \in \Sigma^*$
   s'il existe $w_1, \ldots, w_k \in \Sigma^*$ et $q_0, \ldots, q_k \in Q$ tels que
   (a) $w = w_1 \cdot \ldots \cdot w_k$
   (b) $q_0 = s$ et $q_k = f$
   (c) $\forall i \in [\![1, k]\!] : w_i \in \mathrm{L}(\mathbf{\Delta}(q_{i-1}, q_i))$
2. Le *langage accepté* par M est défini par :

$$\mathrm{L}(M) \triangleq \{\, w \in \Sigma^* \mid w \text{ accepté par } M \,\}$$

2.4.19 Définition (Conversion d'un AFD en AFNG) La fonction G de *généralisation* est définie par :

$$\begin{aligned} \mathrm{G} : \quad & \text{AFD} \to \text{AFNG} \\ & (Q, \Sigma, \delta, s, F) \mapsto (Q_\mathrm{G}, \Sigma, \mathbf{\Delta}, s_\mathrm{G}, f_\mathrm{G}) \end{aligned}$$

avec $Q_\mathrm{G} \triangleq Q \cup \{\, s_\mathrm{G}, f_\mathrm{G} \,\}$ où $Q \cap \{\, s_\mathrm{G}, f_\mathrm{G} \,\} = \emptyset$ et $\mathbf{\Delta}$ telle que

$$\frac{}{\mathbf{\Delta}(s_\mathrm{G}, s) = \boldsymbol{\epsilon}} \qquad \frac{q \in \big(Q \setminus \{s\}\big) \cup \{\, f_\mathrm{G} \,\}}{\mathbf{\Delta}(s_\mathrm{G}, q) = \boldsymbol{\emptyset}}$$

$$\frac{q \in F}{\mathbf{\Delta}(q, f_\mathrm{G}) = \boldsymbol{\epsilon}} \qquad \frac{q \in Q \setminus F}{\mathbf{\Delta}(q, f_\mathrm{G}) = \boldsymbol{\emptyset}}$$

$$\frac{q, q' \in Q \qquad \{\, a \mid \delta(q, a) = q' \,\} = \{\, a_1 \ldots, a_n \,\} \neq \emptyset}{\mathbf{\Delta}(q, q') = \mathbf{a}_1 + \cdots \mathbf{a}_n}$$

$$\frac{q, q' \in Q \qquad \{\, a \mid \delta(q, a) = q' \,\} = \emptyset}{\mathbf{\Delta}(q, q') = \boldsymbol{\emptyset}}$$

□

2.4.20 Théorème Soit M un AFD. Alors $\mathrm{L}(M) = \mathrm{L}(\mathrm{G}(M))$.

2.4.21 Définition (Réduction de AFNG)
Soit $M = (Q, \Sigma, \mathbf{\Delta}, s, f)$ un AFNG et $\widehat{q} \in Q \setminus \{ s, f \}$.
La *réduction* $\mathrm{R}(M, \widehat{q})$ de M par $\widehat{q}$ est définie par

$$\mathrm{R}(M, \widehat{q}) \triangleq (Q \setminus \{ \widehat{q} \}, \Sigma, \mathbf{\Delta}', s, f)$$

avec $\mathbf{\Delta}'$ étant définie par la règle :

$$\frac{q \in Q \setminus \{ f \} \qquad q' \in Q \setminus \{ s \}}{\mathbf{\Delta}'(q, q') = \mathbf{\Delta}(q, q') + \mathbf{\Delta}(q, \widehat{q}) \cdot \big(\mathbf{\Delta}(\widehat{q}, \widehat{q})\big)^* \cdot \mathbf{\Delta}(\widehat{q}, q')}$$

□

2.4.22 Lemme Soit M un AFNG. Alors $\mathrm{L}(M) = \mathrm{L}(\mathrm{R}(M))$.

2.4.23 Définition (Algorithme de conversion de AFD en RE)
On définit l'algorithme *reduce* qui prend un AFNG en argument et le réduit jusqu'à ce qu'il ne contienne plus que deux états, à savoir l'état initial et l'état final.

reduce($(Q, \Sigma, \mathbf{\Delta}, s, f)$) :=
  si $\#(Q) > 2$
    choisir $q \in Q \setminus \{s, f\}$
    retourner *reduce*($\mathrm{R}((Q, \Sigma, \mathbf{\Delta}, s, f), q)$)
  sinon
    retourner $(Q, \Sigma, \mathbf{\Delta}, s, f)$

En utilisant *reduce*, on définit ensuite l'algorithme *afd2re* qui prend un AFD en argument et renvoie une expression régulière.

afd2re$(Q, \Sigma, \delta, s, F)$:=
  calculer $(Q', \Sigma, \mathbf{\Delta}, s', f')$=*reduce*($\mathrm{G}((Q, \Sigma, \delta, s, F))$)
  retourner $\mathbf{\Delta}(s', f')$

2.4.24 Théorème (Correction de l'algorithme de conversion) Soit M un AFD et $\mathbf{x}$ le résultat de l'exécution de *afd2re*(M). Alors $\mathrm{L}(M) = \mathrm{L}(\mathbf{x})$.

$Id: notes-5.tex,v 1.11 2005/01/13 09:28:28 uwe Exp $

semaine 5
lundi

Voir [HMU03] §4.1, §4.2.

2.5 Propriétés des langages réguliers

2.5.1 Théorème (Rappel) Soit Σ un alphabet, L un langage sur Σ. Les proposition suivantes sont équivalentes :

1. L est régulier.
2. Il existe un AFD M tel que $\mathrm{L}(M) = L$.
3. Il existe un AFN M tel que $\mathrm{L}(M) = L$.
4. Il existe un AFN$_\mathbf{e}$ M tel que $\mathrm{L}(M) = L$.
5. Il existe une expression régulière $\mathbf{x} \in \mathbf{RE}_\Sigma$ telle que $\mathrm{L}(\mathbf{x}) = L$.

2.5.2 Lemme Soit $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ et $M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ deux AFD. Posons,

$$\begin{aligned} Q_{1\times 2} &\triangleq Q_1 \times Q_2 \\ \delta_{1\times 2}((q_1, q_2), a) &\triangleq (\delta_1(q_1, a), \delta_2(q_2, a)) \\ s_{1\times 2} &\triangleq (s_1, s_2) \end{aligned}$$

Alors :

$$\forall x \in \Sigma^* : \widehat{\delta_{1\times 2}}((p, q), x) = (\widehat{\delta_1}(p, x), \widehat{\delta_2}(q, x))$$

L'automate $M_{1\otimes 2} \triangleq (Q_{1\times 2}, \Sigma, \delta_{1\times 2}, s_{1\times 2}, F_1 \times F_2)$
reconnaît le langage $L(M_{1\otimes 2}) = L(M_1) \cap L(M_2)$.
L'automate $M_{1\oplus 2} \triangleq (Q_{1\times 2}, \Sigma, \delta_{1\times 2}, s_{1\times 2}, (F_1 \times Q_2) \cup (Q_1 \times F_2))$
reconnaît le langage $L(M_{1\oplus 2}) = L(M_1) \cup L(M_2)$.

2.5.3 Théorème (Propriétés de stabilité (1))
Soit Σ un alphabet et $A, B \subseteq \Sigma^*$ des langages réguliers. Alors,

1. $A \cap B$ est régulier.
2. $A \cup B$ est régulier.
3. AB est régulier.
4. A^* est régulier.
5. $\overline{A}$ est régulier.
6. $A \setminus B$ est régulier.

2.5.4 Définition (Homomorphisme de mots) Une application $h : \Sigma^* \to \Sigma'^*$ est un *homomorphisme de mots* si

$$\begin{aligned} h(\epsilon) &= \epsilon \\ \forall x, y \in \Sigma^* : h(x \cdot y) &= h(x) \cdot h(y) \end{aligned}$$

2.5.5 Lemme Un homomorphisme de mots $h : \Sigma^* \to \Sigma'^*$ est entièrement défini par son image sur Σ, c'est à dire par la donnée d'une application $f : \Sigma \to \Sigma'^*$.

2.5.6 Théorème (Propriétés de stabilité (2)) Soit Σ, Σ' deux alphabets et $h : \Sigma^* \to \Sigma'^*$ un homomorphisme de mots. alors

7. Si $B \subseteq \Sigma'^*$ est régulier, alors $h^R(B)$ est aussi régulier.
8. Si $A \subseteq \Sigma^*$ est régulier, alors $h(A)$ est aussi régulier.

□

Notons pour $A \subseteq \Sigma^*$, $h(A)$ régulier n'implique pas toujours A régulier.

2.6 Lemme de gonflement

2.6.1 Remarque (Principe des tiroirs de Dirichlet) Si l'on range strictement plus de n chaussettes dans n tiroirs, alors il existe un tiroir qui contient au moins deux chaussettes.

2.6.2 Lemme Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD et $w \in \mathrm{L}(M)$ tel que $|w| > \#(Q)$. On décompose w en $a_1 \cdot \ldots \cdot a_n$ avec $n = |w|$ et $a_i \in \Sigma$.
Alors il existe $1 \leq i < j \leq n$ tel que :

$$\forall k \in \mathbb{N} : (a_1 \cdot \ldots \cdot a_i) \cdot (a_{i+1} \cdot \ldots \cdot a_j)^k \cdot (a_{j+1} \cdot \ldots \cdot a_n) \in \mathrm{L}(M)$$

2.6.3 Lemme (Gonflement) Soit Σ un alphabet et L un langage sur Σ. Si L est régulier, alors $\mathbf{G}(L)$ est vrai avec

$$\mathbf{G}(L) \triangleq \exists n \in \mathbb{N} : \forall w \in L : |w| \geq n \Rightarrow \left(\begin{array}{ll} \exists x, y, z \in \Sigma^* : & w = xyz \\ & \wedge\, y \neq \epsilon \\ & \wedge\, |xy| \leq n \\ & \wedge\, \forall k \in \mathbb{N} : xy^k z \in L \end{array} \right)$$

2.6.4 Corollaire Soit Σ un alphabet et L un langage sur Σ. Si $\mathbf{G}(L)$ n'est pas satisfaite, alors L n'est pas régulier. □

Il est important de noter que le lemme de gonflement énonce une propriété *nécessaire* des langages réguliers, mais pas *suffisante*. On peut donc utiliser le lemme de gonflement — le corollaire plus précisement — pour montrer qu'un langage n'est *pas* régulier. Mais on ne peut pas l'utiliser pour montrer qu'un langage *est* régulier.

$Id: notes-6.tex,v 1.14 2004/11/25 10:57:31 uwe Exp $

semaine 6

Voir [HMU03] §4.4, [Koz97] §15–16, [Sch95] §1.2.5.

2.7 Régularité par équivalences

2.7.1 Définition Soit Σ un alphabet, $L \subseteq \Sigma^*$ et R une équivalence sur Σ^*.
R est une *L-congruence* si :

1. R est une congruence pour la concaténation de symboles à droite :
   si $x \; R \; y$, alors $\forall a \in \Sigma \,.\, xa \; R \; ya$.
2. R raffine $L \times L$, c'est à dire :
   si $x \; R \; y$, alors $x \in L \iff y \in L$.

R est une *relation de Myhill-Nerode pour L* si, de plus,

3. L'index de R est fini : $\#(\Sigma^*/_R) < \infty$.

2.7.2 Définition Soit R une relation de Myhill-Nerode pour $L \subseteq \Sigma^*$.
On définit l'AFD $M_R = (Q, \Sigma, \delta, s, F)$ par :

$$\begin{aligned} Q &\triangleq \Sigma^*/_R = \{\, [x]_R \mid x \in \Sigma^* \,\} \\ s &\triangleq [\epsilon]_R \\ F &\triangleq \{\, [x]_R \mid x \in L \,\} \\ \delta :\; & \Sigma^*/_R \times \Sigma \to \Sigma^*/_R \\ & ([x]_R, a) \mapsto [xa]_R \end{aligned}$$

2.7.3 Proposition Soit R une relation de Myhill-Nerode pour L.
Alors $\mathrm{L}(M_R) = L$.

2.7.4 Définition Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD.
On définit la relation $\equiv_M \subseteq \Sigma^* \times \Sigma^*$ par :

$$x \equiv_M y \quad \text{ssi} \quad \widehat{\delta}(s, x) = \widehat{\delta}(s, y)$$

2.7.5 Lemme Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD. Alors $\#(\Sigma^*/_{\equiv_M}) \subseteq \#(Q)$.

On a l'égalité $=$ à la place de l'inclusion $\subseteq$ si tous tes états de M sont accessibles (avec Définition 2.8.1 : si $M = \mathrm{acc}(M)$).

2.7.6 Proposition Soit M un AFD.
Alors $\equiv_M$ est une relation de Myhill-Nerode pour $\mathrm{L}(M)$.

2.7.7 Définition Soit Σ un alphabet et $L \subseteq \Sigma^*$.
On définit la relation $\equiv_L \subseteq \Sigma^* \times \Sigma^*$ par :

$$x \equiv_L y \quad \text{ssi} \quad \forall z \in \Sigma^* \,.\, \big(xz \in L \iff yz \in L\big)$$

2.7.8 Lemme Soit Σ un alphabet et $L \subseteq \Sigma^*$.

1. $\equiv_L$ est une L-congruence.

2. Pour toute L-congruence R, on a $R \subseteq \equiv_L$.

C'est-à-dire $\equiv_L$ est la plus grande L-congruence.

2.7.9 Corollaire Soit M un AFD. Soit $L = \mathrm{L}(M)$.

1. $\equiv_M \subseteq \equiv_L$.
2. $\#(\Sigma^*/_{\equiv_L}) \leq \#(\Sigma^*/_{\equiv_M}) < \infty$
3. $\equiv_L$ est une relation de Myhill-Nerode pour L.

2.7.10 Théorème (Myhill-Nerode) Soit Σ un alphabet et $L \subseteq \Sigma^*$.
Alors L est régulier ssi $\#(\Sigma^*/_{\equiv_L}) < \infty$.

2.7.11 Définition Soit $L \subseteq \Sigma^*$ tel que $\equiv_L$ est d'index fini.
Alors l'automate $M_{\equiv_L}$ est dit *automate des classes d'équivalences de L*.

2.7.12 Lemme Soit $L \subseteq \Sigma^*$ tel que $\equiv_L$ est d'index fini. Soit $Q_L = \Sigma^*/_{\equiv_L}$ l'ensemble d'états de l'automate des classes d'équivalences de L.
Si $M = (Q, \Sigma, \delta, s, F)$ un AFD tel que $\mathrm{L}(M) = L$, alors $\#(Q) \geq \#(Q_L)$.

2.8 Minimisation des automates

2.8.1 Définition Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD.

1. Un état $q \in Q$ est dit *accessible*
   s'il existe un mot $w \in \Sigma^*$ tel que $\widehat{\delta}(s, w) = q$.
2. L'AFD $\mathrm{acc}(M) = (Q', \Sigma, \delta', s, F')$ est défini par :

$$\begin{aligned} Q' &\triangleq \{\, q \in Q \mid q \text{ accessible} \,\} \\ \delta' &\triangleq \delta \restriction_{Q' \times \Sigma} \\ F' &\triangleq F \cap Q' \end{aligned}$$

2.8.2 Définition Deux AFD $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ et $M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ sont dits *isomorphes*, noté $M_1 \cong M_2$, s'il existe une application bijective $f : Q_1 \to Q_2$ telle que

1. $f(s_1) = s_2$.
2. $\forall q \in Q_1 \,.\, \forall a \in \Sigma \,.\, f(\delta_1(q, a)) = \delta_2(f(q), a)$
3. $q \in F_1$ ssi $f(q) \in F_2$.

2.8.3 Définition Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD et R une équivalence sur Q.
On définit l'automate $M/_R = (Q/_R, \Sigma, \delta/_R, s/_R, F/_R)$ par :

$$\begin{aligned} Q/_R &\triangleq \{\, [p]_R \mid p \in Q \,\} \\ s/_R &\triangleq [s]_R \\ F/_R &\triangleq \{\, [p]_R \mid p \in F \,\} \\ \delta/_R :\ & Q/_R \times \Sigma \to Q/_R \\ & ([p]_R, a) \mapsto [\delta(p, a)]_R \end{aligned}$$

2.8.4 Définition Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD.
On définit la relation $\approx_M \subseteq Q \times Q$ par :

$$p \approx_M q \quad \text{ssi} \quad \forall z \in \Sigma^* \,.\, \big(\widehat{\delta}(p,z) \in F \iff \widehat{\delta}(q,z) \in F\big)$$

2.8.5 Notation Lorsqu'il n'y a pas d'ambiguïté sur l'automate M dont on parle, on écrit $\approx$ à la place de $\approx_M$.

2.8.6 Lemme Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD.

1. $p \approx_M q$ ssi $L_p = L_q$.
2. $\approx_M$ est une équivalence (sur Q). □

Il est alors possible de construire l'automate $M/_{\approx_M}$.

2.8.7 Lemme Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD et $M/_{\approx} = (Q/_{\approx}, \Sigma, \delta/_{\approx}, s/_{\approx}, F/_{\approx})$.

1. $p \in F$ ssi $[p]_{\approx} \in F/_{\approx}$.
2. Si $[p]_{\approx} = [q]_{\approx}$, alors $\forall a \in \Sigma \,.\, [\delta(p,a)]_{\approx} = [\delta(q,a)]_{\approx}$.
3. $\forall z \in \Sigma^* \,.\, \big(\widehat{\delta/_{\approx}}([p], z) = [\widehat{\delta}(p,z)]\big)$.
4. $\mathrm{L}(M/_{\approx}) = \mathrm{L}(M)$. □

2.8.8 Théorème Soit M un AFD avec $\mathrm{acc}(M) = M$. Soit $L = \mathrm{L}(M)$.
Alors, $M/_{\approx_M} \cong M_{\equiv_L}$.

2.8.9 Lemme Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD. Soit $p, q \in Q$ et $a \in \Sigma$.

1. Si $p \in F$ et $q \notin F$, alors $p \not\approx_M q$.
2. Si $\delta(p,a) \not\approx_M \delta(q,a)$, alors $p \not\approx_M q$.

2.8.10 Notation Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD. Soit $p, q \in Q$ et $a \in \Sigma$.
Nous écrivons $\{p,q\} \xrightarrow{a} \{p', q'\}$ si $\delta(p,a) = p'$ et $\delta(q,a) = q'$.
Nous définissons $\mathcal{P}_2(Q) \triangleq \{\, P \subseteq Q \mid \#(P) = 2 \,\}$.

En d'autres mots : $\mathcal{P}_2(Q)$ contient comme éléments tous les *couples non-ordonnés* $\{p,q\}$ avec $p, q \in Q$ et $p \neq q$. (Noter que $\{p,p\}$ dénote un ensemble de taille 1, donc $\{p,p\} \notin \mathcal{P}_2(Q)$.)

2.8.11 Définition (Algorithme de minimisation) Soit M un AFD.

1. Transformer M en $\mathrm{acc}(M) = (Q, \Sigma, \delta, s, F)$.
2. Calculer l'ensemble $R_{\surd} \subseteq \mathcal{P}_2(Q)$ par la procédure suivante :
   (a) Construire un tableau avec une entrée par élément de $\mathcal{P}_2(Q)$.
   (b) Marquer tous les couples $\{p,q\}$ tels que $p \in F$ et $q \notin F$:
   INIT Soit $p, q \in Q$.
   Si $p \in F$ et $q \notin F$, alors $\{p,q\} \in R_{\surd}$.
   (c) Répéter l'application de la règle suivante tant qu'elle permet de rajouter des nouveaux éléments à $R_{\surd}$:
   STEP Soit $a \in \Sigma$ et $p, p'q, q' \in Q$.
   Si $\{p,q\} \xrightarrow{a} \{p', q'\}$ et $\{p', q'\} \in R_{\surd}$, alors $\{p,q\} \in R_{\surd}$.

3. Posons $R \triangleq \{ (p,q) \mid \{ p,q \} \not\in R_{\surd} \}$.
   Noter que $R \subseteq Q \times Q$ et que $\mathrm{Id}_Q \subseteq R$.
4. L'automate minimal est l'AFD $\mathrm{acc}(M)/_R$.

2.8.12 Théorème Soit un AFD M et R la relation obtenue en appliquant l'algorithme de minimisation sur M. Alors $R = \approx_{\mathrm{acc}(M)}$.

2.8.13 Lemme Si M' est le résultat de la minimisation de M, et si M'' est le résultat de la minimisation de M', alors $M' \cong M''$.

Chapitre 3

Langages non-contextuels

$Id: notes-7.tex,v 1.17 2004/12/01 18:03:31 sbriais Exp $

semaine 7

Voir [HMU03] §5.1–2, §5.4.

3.0.14 Notation Soit $G = (V, \Sigma, P, S)$ une grammaire.
On note parfois :

$$\alpha \to \quad \beta_1 \mid \cdots \mid \beta_n$$

pour des productions $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$ dans P.

Par ailleurs on utilise les symboles $\alpha, \beta, \ldots$ pour les éléments de $(V \cup \Sigma)^*$ et $A, B, \ldots$ pour les éléments de V.

3.0.15 Lemme Soit $G = (V, \Sigma, P, S)$ une grammaire. Soit $\alpha, \alpha' \in (V \cup \Sigma)^*$.
Si $\alpha \Rightarrow_G^* \alpha'$, alors pour tout $\beta, \gamma \in (V \cup \Sigma)^*$: $\beta\alpha\gamma \Rightarrow_G^* \beta\alpha'\gamma$.

3.1 Grammaires non-contextuelles

Une grammaire non-contextuelle (c.-à-d. de type 2, voir §1.4) génère un langage dit *libre de contexte* ou *algébrique* ou *non-contextuel* (en anglais : *context-free language*).

Nous ne considérons que des grammaires $G = (V, \Sigma, P, S)$ telle que :
- si $A \in V$, alors il existe $\alpha \in (V \cup \Sigma)^*$ tel que $(A, \alpha) \in P$;
- si $(A, \alpha) \in P$, alors $\alpha \neq A$.

3.1.1 Définition (Dérivation la plus à gauche)
Une dérivation *la plus à gauche* (resp. *droite*) $\alpha_o \Rightarrow_G \ldots \Rightarrow_G \alpha_n$ est telle que pour tout $i \in [1, n]$ la dérivation directe $\alpha_{i-1} \Rightarrow_G \alpha_i$ remplace le symbole non-terminal le plus à gauche (resp. droite).

3.1.2 Lemme Soit $G = (V, \Sigma, P, S)$ une grammaire.
Si $w \in \mathrm{L}(G)$, il existe une dérivation la plus à gauche $S \Rightarrow_G^* w$.

3.2 Arbres de dérivation

3.2.1 Définition (Arbre)
On définit l'ensemble des arbres $\mathcal{T}_N$ avec nœuds N par :

$$\text{(T-FEUILLE)}\ \frac{}{(n) \in \mathcal{T}_{\{n\}}} \qquad \text{(T-NŒUD)}\ \frac{k \geq 1 \qquad t_1 \in \mathcal{T}_{N_1}\ \ldots\ t_k \in \mathcal{T}_{N_k}}{(n\ t_1 \ldots t_k) \in \mathcal{T}_{\{n\} \uplus N_1 \uplus \ldots \uplus N_k}}$$

Pour un arbre $t \in \mathcal{T}_N$, on définit :

1. la *racine* de t par :

$$\begin{aligned} \text{racine}\big((n)\big) &= n \\ \text{racine}\big((n\ t_1 \ldots t_k)\big) &= n \end{aligned}$$

2. l'ensemble des *sous-arbres* de t par :

$$\begin{aligned} \text{sous-arbres}\big((n)\big) &= \emptyset \\ \text{sous-arbres}\big((n\ t_1 \ldots t_k)\big) &= \bigcup_{i \in [\![1,k]\!]} \text{sous-arbres}(t_i) \cup \{(n\ t_1 \ldots t_k)\} \end{aligned}$$

3.2.2 Définition (Arbre de dérivation) Soit $G = (V, \Sigma, P, S)$ une grammaire. Un arbre de dérivation (t, ψ) est un arbre $t \in \mathcal{T}_N$ muni d'une fonction d'étiquetage $\psi : N \to (V \cup \Sigma \cup \{\epsilon\})$ telle que pour tout $(n\ t_1 \ldots t_k) \in \text{sous-arbres}(t)$:

1. $\psi(n) \in V$.
2. Si $\psi(n) = A$ et pour tout $i \in [\![1, k]\!]$ on a $\psi\big(\text{racine}(t_i)\big) = X_i$, alors $(A \to X_1 X_2 \cdots X_k) \in P$.
   De plus, si l'un des $X_i = \epsilon$ alors $k = 1$ (et donc $i = 1$).

3.2.3 Définition (Mot d'un arbre de dérivation) Le *mot* $\text{mot}(t, \psi)$ *généré* par un arbre de dérivation (t, ψ) est donné par :

$$\begin{aligned} \text{mot}((n), \psi) &\triangleq \psi(n) \\ \text{mot}((n\ t_1 \ldots t_k), \psi) &\triangleq \text{mot}(t_1, \psi) \cdot \ldots \cdot \text{mot}(t_k, \psi) \end{aligned}$$

Si $\alpha = \text{mot}(t, \psi)$, on dit que (t, ψ) est un *arbre de dérivation* pour α.

3.2.4 Théorème Soit $G = (V, \Sigma, P, S)$ une grammaire, $A \in V$ et $\alpha \in (V \cup \Sigma)^*$. Les deux propositions suivantes sont équivalentes :

1. Il existe une dérivation $A \Rightarrow_G^* \alpha$.
2. Il existe dans G un arbre de dérivation (t, ψ) avec $\text{racine}(t) = A$ et $\text{mot}(t, \psi) = \alpha$.

3.2.5 Corollaire (Arbre d'analyse) Soit $G = (V, \Sigma, P, S)$ une grammaire et $w \in \Sigma^*$. Alors $w \in \text{L}(G)$ ssi il existe dans G un arbre de dérivation (t, ψ), dit *arbre d'analyse*, avec $\text{racine}(t) = S$ et $\text{mot}(t, \psi) = w$.

3.3 Ambiguïté

3.3.1 Définition Une grammaire $G = (V, \Sigma, P, S)$ est *ambiguë* s'il existe un mot $w \in \mathrm{L}(G)$ avec au moins deux arbres d'analyse distincts.

3.3.2 Théorème Il n'existe pas d'algorithme qui détermine si une grammaire est ambiguë.

3.3.3 Théorème Soit $G = (V, \Sigma, P, S)$ une grammaire et $w \in \Sigma^*$.
Les deux propositions suivantes sont équivalentes :

1. Il existe au moins deux arbres d'analyse distincts pour w.
2. Il existe au moins deux dérivations la plus à gauche distinctes pour w.

3.3.4 Définition Un langage L non-contextuel est *intrinsèquement ambigu* si toute grammaire G telle que $\mathrm{L}(G) = L$ est ambiguë.

3.3.5 Théorème Il existe des langages intrinsèquement ambigus.

$Id: notes-8.tex,v 1.10 2004/12/08 11:46:23 uwe Exp $

semaine 8

Voir [HMU03] §6.1–3.

3.4 Automates à pile

3.4.1 Définition (AAP)
Un *automate à pile* (AAP) est un septuplet

$$M = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$$

où
- Q est un ensemble fini d'*états*,
- Σ est un ensemble fini de symboles, *l'alphabet d'entrée*,
- Γ est un ensemble fini de symboles, *l'alphabet de la pile*,
- $\Delta : (Q \times (\Sigma \cup \{\mathbf{e}\}) \times \Gamma) \to \mathcal{P}(Q \times \Gamma^*)$ est la *fonction de transition*,
- $q_0 \in Q$ est l'*état initial* (ou de *départ*),
- $Z_0 \in \Gamma$ est le *symbole initial* (ou de *départ*),
- $F \subseteq Q$ est l'ensemble des *états accepteurs* (ou *finaux*).

Δ doit être une application ; si $\Delta(q, s, \gamma)$ n'est pas explicitement défini, alors nous admettons que $\Delta(q, s, \gamma) = \emptyset$ de manière implicite.

3.4.2 Notation On adopte les conventions d'écriture suivantes :
- $p, q, \ldots \in Q$
- $a, b, \ldots \in \Sigma$ et $u, v \ldots \in \Sigma^*$
- $X, Y, Z \in \Gamma$ et $\alpha, \beta, \gamma \in \Gamma^*$

3.4.3 Notation (Graphe d'un AAP) On utilise la représentation des automates finis mais les flèches sont dessinées comme suit :

> Si $(q', \alpha) \in \Delta(q, a, X)$,
> on dessine une flèche étiquetée par $a, X/\alpha$ de q à q'.

3.4.4 Définition (Configurations et calculs)
Soit $M = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$ un AAP.

1. Une *configuration* de M est un triplet $(q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$.
2. Soit $\vdash_M$ la relation sur les configurations de M définie par :

$$\begin{aligned} (q, aw, X\beta) &\vdash_M (q', w, \alpha\beta) && \text{si } (q', \alpha) \in \Delta(q, a, X) \\ (q, w, X\beta) &\vdash_M (q', w, \alpha\beta) && \text{si } (q', \alpha) \in \Delta(q, \mathbf{e}, X) \end{aligned}$$

   Si $c \vdash_M c'$ alors la paire (c, c') est une *étape de calcul* de M.
3. Un *calcul de M pour w* est une séquence de configurations de M,

$$(q_0, w, Z_0) \vdash_M (q_1, w_1, \gamma_1) \vdash_M \ldots \vdash_M (q_i, w_i, \gamma_i) \vdash_M \ldots$$

   où (q_0, w, Z_0) représente la *configuration initiale* du calcul. □

Lorsque l'automate M dont on parle est clair dans le contexte, on écrit simplement $\vdash$ à la place de $\vdash_M$.

3.4.5 Lemme Soit $M = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$ un AAP.

1. Si $(q, x, \alpha) \vdash^* (q', y, \beta)$,
   alors pour tout $w \in \Sigma^*$ et $\gamma \in \Gamma^*$: $(q, xw, \alpha\gamma) \vdash^* (q', yw, \beta\gamma)$.
2. Si $(q, xw, \alpha) \vdash^* (q', yw, \beta)$,
   alors $(q, x, \alpha) \vdash^* (q', y, \beta)$.

3.4.6 Définition (Langages acceptés par un AAP)
Soit $M = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$ un AAP.

1. Le *langage accepté par M par état final* est défini par :

$$\mathrm{L}_{\text{état}}(M) \triangleq \{\, w \in \Sigma^* \mid \exists q \in F, \alpha \in \Gamma^* \,.\, (q_0, w, Z_0) \vdash^* (q, \epsilon, \alpha) \,\}$$

2. Le *langage accepté par M par pile vide* est défini par :

$$\mathrm{L}_{\text{pile}}(M) \triangleq \{\, w \in \Sigma^* \mid \exists q \in Q, \alpha \in \Gamma^* \,.\, (q_0, w, Z_0) \vdash^* (q, \epsilon, \epsilon) \,\}$$

3.4.7 Notation Lorsque l'on s'intéresse uniquement au langage accepté par pile vide d'un automate, on ne précise pas l'ensemble des états finaux. Un AAP est appelé AAP_{pile} ou $\text{AAP}_{\text{état}}$ en fonction du mécanisme d'acceptation choisi.

3.4.8 Lemme Soit $M = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0)$ un AAP_{pile}.
Soit $M' = (Q \cup \{\, p_0, p_{\text{f}} \,\}, \Sigma, \Gamma \cup \{X_0\}, \Delta', p_0, X_0, \{\, p_{\text{f}} \,\})$ l'$\text{AAP}_{\text{état}}$
tel que $Q \cap \{\, p_0, p_{\text{f}} \,\} = \emptyset = \Gamma \cap \{X_0\}$ et

$$\begin{aligned}
\Delta' :\ & (p_0, \mathbf{e}, X_0) \mapsto \{\, (q_0, Z_0 X_0) \,\} \\
& (q, a, Y) \mapsto \Delta(q, a, Y) && \text{si } q \in Q, Y \in \Gamma, a \in \Sigma \cup \{\mathbf{e}\} \\
& (q, \mathbf{e}, X_0) \mapsto \{\, (p_{\text{f}}, \epsilon) \,\} && \text{si } q \in Q
\end{aligned}$$

Alors $\mathrm{L}_{\text{pile}}(M) = \mathrm{L}_{\text{état}}(M')$.

3.4.9 Lemme Soit $M = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$ un $\text{AAP}_{\text{état}}$.
Soit $M' = (Q \cup \{\, p_0, p \,\}, \Sigma, \Gamma \cup \{X_0\}, \Delta', p_0, X_0)$ l'AAP_{pile}
tel que $Q \cap \{\, p_0, p \,\} = \emptyset = \Gamma \cap \{X_0\}$ et

$$\begin{aligned}
\Delta' :\ & (p_0, \mathbf{e}, X_0) \mapsto \{\, (q_0, Z_0 X_0) \,\} \\
& (q, a, Y) \mapsto \Delta(q, a, Y) && \text{si } q \in Q, Y \in \Gamma, a \in \Sigma \\
& (q, \mathbf{e}, Y) \mapsto \Delta(q, \mathbf{e}, Y) && \text{si } q \in Q \backslash F, Y \in \Gamma \\
& (q, \mathbf{e}, Y) \mapsto \Delta(q, \mathbf{e}, Y) \cup \{\, (p, \epsilon) \,\} && \text{si } q \in F, Y \in \Gamma \\
& (q, \mathbf{e}, X_0) \mapsto \{\, (p, \epsilon) \,\} && \text{si } q \in F \\
& (p, \mathbf{e}, Y) \mapsto \{\, (p, \epsilon) \,\} && \text{si } Y \in \Gamma \cup \{X_0\}
\end{aligned}$$

Alors $\mathrm{L}_{\text{état}}(M) = \mathrm{L}_{\text{pile}}(M')$.

3.4.10 Corollaire Soit Σ un alphabet et $L \subseteq \Sigma^*$.
Alors les deux propositions suivantes sont équivalentes :

1. Il existe un $\mathrm{AAP}_{\text{pile}}$ M tel que $L = \mathrm{L}_{\text{pile}}(M)$.
2. Il existe un $\mathrm{AAP}_{\text{état}}$ M tel que $L = \mathrm{L}_{\text{état}}(M)$.

3.4.11 Lemme Soit $G = (V, \Sigma, P, S)$ une grammaire non-contextuelle, et $\Gamma \triangleq \Sigma \cup V$. Soit $M \triangleq (\{q\}, \Sigma, \Gamma, \Delta, q, S)$ l'$\mathrm{AAP}_{\text{pile}}$ défini par :

$$\begin{aligned}\Delta : (\{q\} \times (\Sigma \cup \{\mathbf{e}\}) \times \Gamma) &\to \mathcal{P}(\{q\} \times \Gamma^*) \\ (q, \mathbf{e}, A) &\mapsto \{\, (q, \alpha) \mid (A \to_G \alpha) \in P \,\} \quad \text{si } A \in V \\ (q, a, a) &\mapsto \{\, (q, \epsilon) \,\} \quad \text{si } a \in \Sigma\end{aligned}$$

Alors $\mathrm{L}_{\text{pile}}(M) = \mathrm{L}(G)$.

3.4.12 Lemme Soit $M = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0)$ un $\mathrm{AAP}_{\text{pile}}$.
Soit $G \triangleq (V, \Sigma, P, S)$ une grammaire non-contextuelle définie par :

1. S un nouveau symbole,
2. $V \triangleq \{\, [pZq] \mid p, q \in Q \wedge Z \in \Gamma \,\}$,
3. $\begin{aligned} P \triangleq\ & \{\, S \to_G [q_0 Z_0 q] \mid q \in Q \,\} \\ & \cup \{\, [qZr_k] \to_G s[rY_1r_1][r_1Y_2r_2]\cdots[r_{k-1}Y_kr_k] \\ & \qquad \mid (r, Y_1 \cdots Y_k) \in \Delta(q, s, Z) \ \wedge\ r_1 \ldots, r_k \in Q \ \wedge\ s \in \Sigma \,\} \\ & \cup \{\, [qZr_k] \to_G [rY_1r_1][r_1Y_2r_2]\cdots[r_{k-1}Y_kr_k] \\ & \qquad \mid (r, Y_1 \cdots Y_k) \in \Delta(q, \mathbf{e}, Z) \ \wedge\ r_1 \ldots, r_k \in Q \,\} \end{aligned}$

Alors $\mathrm{L}(G) = \mathrm{L}_{\text{pile}}(M)$.

3.4.13 Corollaire Soit Σ un alphabet et $L \subseteq \Sigma^*$.
Alors les deux propositions suivantes sont équivalentes :

1. Il existe un $\mathrm{AAP}_{\text{pile}}$ M tel que $L = \mathrm{L}(M)$.
2. Il existe une grammaire non-contextuelle G telle que $L = \mathrm{L}(G)$.

$Id: notes-9.tex,v 1.13 2004/12/13 16:20:38 uwe Exp $

semaine 9

Voir [HMU03] §7.1 et §7.2.

3.5 Formes normales des grammaires non-contextuelles

3.5.1 Définition (Symbole potentiellement vide)
Soit $G = (V, \Sigma, P, S)$ une grammaire non-contextuelle.
Un symbole $A \in V$ est *potentiellement vide*, si $A \Rightarrow_G^* \epsilon$.

3.5.2 Lemme Soit $G = (V, \Sigma, P, S)$ une grammaire non-contextuelle.
Soit $V_\epsilon \subseteq V$ défini de manière inductive par :

$$\frac{(A \to \epsilon) \in P}{A \in V_\epsilon} \qquad \frac{(A \to B_1 B_2 \cdots B_k) \in P \qquad \forall i \in [1, k] \,.\, B_i \in V_\epsilon}{A \in V_\epsilon}$$

Alors A est potentiellement vide ssi $A \in V_\epsilon$.

3.5.3 Lemme (Elimination des ϵ-productions)
Soit $G = (V, \Sigma, P, S)$ une grammaire non-contextuelle, et $\Gamma = V \cup \Sigma$.
Soit V_ϵ l'ensemble des symboles potentiellement vides.
Soit $G' = (V, \Sigma, P', S)$ avec P' défini de manière inductive par :

$$\frac{(A \to \alpha) \in P \qquad \alpha \in \Gamma^+}{(A \to \alpha) \in P'} \qquad \frac{(A \to \gamma B \gamma') \in P' \qquad B \in V_\epsilon \qquad \gamma\gamma' \in \Gamma^+}{(A \to \gamma\gamma') \in P'}$$

Alors $\mathrm{L}(G') = \mathrm{L}(G) \setminus \{\epsilon\}$.

3.5.4 Définition (Production unitaire)
Soit $G = (V, \Sigma, P, S)$ une grammaire non-contextuelle.
Une production $(A \to \alpha) \in P$ est dite *unitaire* si $\alpha \in V$.
Deux variables $A, B \in V$ forment une *paire unitaire* (A, B) dans G,
s'il existe une dérivation $A \Rightarrow_G^* B$ qui utilise uniquement des productions unitaires.

3.5.5 Lemme Soit $G = (V, \Sigma, P, S)$ une grammaire non-contextuelle.
Soit $\mathrm{U}_G \subseteq V \times V$ défini de manière inductive par :

$$\frac{A \in V}{(A, A) \in \mathrm{U}_G} \qquad \frac{(A, B) \in \mathrm{U}_G \qquad (B \to C) \in P}{(A, C) \in \mathrm{U}_G}$$

Alors, (A, B) est une paire unitaire ssi $(A, B) \in \mathrm{U}_G$.

3.5.6 Lemme (Élimination des productions unitaires)
Soit $G = (V, \Sigma, P, S)$ une grammaire non-contextuelle, et U_G l'ensemble des paires unitaires de G. Soit $G' = (V, \Sigma, P', S)$ la grammaire dont les productions P' sont telles que :

$$\frac{(A, B) \in \mathrm{U}_G \qquad (B \to \alpha) \in P \qquad \alpha \notin V}{(A \to \alpha) \in P'}$$

Alors $\mathrm{L}(G') = \mathrm{L}(G)$.

3.5.7 Définition (Symboles utiles)
Soit $G = (V, \Sigma, P, S)$ une grammaire non-contextuelle et $\Gamma = V \cup \Sigma$.

1. Un symbole $X \in \Gamma$ est *génératif*
   s'il existe $w \in \Sigma^*$ tel que $X \Rightarrow_G^* w$.
2. Un symbole $X \in \Gamma$ est *accessible*
   s'il existe $\alpha, \beta \in \Gamma^*$ tel que $S \Rightarrow_G^* \alpha X \beta$.
3. Un symbole $X \in \Gamma$ est *utile* s'il est génératif et accessible.
   Sinon il est *inutile*.

3.5.8 Lemme
Soit $G = (V, \Sigma, P, S)$ une grammaire non-contextuelle, et $\Gamma = V \cup \Sigma$.
Soit $\Gamma_{\text{gen}} \subseteq \Gamma$ défini de manière inductive par :

$$\frac{s \in \Sigma}{s \in \Gamma_{\text{gen}}} \qquad \frac{(A \to B_1 B_2 \cdots B_k) \in P \qquad \forall i \in [\![1, k]\!] \,.\, B_i \in \Gamma_{\text{gen}}}{A \in \Gamma_{\text{gen}}}$$

Soit $\Gamma_{\text{acc}} \subseteq \Gamma$ défini de manière inductive par :

$$\frac{}{S \in \Gamma_{\text{acc}}} \qquad \frac{A \in \Gamma_{\text{acc}} \qquad (A \to B_1 B_2 \cdots B_k) \in P \qquad i \in [\![1, k]\!]}{B_i \in \Gamma_{\text{acc}}}$$

Alors

1. $X \in \Gamma$ est *génératif* ssi $X \in \Gamma_{\text{gen}}$.
2. $X \in \Gamma$ est *accessible* ssi $X \in \Gamma_{\text{acc}}$.

Noter que, dans le calcul des symboles génératifs, le cas $k = 0$ traite les productions de la forme $A \to \epsilon$.

3.5.9 Lemme Soit $G = (V, \Sigma, P, S)$ une grammaire non-contextuelle et Γ_{gen} tel que défini au lemme 3.5.8. Si $\mathrm{L}(G) = \emptyset$, alors $S \notin \Gamma_{\text{gen}}$.

3.5.10 Lemme (Élimination des symboles inutiles)
Soit $G \triangleq (V, \Sigma, P, S)$ une grammaire non-contextuelle
telle que $\mathrm{L}(G) \neq \emptyset$ et Γ_{gen} tel que défini au lemme 3.5.8.
Soit $G' \triangleq (V', \Sigma, P', S)$ avec $V' \triangleq V \cap \Gamma_{\text{gen}}$, $\Gamma' \triangleq V' \cup \Sigma$, et P' tel que :

$$\frac{(A \to \alpha) \in P \qquad A \in V' \qquad \alpha \in \Gamma'^*}{(A \to \alpha) \in P'}$$

Soit Γ'_{acc} tel que défini (pour G') au lemme 3.5.8, et $G'' \triangleq (V'', \Sigma'', P'', S)$ avec $V'' \triangleq V' \cap \Gamma'_{\text{acc}}$, $\Sigma'' \triangleq \Sigma \cap \Gamma'_{\text{acc}}$, $\Gamma'' \triangleq V'' \cup \Sigma''$ et P'' tel que :

$$\frac{(A \to \alpha) \in P' \qquad A \in V'' \qquad \alpha \in \Gamma''^*}{(A \to \alpha) \in P''}$$

Alors :

1. G'' ne contient que de symboles utiles.
2. $\mathrm{L}(G) = \mathrm{L}(G') = \mathrm{L}(G'')$.

3.5.11 Proposition Soit G une grammaire non-contextuelle.
Soit G' la grammaire résultant de (dans cet ordre) :

1. l'élimination des ϵ-productions selon le lemme 3.5.3,
2. puis de l'élimination des productions unitaires selon le lemme 3.5.6,
3. et de l'élimination des symboles inutiles selon le lemme 3.5.10.

Alors, G' ne contient aucune ϵ-production, aucune production unitaire, et aucun symboles inutile.

Noter que si les trois transformations ne sont pas appliquées dans l'ordre donné alors la proposition n'est pas garantie.

3.5.12 Définition (Forme normale de Chomsky)
Soit $G = (V, \Sigma, P, S)$ une grammaire.
G est sous *forme normale de Chomsky* (FNC) si :

1. $V \cup \Sigma$ ne contient pas de symboles inutiles.
2. Les productions de P sont de l'une des formes suivantes :
   (a) $A \to BC$ avec $A, B, C \in V$
   (b) $A \to a$ avec $A \in V$ et $a \in \Sigma$.

3.5.13 Théorème (Chomsky)
Soit G une grammaire non-contextuelle telle que $\mathrm{L}(G) \setminus \{\epsilon\} \neq \emptyset$.
Alors, il existe une grammaire G' en FNC telle que $\mathrm{L}(G') = \mathrm{L}(G) \setminus \{\epsilon\}$.

3.6 Lemme de gonflement

3.6.1 Définition La *hauteur* $\mathrm{h}(t)$ d'un arbre $t \in \mathcal{T}_N$ est défini par :

$$\begin{aligned} \mathrm{h}\big((n)\big) &= 0 \\ \mathrm{h}\big((n\, t_1 \dots t_k)\big) &= 1 + \max\{\, \mathrm{h}(t) \mid t \in \{\, t_1 \dots t_k) \,\} \,\} \end{aligned}$$

3.6.2 Lemme Soit $G = (V, \Sigma, P, S)$ une grammaire en FNC.
Soit (t, ψ) un arbre de dérivation pour $w \in \Sigma$, et $n = \mathrm{h}(t)$.
Alors, $|w| \leq 2^{n-1}$.

3.6.3 Théorème Soit Σ un alphabet et L un langage sur Σ.
Si L est non-contextuel, alors :

$$\exists n \in \mathbb{N} : \forall z \in L : |z| \geq n \Rightarrow \left(\begin{array}{ll} \exists u, v, w, x, y \in \Sigma^* : & z = uvwxy \\ & \wedge\ vx \neq \epsilon \\ & \wedge\ |vwx| \leq n \\ & \wedge\ \forall i \in \mathbb{N} : uv^i wx^i y \in L \end{array} \right)$$

$Id: notes-10.tex,v 1.8 2004/12/21 13:25:57 uwe Exp $

semaine 10

3.7 Automates à pile déterministes

Voir [HMU03] §6.4.

3.7.1 Définition (AAPD)
Un *automate à pile déterministe* (AAPD) est un automate à pile

$$M = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$$

qui satisfait, pour tout $q \in Q$, $a \in \Sigma$ et $X \in \Gamma$:

$$\#(\Delta(q, a, X)) \;+\; \#(\Delta(q, \mathbf{e}, X)) \quad \leq \quad 1$$

3.7.2 Définition (Propriété préfixe) Soit L un langage. On dit que L satisfait la *propriété préfixe* si :

$$\forall x, y \in L \,.\, \big(x \neq y \implies \neg\exists w \in \Sigma^* \,.\, (\, x = yw \vee y = xw \,)\big)$$

3.7.3 Théorème

1. Si L est régulier, alors il existe un AAPD M tel que $\mathrm{L}_{\text{état}}(M) = L$.
2. Il existe L régulier tel que pour tout AAPD M : $\mathrm{L}_{\text{pile}}(M) \neq L$.

3.7.4 Théorème Soit L un langage. Les deux propositions suivantes sont équivalentes :

1. Il existe un AAPD M tel que $\mathrm{L}_{\text{pile}}(M) = L$.
2. Il existe un AAPD M tel que $\mathrm{L}_{\text{état}}(M) = L$ et L satisfait la propriété préfixe.

3.7.5 Définition (Langage non-contextuel déterministe) Un langage L est dit *non-contextuel déterministe* s'il existe un AAPD tel que $\mathrm{L}_{\text{état}}(M) = L$.

3.7.6 Théorème Soit L un langage.

L régulier $\Rightarrow$ L non-contextuel déterministe $\Rightarrow$ L non-contextuel.

3.7.7 Théorème Soit M un AAPD. Il existe une grammaire non-ambiguë G telle que $\mathrm{L}(G) = \mathrm{L}(M)$.

3.8 Propriétés de langages non-contextuelles

Voir [HMU03] §7.3, et [Koz97] §G.

3.8.1 Définition (Substitution) Soit Σ, Σ' deux alphabets. Une *substitution* est une application $\sigma : \Sigma \to \mathcal{P}(\Sigma'^*)$ qui associe à toute lettre de Σ un langage sur Σ'.

On étend σ au mots de Σ et au langages sur Σ de la façon suivante :

$$\begin{array}{ll} \sigma : & \Sigma^* \to \mathcal{P}(\Sigma'^*) \\ & a_1 \cdots a_n \mapsto \sigma(a_1) \cdot \ldots \cdot \sigma(a_n) \end{array} \qquad \begin{array}{ll} \sigma : & \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma'^*) \\ & L \mapsto \bigcup_{w \in L} \sigma(w) \end{array}$$

3.8.2 Théorème Soit Σ, Σ' deux alphabets et L un langage non-contextuel sur Σ. Soit $\sigma : \Sigma \to \mathcal{P}(\Sigma'^*)$ une *substitution*.

Si pour tout $a \in \Sigma$, $\sigma(a)$ est un langage non-contextuel. Alors $\sigma(L)$ est un langage non-contextuel.

3.8.3 Théorème (Propriétés de stabilité (1)) Soient A, B deux langages non-contextuels sur Σ. Alors :

1. $A \cup B$ est non-contextuel.
2. AB est non-contextuel.
3. A^* est non-contextuel.

et, si $C \subseteq \Sigma^*$ est un langage régulier :

4. $A \cap C$ est non-contextuel.
5. $A \setminus C$ est non-contextuel.

En revanche :

6. $A \cap B$ n'est pas forcément non-contextuel.
7. $\overline{A}$ n'est pas forcément non-contextuel.
8. $A \setminus B$ n'est pas forcément non-contextuel.

Soit Σ, Σ' deux alphabets et $h : \Sigma^* \to \Sigma'^*$ un homomorphisme de mots alors :

9. Si $A \subset \Sigma^*$ est non-contextuel alors $h(A)$ est non-contextuel.
10. Si $B \subset \Sigma'^*$ est non-contextuel alors $h^R(B)$ est non-contextuel.

3.8.4 Théorème (Propriétés de stabilité (2)) Soient A, B deux langages non-contextuels déterministes sur Σ. Alors :

1. $\overline{A}$ est non-contextuel déterministe.

En revanche :

2. A^* n'est pas forcément non-contextuel déterministe.
3. $A \cup B$ n'est pas forcément non-contextuel déterministe.
4. $A \cap B$ n'est pas forcément non-contextuel déterministe.
5. $A \setminus B$ n'est pas forcément non-contextuel déterministe.

3.8.5 Définition Soit $\mathrm{Par}_n = \left\{ \overset{1}{[}, \overset{1}{]}, \ldots, \overset{n}{[}, \overset{n}{]} \right\}$ un alphabet qui contient n types de parenthèses. Le langage Equil_n sur Par_n est l'ensemble des mots dont les parenthèses sont bien équilibrées. Equil_n est généré par la grammaire suivante :

$$S \to \overset{1}{[}S\overset{1}{]} \mid \overset{2}{[}S\overset{2}{]} \mid \cdots \mid \overset{n}{[}S\overset{n}{]} \mid SS \mid \epsilon$$

3.8.6 Théorème (Chomsky-Schützenberger)
Soit L un langage non-contextuel sur Σ. Il existe un langage R régulier, $n \geq 0$, et un homomorphisme $h : \mathrm{Par}_n \to \Sigma$ tels que :

$$L = h(\mathrm{Equil}_n \cap R)$$

Chapitre 4

Langages récursifs et enumérables

$Id: notes-11.tex,v 1.17 2005/01/13 09:28:28 uwe Exp $

semaine 11

4.1 Les machines de Turing

Voir [HMU03] §8 (en particulier §8.2).

4.1.1 Définition (MT) Une *machine de Turing* (MT) est un septuplet

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \mathrm{B}, F)$$

où
- Q est un ensemble fini d'*états*,
- Σ est un ensemble fini de symboles, *l'alphabet d'entrée*,
- $\Gamma \supset \Sigma$ est un ensemble fini de symboles, *l'alphabet du ruban*,
- $\delta : (Q \times \Gamma) \to (Q \times \Gamma \times \{-1, 0, 1\})$ est la *fonction de transition*,
- $q_0 \in Q$ est l'*état initial* (ou de *départ*),
- $\mathrm{B} \in \Gamma \setminus \Sigma$ est le *symbole blanc*,
- $F \subseteq Q$ est l'ensemble des *états accepteurs* (ou *finaux*).

En général, δ est une fonction partielle.
Il existe aussi des versions des MT où δ est de type

$$\begin{array}{rl} \textit{multi/k-rubans}: & (Q \times \Gamma^k) \to (Q \times \Gamma^k \times \{-1, 0, 1\}^k) \\ \textit{non-déterministe}: & (Q \times \Gamma) \to \mathcal{P}(Q \times \Gamma \times \{-1, 0, 1\}) \end{array}$$

4.1.2 Notation On adopte les conventions d'écriture suivantes :
- $p, q, \ldots \in Q$
- $a, b, \ldots \in \Sigma$ et $u, v \ldots \in \Sigma^*$
- $X, Y, Z \in \Gamma$ et $\alpha, \beta, \gamma \in \Gamma^*$

4.1.3 Notation (Graphe d'une MT) On utilise la représentation des AAP mais les flèches sont dessinées comme suit :

Si $\delta(q, X) = (q', Y, d)$,
on dessine une flèche étiquetée par $X/Y, d$ de q à q'.

4.1.4 Définition (Configurations et calculs)
Soit $M = (Q, \Sigma, \Gamma, \delta, q_0, \mathrm{B}, F)$ une MT.

1. Une *configuration* de M est un triplet (q, i, rub) où
   - $q \in Q$ est l'état dans lequel se trouve la machine M,
   - $i \in \mathbb{Z}$ est l'emplacement courant de la tête de lecture/écriture, et
   - $\mathrm{rub} : \mathbb{Z} \to \Gamma$ est une application représentant l'état du ruban telle que $\mathrm{rub}(k) \neq \mathrm{B}$ pour au plus un nombre fini d'entiers k.

   Si $q \in F$, on dit que (q, i, rub) est une *configuration acceptante* de M.
2. Soit $c = (q, i, \mathrm{rub})$ une configuration de M et $X = \mathrm{rub}(i)$.
   Si $\delta(q, X) = (q', Y, d)$,
   alors la configuration c *se réduit en une étape de calcul*
   en la configuration $c' = (q', i + d, \mathrm{rub}\{i \mapsto Y\})$, noté $c \vdash_M c'$, avec

$$\begin{array}{lll} \mathrm{rub}\{i \mapsto Y\} : \mathbb{Z} \to \Gamma & & \\ \qquad\qquad k \mapsto Y & \text{si } k = i \\ \qquad\qquad k \mapsto \mathrm{rub}(k) & \text{sinon} \end{array}$$

   Comme d'habitude, on note $\vdash_M^*$ la fermeture réflexive et transitive de la relation $\vdash_M$ ainsi définie.
3. On identifie tout mot $w \in \Gamma^*$ avec le ruban $w : \mathbb{Z} \to \Gamma$ défini par :

$$\begin{array}{lll} w : \mathbb{Z} \to \Gamma & & \\ \qquad k \mapsto (w)_k & \text{si } 1 \leq k \leq |w| \\ \qquad k \mapsto \mathrm{B} & \text{sinon} \end{array}$$

   Autrement dit, le ruban w est un ruban rempli de caractères blancs si ce n'est que le mot w est écrit sur celui-ci à partir de la position 1.
4. Soit $w \in \Sigma^*$. Un *calcul* de M pour w est une séquence

$$c_0 \vdash_M c_1 \vdash_M \ldots \vdash_M c_i \vdash_M \ldots$$

   où pour tout k, c_k est une configuration de M et $c_0 = (q_0, 1, w)$ est la configuration initiale du calcul.
5. Soit c un configuration de M. S'il n'existe aucune configuration c' telle que $c \vdash_M c'$, on note $c \nvdash_M$.

Lorsque la machine M dont on parle est claire dans le contexte, on écrit simplement $\vdash$ à la place de $\vdash_M$.

4.1.5 Définition (Langage accepté par une MT)
Soit $M = (Q, \Sigma, \Gamma, \delta, q_0, \mathrm{B}, F)$ une MT.

1. M accepte le mot $w \in \Sigma^*$
   s'il existe une configurations c acceptante telle que $(q_0, 1, w) \vdash_M^* c$.

2. Le *langage accepté par* M est défini par :

$$\mathrm{L}(M) \triangleq \{\, w \in \Sigma^* \mid w \text{ accepté par } M \,\}$$

4.1.6 Théorème Soit Σ un alphabet et $L \subseteq \Sigma^*$.
Alors les deux propositions suivantes sont équivalentes :

1. Il existe une MT M telle que $L = \mathrm{L}(M)$.
2. Il existe une grammaire G de type 0 telle que $L = \mathrm{L}(G)$.

4.2 Calculabilité

Voir [HMU03] §8.2.6 et [Sch95] §2.2.

4.2.1 Définition (Arrêt)
Soit $M = (Q, \Sigma, \Gamma, \delta, q_0, \mathrm{B}, F)$ une MT.
On dit que M s'arrête pour $w \in \Sigma^*$
s'il existe une configuration c telle que $(q_0, 1, w) \vdash_M^* c \nvdash_M$.
Si M s'arrête pour tout $w \in \Sigma^*$, alors M est dite *totale*.

4.2.2 Notation Soit $M = (Q, \Sigma, \Gamma, \delta, q_0, \mathrm{B}, F)$ une MT.

1. Si l'on ne s'intéresse pas au langage accepté par une MT M, mais seulement à la question de l'arrêt, on peut poser $F = \emptyset$ ou bien omettre F.
2. Si l'on veut imposer qu'une MT s'arrête au moment où elle accepte (c-à-d, où elle passe dans un état accepteur), on peut restreindre le type de δ à

$$((Q \setminus F) \times \Gamma) \to (Q \times \Gamma \times \{-1, 0, 1\})$$

4.2.3 Définition (Turing-calculabilité)

1. Une fonction $f : \Sigma^* \to \Sigma^*$ est dite *Turing-calculable*,
s'il existe une MT $M = (Q, \Sigma, \Gamma, \delta, q_0, \mathrm{B}, \emptyset)$
telle que pour tout $x, y \in \Sigma^*$

$$f(x) = y$$
ssi
$$(q_0, 1, x) \vdash^* (q, 1, y) \nvdash$$

2. Soit $\mathrm{rep}(n)$ la suite $1 \cdots 1$ de longueur $n \in \mathbb{N}$.
Une fonction $f : \mathbb{N}^k \to \mathbb{N}$ est dite *Turing-calculable*,
s'il existe une MT $M = (Q, \{1\}, \{1, \#\} \cup \Gamma, \delta, q_0, \mathrm{B}, \emptyset)$
telle que pour tout $n_1 \ldots, n_k, m \in \mathbb{N}$

$$f(n_1 \ldots, n_k) = m$$
ssi
$$(q_0, 1, \mathrm{rep}(n_1) \# \cdots \# \mathrm{rep}(n_k)) \vdash^* (q, 1, \mathrm{rep}(m)) \nvdash$$

4.2.4 Thèse (Church-Turing) Tout *algorithme*, c-à-d toute *procédure effective*, peut être représenté par une MT.

4.3 Décidabilité des langages

Voir [HMU03] §8.2.6 et §9.2.1–2.

4.3.1 Définition Soit L un langage sur l'alphabet Σ.

1. L est dit *décidable* (ou : *récursif*)
   s'il existe une MT M telle que $\mathrm{L}(M) = L$ et M est totale.
2. L est dit *semi-décidable* (ou : *récursivement énumerable*, ou : *r.e.*)
   s'il existe une MT M telle que $\mathrm{L}(M) = L$.
3. L est dit *co-semi-décidable* (ou : *co-r.e.*)
   si son complément $\overline{L}$ est semi-décidable.
4. L est dit *indécidable*
   si L n'est pas décidable.
5. L est dit *non-semi-décidable* (ou : *non-r.e.*)
   si L n'est pas semi-décidable.

4.3.2 Proposition Soit L un langage sur l'alphabet Σ.

1. Si L est décidable, alors son complément $\overline{L}$ est aussi décidable.
2. Si L et $\overline{L}$ sont semi-décidables, alors L est décidable.

$Id: notes-12.tex,v 1.10 2005/01/17 10:10:45 uwe Exp $

semaine 12

Voir [HMU03] §9.1.

4.4 Codage des entiers

4.4.1 Définition (Représentation binaire des entiers) Soit $\mathrm{bin} : \mathbb{N} \to \{0,1\}^*$ la fonction définie par induction sur les entiers naturels par :

$$\begin{aligned} \mathrm{bin}(0) &\triangleq \epsilon \\ \mathrm{bin}(2n) &\triangleq \mathrm{bin}(n) \cdot 0 && \text{pour } n > 0 \\ \mathrm{bin}(2n+1) &\triangleq \mathrm{bin}(n) \cdot 1 && \text{pour } n \geq 0 \end{aligned}$$

4.4.2 Lemme

1. bin est injective, mais pas surjective.
2. $\mathrm{bin}(\mathbb{N}^*) = 1 \cdot \{0,1\}^*$ donc $\mathrm{bin} : \mathbb{N}^* \to 1 \cdot \{0,1\}^*$ est bijective.

4.4.3 Définition (Codage des entiers en binaire)

$$\begin{aligned} [\cdot] : \mathbb{N}^* &\to \{0,1\}^* \\ i &\mapsto w \qquad \text{si } \mathrm{bin}(i) = 1w \end{aligned}$$

4.4.4 Lemme

1. L'application $[\cdot]$ est bijective
   sa réciproque est définie par $[w]^{-1} = \mathrm{bin}^{-1}(1w)$.
2. De plus $[\cdot]$ respecte l'ordre lexicographique, c'est à dire :

$$\forall i, j \in \mathbb{N} \,.\, i \leq j \Leftrightarrow [i] \ll_1 [j]$$

Ce lemme dit que le n^{e} mot selon l'ordre lexicographique est $[n]$. Réciproquement, étant donné un mot $w \in \{0,1\}^*$, sa position dans l'ordre lexicographique est $[w]^{-1}$.

4.5 Codage des machines de Turing

Dans ce qui suit, on s'intéresse aux machines de Turing avec alphabet d'entrée $\{0,1\}$ qui comportent un unique état final, qui de sucroît est différent de l'état initial.

Soit $M = (Q, \{0,1\}, \Gamma, \delta, q_1, \mathrm{B}, F)$ une MT qui satisfait ces contraintes, on note alors :

- $F = \{ q_2 \}$,
- $Q = \{ q_1 \ldots, q_r \}$ avec $r \in \mathbb{N}$,
- $X_1 = 0$, $X_2 = 1$, et $X_3 = \mathrm{B}$,
- $\Gamma = \{ X_1 \ldots, X_s \}$ avec $s \in \mathbb{N}$.

On supposera en outre que δ mentionne au moins une fois, dans son domaine ou dans son image, chacun des éléments de Q et de Γ (cela signifie que les ensemble Q et Γ ne contiennent que des éléments « utiles »).

Ces restrictions ne sont en fait pas contraignantes. L'ensemble des machines qui les satisfont est noté **MT**.

4.5.1 Définition (Codage binaire d'une MT)
Posons $d_1 = -1$, $d_2 = 0$, $d_3 = 1$.
Soit $M = (Q, \{0,1\}, \Gamma, \delta, q_1, \mathrm{B}, F)$ une **MT**.
On code un élément $(q_i, X_j, q_k, X_l, d_m) \in \delta$ avec $i, j, k, l, m \in \mathbb{N}^*$, par le mot binaire suivant :

$$[\ (q_i, X_j, q_k, X_l, d_m)\ ] \triangleq 0^i\ \ 1\ \ 0^j\ \ 1\ \ 0^k\ \ 1\ \ 0^l\ \ 1\ \ 0^m$$

On ordonne les éléments de δ de la manière suivante :

$$(p, X, q, Y, d) \leq (p', X', q', Y', d') \Leftrightarrow p < p' \vee (p = p' \wedge X \leq X')$$

On code la fonction $\delta = \{z_1, \ldots, z_n\}$ (où $z_i < z_j$ si $i < j$) par :

$$[\delta] \triangleq z_1\ \ 11\ \ z_2\ \ \cdots\ \ \ 11\ \ z_n$$

Finalement le codage de M est donné par le codage de sa fonction de transition :

$$[M] \triangleq [\delta]$$

4.5.2 Lemme
La fonction $[\cdot] : \mathbf{MT} \to \{\,0,1\,\}^*$ est injective, mais pas surjective.

4.5.3 Lemme (Machine triviale)
Soit $M^{\mathrm{triv}} \triangleq (\{q_1, q_2\}, \{0,1\}, \{0,1,\mathrm{B}\}, \delta, q_1, \mathrm{B}, \{q_2\})$ avec δ définie par :

$$\delta(q_2, 0) \triangleq (q_1, 0, 0)$$
$$\delta(q_2, 1) \triangleq (q_1, 1, 0)$$
$$\delta(q_2, \mathrm{B}) \triangleq (q_1, \mathrm{B}, 0)$$

alors on a $\forall w \in \{0,1\}^*\ .\ (q_1, 1, w) \not\vdash_{M^{\mathrm{triv}}}$ et par conséquent $\mathrm{L}(M^{\mathrm{triv}}) = \emptyset$.

4.5.4 Définition Soit dec l'application définie par :

$$\mathrm{dec} : \{\,0,1\,\}^* \to \mathbf{MT}$$
$$w \mapsto \begin{cases} M & \text{si } w = [M] \\ M^{\mathrm{triv}} & \text{sinon} \end{cases}$$

On appelle n^{e} MT, la machine $M = \mathrm{dec}([n])$.

4.5.5 Notation Soit M une MT.

1. On écrit w_M pour $[M]$.
2. On écrit M_w pour $\mathrm{dec}(w)$.

4.6 Diagonalisation

4.6.1 Définition (Langage de diagonalisation)

$$L_{\mathrm{D}} \triangleq \{\, w \in \{\,0,1\,\}^* \mid w \notin \mathrm{L}(M_w) \,\}$$

4.6.2 Théorème

1. L_{D} n'est pas semi-décidable.
2. $\overline{L_{\mathrm{D}}}$ est semi-décidable.

4.7 Universalité et problème de l'arrêt

Voir [HMU03] §9.2.

4.7.1 Définition Soit $M \in$ **MT** et $w \in \{\,0,1\,\}^*$. La paire (M, w) est codée par :

$$[(M, w)] \triangleq [M]\ 111\ w$$

4.7.2 Définition (Langage universel)

$$L_{\mathrm{U}} \triangleq \big\{ [(M, w)] \in \{\,0,1\,\}^* \mid w \in \mathrm{L}(M) \big\}$$

4.7.3 Théorème

1. L_{U} est semi-décidable.
2. L_{U} n'est pas décidable.
3. $\overline{L_{\mathrm{U}}}$ n'est pas semi-décidable.

4.7.4 Définition (Problème de l'arrêt)

$$L_{\mathrm{H}} \triangleq \big\{ [(M, w)] \in \{\,0,1\,\}^* \mid M \text{ s'arrête pour } w \big\}$$

4.7.5 Théorème

1. L_{H} est semi-décidable.
2. L_{H} n'est pas décidable.
3. $\overline{L_{\mathrm{H}}}$ n'est pas semi-décidable.

$Id: notes-13.tex,v 1.8 2005/01/24 10:40:21 uwe Exp $

semaine 13

Voir [HMU03] §9.3, et [Koz97] §32–34.

4.8 Raisonnement par réduction

4.8.1 Définition (Réduction)
Soit Σ, Δ deux alphabets et $A \subseteq \Sigma^*$, $B \subseteq \Delta^*$.
Une *réduction* de A à B est une application $\sigma : \Sigma^* \to \Delta^*$ telle que :

1. σ est Turing-calculable par une machine totale.
2. $w \in A$ ssi $\sigma(w) \in B$

Nous écrivons $A \leq_{\mathrm{red}} B$ s'il existe une réduction de A à B.

4.8.2 Lemme
Soit $\Sigma_1, \Sigma_2, \Sigma_3$ trois alphabets et $A_1 \subseteq \Sigma_1^*$, $A_2 \subseteq \Sigma_2^*$, et $A_3 \subseteq \Sigma_3^*$.
Si $A_1 \leq_{\mathrm{red}} A_2$ et $A_2 \leq_{\mathrm{red}} A_3$, alors $A_1 \leq_{\mathrm{red}} A_3$.

4.8.3 Corollaire Soit Σ un alphabet, $(\Sigma^*, \leq_{\mathrm{red}})$ est un pré-ordre.

4.8.4 Théorème
Soit Σ, Δ deux alphabets et $A \subseteq \Sigma^*$, $B \subseteq \Delta^*$ tels que $A \leq_{\mathrm{red}} B$.

1. Si B est décidable, alors A est décidable.
2. Si B est semi-décidable, alors A est semi-décidable.

4.8.5 Corollaire (Principe de réduction)
Soit Σ et Δ deux alphabets, $A \subseteq \Sigma^*$, $B \subseteq \Delta^*$ tels que $A \leq_{\mathrm{red}} B$.

1. Si A est indécidable, alors B est indécidable.
2. Si A n'est pas semi-décidable, alors B n'est pas semi-décidable.

4.9 Propriétés des langages semi-décidables

4.9.1 Définition (Rappel) $\mathcal{L}_0$ est l'ensemble des langages semi-décidables. Sans perte de généralité on suppose que l'alphabet de ces langages est $\{0, 1\}$.

4.9.2 Définition (Propriété)
Une *propriété* P sur les éléments de X est un sous-ensemble de X.
On peut le décrire :

1. par un prédicat $P(x)$, on a alors $P = \{x \in X \mid P(x)\}$,
2. ou par une fonction caractéristique $P : X \to \{0, 1\}$.

4.9.3 Définition (Décidabilité des propriétés)
Soit $P \subseteq \mathcal{L}_0$ une propriété sur les langages semi-décidables.

1. P est *décidable* si sa fonction caractéristique

$$P : \mathcal{L}_0 \to \{0, 1\}$$
$$L \mapsto \begin{cases} 1 & \text{si } L \in P \\ 0 & \text{sinon} \end{cases}$$

   est totale et Turing-calculable.

2. P est *semi-décidable* si sa fonction semi-caractéristique

$$P : \mathcal{L}_0 \rightharpoonup \{0, 1\}$$
$$L \mapsto 1 \qquad \text{si } L \in P$$

   est Turing-calculable.

4.9.4 Définition
Soit $P \subseteq \mathcal{L}_0$ une propriété sur les langages semi-décidables.
P est *triviale* si :

1. $\forall L \in \mathcal{L}_0 \,.\, P(L) = 1$, ou si
2. $\forall L \in \mathcal{L}_0 \,.\, P(L) = 0$.

Sinon P est *non-triviale*.

4.9.5 Théorème (Rice I)
Toute propriété $P \subseteq \mathcal{L}_0$ *non-triviale* est indécidable.

4.9.6 Définition
Soit $P \subseteq \mathcal{L}_0$ une propriété sur les langages semi-décidables.
P est *monotone* si :

$$\forall L, L' \in \mathcal{L}_0 \,.\, L \subseteq L' \Rightarrow P(L) \leq P(L')$$

Sinon P est *non-monotone*.

4.9.7 Théorème (Rice II)
Toute propriété $P \subseteq \mathcal{L}_0$ *non-monotone* est non-semi-décidable.

$Id: notes-14.tex,v 1.5 2005/01/29 19:17:44 uwe Exp $

semaine 13

Voir [HMU03] §9.4.

4.10 Problèmes pratiques . . . et indécidables

4.10.1 Définition (Jeu de Domino) Soit Σ un alphabet.
Une *jeu de domino* (A, B) de taille $k \in \mathbb{N}$
est une paire d'applications $A, B : [1, k] \to \Sigma^*$.

4.10.2 Notation Un jeu de domino (A, B) de taille $k \in \mathbb{N}$ définit une liste finie de paires que nous notons :

$$\left[\frac{A(1)}{B(1)}\right], \left[\frac{A(2)}{B(2)}\right], \cdots, \left[\frac{A(k)}{B(k)}\right]$$

4.10.3 Définition Soit Σ un alphabet et $f : [1, k] \to \Sigma^*$.
Soit $I = i_1 \cdot i_2 \ldots \cdot i_m \in [1, k]^+$ un mot non vide. On définit :

$$f\langle I\rangle \triangleq f(i_1) \cdot f(i_2) \cdot \ldots \cdot f(i_m)$$

4.10.4 Définition (PCP) Soit Σ un alphabet.
Soit (A, B) un jeu de domino de taille $k \in \mathbb{N}$ et $I \in [1, k]^+$.

1. I est une *solution* de (A, B) si $A\langle I\rangle = B\langle I\rangle$.
2. I est une *solution partielle* de (A, B) si
   (a) $A\langle I\rangle$ est préfixe de $B\langle I\rangle$, ou
   (b) $B\langle I\rangle$ est préfixe de $A\langle I\rangle$.

Le *problème de correspondance de Post* (PCP) consiste à déterminer si un jeu de domino (A, B) donné a une solution ou non. Un jeu (A, B) est une *instance du PCP*.

4.10.5 Lemme Soit Σ un alphabet, (A, B) une instance de taille k du PCP et $I \in [1, k]^+$ une solution de (A, B).
Alors tout préfixe de I est une solution partielle de (A, B).

4.10.6 Définition (PCPM) Soit Σ un alphabet, (A, B) un jeu de domino de taille k et $I \in [1, k]^*$. I est *solution* de (A, B) selon le *PCP modifié* (PCPM) si :

$$A\langle 1 \cdot I\rangle = B\langle 1 \cdot I\rangle$$

(A, B) est une *instance du PCPM*.

4.10.7 Définition Toute instance (A, B) du PCP(M) peut être encodé en binaire, et en utilisant un symbole séparateur de la façon suivante :

- prendre la taille de l'alphabet ;
- déterminer le nombre de bits nécessaire ;
- commencer par coder ce nombre en binaire ;
- puis, en binaire, toutes les paires de l'instance du PCP ;

– séparer à chaque fois la suite avec le séparateur.

Les langages PCP et PCPM sont ainsi définis (de maniére injective) par des ensembles de mots sur un alphabet fini (ici, de taille 3).

$$\begin{array}{lcl} \text{PCP}^{\oplus} & \triangleq & \{\, w \in \text{PCP} \mid w \text{ est soluble} \,\} \\ \text{PCPM}^{\oplus} & \triangleq & \{\, w \in \text{PCPM} \mid w \text{ est soluble} \,\} \end{array}$$

4.10.8 Théorème $\text{PCPM}^{\oplus} \leq_{\text{red}} \text{PCP}^{\oplus}$.

4.10.9 Théorème $\text{PCP}^{\oplus}$ est semi-décidable.

4.10.10 Définition Soit $M = (Q, \{0,1\}, \Gamma, \delta, q_0, \text{B}, F)$ une MT.
Soit $c = (q, i, \text{rub})$ une configuration de M.
Si $\text{rub}(z) = \text{B}$ pour tout $z \in \mathbb{Z}$, alors $w_c \triangleq \epsilon$.
S'il existe $z \in \mathbb{Z}$ tel que $\text{rub}(z) \neq \text{B}$, alors soit z_d, z_f définis par :
– $\text{rub}(z_d) \neq \text{B} \neq \text{rub}(z_f)$, et
– $\forall j \in \mathbb{Z} \setminus [z_d, z_f]\,.\ \text{rub}(z) = \text{B}$.
Soit $w_c \triangleq \text{rub}(z_d) \cdot \ldots \cdot \text{rub}(z_f)$.
Nous représontons c aussi comme élément de $\Gamma^* Q \Gamma^+$ par :

$$\Theta(c) \triangleq \begin{cases} q \cdot \text{B} & \text{si } w_c = \epsilon \\ \text{rub}(z_d) \cdot \ldots \cdot \text{rub}(i{-}1) \cdot q \cdot \text{rub}(i) \cdot \ldots \cdot \text{rub}(z_f) & \text{si } i \in [z_d, z_f] \\ q \cdot \text{B}^{z_d - i} \cdot w_c & \text{si } i < z_d \\ w_c \cdot \text{B}^{i-1-z_f} \cdot q \cdot \text{B} & \text{si } z_f < i \end{cases}$$

Comme inverse, nous définissons pour le cas « standard » dans lequel la tête se trouve à l'intérieur du mot commençant en position 1 :

$$\Theta^{-1}(X_1 \cdot \ldots \cdot X_{i-1} \cdot q \cdot X_i \cdot \ldots \cdot X_n) \triangleq (q, i, \text{rub})$$

où $X_1 \neq \text{B} \neq X_n$ et tel que $\text{rub}(j) \triangleq \begin{cases} X_j & \text{si } j \in [1,n] \\ \text{B} & \text{sinon} \end{cases}$

4.10.11 Théorème $L_{\text{U}} \leq_{\text{red}} \text{PCPM}^{\oplus}$.

4.10.12 Corollaire $\text{PCPM}^{\oplus}$ et $\text{PCP}^{\oplus}$ sont indécidable.

4.10.13 Définition
Soit (A, B) une instance de taille k du PCP sur l'alphabet Σ.
Soit, pour tout $1 \leq j \leq k$: $w_j \triangleq A(j)$ et $x_j \triangleq B(j)$.
Soit $a_1, \ldots, a_k$ des nouveaux symboles. Alors,

$$G_{AB} \triangleq \big(\{\, S, S_A, S_B \,\}, \Sigma \cup \{\, a_1, \ldots, a_k \,\}, P, S\big)$$

est la grammaire (non-contextuelle) définie par :

$$\begin{array}{lcl} S & \to_G & S_A \mid S_B \\ S_A & \to_G & w_1 a_1 \mid \cdots \mid w_k a_k \mid w_1 S_A a_1 \mid \cdots \mid w_k S_A a_k \\ S_B & \to_G & x_1 a_1 \mid \cdots \mid x_k a_k \mid x_1 S_B a_1 \mid \cdots \mid x_k S_B a_k \end{array}$$

4.10.14 Théorème Soit (A, B) une instance du PCP.
(A, B) est soluble ssi G_{AB} est ambiguë.

4.10.15 Corollaire Le problème qui consiste à décider l'ambiguïté d'une grammaire quelconque est indécidable.