L2/L3 MASS/MIAGE **Année universitaire 2013 – 2014**
Université Paris 1

Aide mémoire pour le cours de mathématiques pour l'informatique

Fabrice Rossi

1 Automates et langages rationnels

1.1 Notations et définitions

- on note en général Σ l'alphabet considéré, un ensemble fini de symboles (souvent a,b,c,...) ;
- si Σ est un alphabet, on note Σ^* l'ensemble de tous les mots formés des symboles de Σ, en incluant notamment le mot vide ε ;
- le nombre de symboles d'un mot $w \in \Sigma^*$ est sa longueur, notée $|w|$;
- si L et M sont des langages (des sous-ensembles de Σ^*) :
  - $L.M = LM$ est l'ensemble des mots constitués d'un mot de L suivi d'un mot de M ;
  - $L|M = L \cup M$ est l'union des langages, c'est-à-dire l'ensemble des mots soit dans L soit dans M ;
- si L est un langage, on note :
  - $L^0 = \{\varepsilon\}$;
  - $L^1 = L$;
  - $L? = L|L^0$;
  - $L^{k+1} = L.L^k = L^k.L$;
  - $L^+ = \bigcup_{k\geq 1} L^k$;
  - $L^* = \bigcup_{k\geq 0} L^k$.
- pour tout mot $w \in \Sigma^*$, on note par simplification w le langage réduit au mot w seul, c'est-à-dire le langage $L = \{w\}$;
- un langage L sur Σ est rationnel (ou régulier) si et seulement si soit il est vide $L = \emptyset$, soit il vérifie une des conditions suivantes :
  1. $L = \{\varepsilon\}$;
  2. $L = \{a\}$ pour un symbole $a \in \Sigma$;
  3. $L = L_1|L_2$ pour deux langages rationnels L_1 et L_2 ;
  4. $L = L_1L_2$ pour deux langages rationnels L_1 et L_2 ;
  5. $L = L_1^*$ pour un langage rationnel L_1.

1.2 Résultats théoriques

1.2.1 Lemme de Arden

Soit α et β deux langages sur l'alphabet Σ. Si $\varepsilon \notin \alpha$, alors le seul langage X qui vérifie

$$X = \alpha X|\beta$$

est $X = \alpha^*\beta$.

1.2.2 Lemme de l'étoile

Soit L un langage rationnel sur Σ. Il existe un entier N ne dépendant que de L tel que tout mot $w \in L$ de longueur $|w| \geq N$ possède une décomposition sous la forme $w = xyz$ vérifiant les propriétés suivantes :

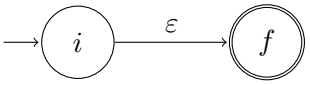
1. $0 < |y| \leq N$;
2. pour tout $n > 0$, $xy^n z \in L$.

1.3 Algorithme de Thompson

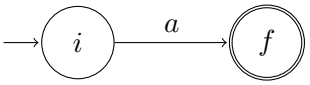
L'algorithme de Thompson construit à partir d'un langage rationnel non vide un automate (en général non déterministe) qui le reconnaît. La construction garantit que l'automate obtenu a un unique état terminal distinct de son état initial. Elle fonctionne récursivement à partir de la définition des langages rationnels rappelée ci-dessus. Dans les schémas qui suivent, les flèches ondulées résument un automate complet reconnaissant le langage indiqué au dessus de la flèche. L'état i_k était l'état initial de l'automate reconnaissant L_k et l'état f_k était son état final. Pour le langage L, l'état initial est noté i et l'état final f.

Soit L rationnel non vide, alors une des conditions suivantes est vraie :

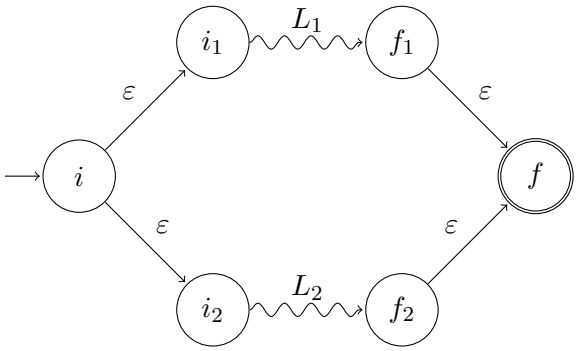
1. si $L = \{\varepsilon\}$, l'algorithme construit l'automate :



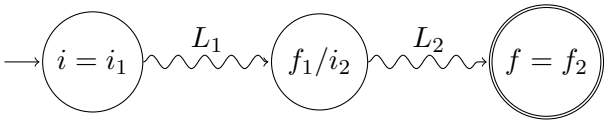
2. si $L = \{a\}$, l'algorithme construit l'automate :



3. si $L = L_1|L_2$, l'algorithme construit récursivement un automate pour L_1 et un automate pour L_2, puis les combine en :

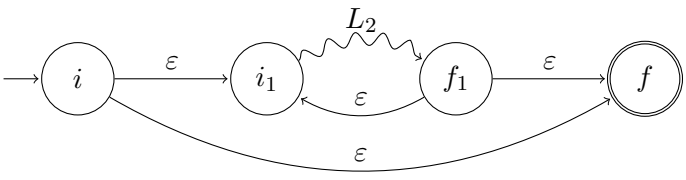


4. si $L = L_1 L_2$, l'algorithme construit récursivement un automate pour L_1 et un automate pour L_2, puis les combine en :



L'état f_1/i_2 est la fusion de l'état final de l'automate reconnaissant L_1 avec l'état initial de l'automate reconnaissant L_2.

5. si $L = L_1^*$, l'algorithme construit récursivement un automate pour L_1 puis construit l'automate suivant :



2 Calcul propositionnel

2.1 Formules

Les formules de la logique propositionnelle sont construites à partir :
- d'un ensemble fini de variables propositionnelles, V (en général des lettres minuscules, par exemple $V = \{a, b, c\}$) ;
- d'un opérateur unaire de négation $\neg$;
- de quatre opérateurs binaires :
  1. ou : $\vee$;
  2. et : $\wedge$;
  3. implique : $\Rightarrow$;
  4. équivalent : $\Leftrightarrow$.

On note $\mathcal{F}(V)$ l'ensemble des formules construites sur les variables propositionnelles V.

2.2 Notations

- on utilise souvent deux constantes : $\top$ dont la valuation est toujours 1 et $\bot$ dont la valuation est toujours 0 ;
- on note $F \equiv G$ si et seulement si pour toute valuation v, $v(F) = v(G)$ (les formules F et G sont sémantiquement équivalentes).

2.3 Équivalences notables

Si cela est indiqué dans l'énoncé, on peut utiliser les équivalences notables suivantes pour simplifier ou transformer des formules :

$$(X \Rightarrow Y) \equiv ((\neg X) \vee Y) \qquad (X \Leftrightarrow Y) \equiv ((X \Rightarrow Y) \wedge (Y \Rightarrow X))$$
$$((X \wedge Y) \wedge Z) \equiv (X \wedge (Y \wedge Z)) \qquad ((X \vee Y) \vee Z) \equiv (X \vee (Y \vee Z))$$
$$(X \wedge Y) \equiv (Y \wedge X) \qquad (X \vee Y) \equiv (Y \vee X)$$
$$(X \wedge (Y \vee Z)) \equiv (X \wedge Y) \vee (X \wedge Z)) \qquad (X \vee (Y \wedge Z)) \equiv (X \vee Y) \wedge (X \vee Z))$$
$$(\neg(\neg X)) \equiv X$$
$$(\neg(X \wedge Y)) \equiv ((\neg X) \vee (\neg Y) \qquad (\neg(X \vee Y)) \equiv ((\neg X) \wedge (\neg Y)$$

2.4 Règles de Quine

Dans l'algorithme de Quine, on utilise les équivalences notables suivantes :

$$(\neg\bot) \equiv \top \qquad (\neg\top) \equiv \bot$$
$$(\top \Rightarrow X) \equiv X \qquad (X \Rightarrow \top) \equiv \top$$
$$(\bot \Rightarrow X) \equiv \top \qquad (X \Rightarrow \bot) \equiv (\neg X)$$
$$(X \vee \top) \equiv \top \qquad (X \vee \bot) \equiv X$$
$$(\top \vee X) \equiv \top \qquad (\bot \vee X) \equiv X$$
$$(X \wedge \top) \equiv X \qquad (X \wedge \bot) \equiv \bot$$
$$(\top \wedge X) \equiv X \qquad (\bot \wedge X) \equiv \bot$$
$$(X \Leftrightarrow \top) \equiv X \qquad (X \Leftrightarrow \bot) \equiv (\neg X)$$
$$(\top \Leftrightarrow X) \equiv X \qquad (\bot \Leftrightarrow X) \equiv (\neg X)$$

2.5 Calcul des séquents

Le calcul des séquents repose (exclusivement) sur huit règles (sauf mention contraire, on n'utilise pas les équivalences notables) :

$$\frac{\Gamma,A,B,\Delta \vdash \Theta}{\Gamma,(A \wedge B),\Delta \vdash \Theta} \wedge_g \qquad \frac{\Theta \vdash \Gamma,A,\Delta \qquad \Theta \vdash \Gamma,B,\Delta}{\Theta \vdash \Gamma,(A \wedge B),\Delta} \wedge_d$$

$$\frac{\Gamma,A,\Delta \vdash \Theta \qquad \Gamma,B,\Delta \vdash \Theta}{\Gamma,(A \vee B),\Delta \vdash \Theta} \vee_g \qquad \frac{\Theta \vdash \Gamma,A,B,\Delta}{\Theta \vdash \Gamma,(A \vee B),\Delta} \vee_d$$

$$\frac{\Gamma,\Delta \vdash A,\Theta \qquad \Gamma,B,\Delta \vdash \Theta}{\Gamma,(A \Rightarrow B),\Delta \vdash \Theta} \Rightarrow_g \qquad \frac{A,\Gamma \vdash \Delta, B, \Theta}{\Gamma \vdash \Delta,(A \Rightarrow B),\Theta} \Rightarrow_d$$

$$\frac{\Gamma,\Delta \vdash A,\Theta}{\Gamma,(\neg A),\Delta \vdash \Theta} \neg_g \qquad \frac{A,\Gamma \vdash \Delta,\Theta}{\Gamma \vdash \Delta,(\neg A),\Theta} \neg_d$$

On admet comme axiomes les séquents de la forme suivante :

$$\Gamma,A \vdash \Theta,A$$

où A désigne une formule (on ne limite pas A à être une variable propositionnelle).

3 Logique de Hoare

3.1 Interprétation

Sauf mention contraire explicite, on suppose en général que l'interprétation des symboles de fonctions, des symboles de constantes et des symboles de prédicats est celle de l'arithmétique dans $\mathbb{Z}$. En particulier, le symbole / désigne la division euclidienne, c'est-à-dire que 5/2 est le quotient de la division de 5 par 2 (soit 2). Le symbole % correspond au calcul du reste de la division (par exemple $5\%2 = 1$).

3.2 Notations

On rappelle que dans les triplets de Hoare, les accolades {} sont utilisées pour la correction partielle (on ne considère que les cas où le programme s'arrête) alors que les crochets $\langle\rangle$ sont utilisés pour la correction totale (le programme doit toujours s'arrêter).

Si F est un terme (resp. une formule) du calcul des prédicats, $F[x \leftarrow T]$ est le terme obtenu (resp. la formule obtenue) en remplaçant toutes les occurrences libres de x par T. Si on autorise les variables booléennes, T peut être une formule, sinon T est un terme.

3.2.1 Logique de Hoare

La logique de Hoare repose sur des règles de déduction et des axiomes. Les axiomes sont ceux de l'affectation (Hoare et Floyd) :

$$\frac{}{\langle F[x \leftarrow T]\rangle \ \texttt{x} \ \leftarrow \texttt{T} \ \langle F\rangle} \leftarrow \text{Hoare}$$

Dans l'axiome ci-dessus, il faut que x soit une variable libre dans F.

$$\frac{}{\langle F\rangle \ \texttt{x} \ \leftarrow \texttt{T} \ \langle(\exists y, (F[x \leftarrow y] \wedge (x = T[x \leftarrow y])))\rangle} \leftarrow \text{Floyd}$$

Le reste du système consiste en les règles de déduction suivantes :

$$\frac{\langle F\rangle \ \texttt{prog1} \ \langle G\rangle \qquad \langle G\rangle \ \texttt{prog2} \ \langle H\rangle}{\langle F\rangle \ \texttt{prog1; prog2} \ \langle H\rangle} \ ;$$

$$\frac{(F \Rightarrow G) \qquad \langle G \rangle \ \texttt{prog} \ \langle H \rangle \qquad (H \Rightarrow I)}{\langle F \rangle \ \texttt{prog} \ \langle I \rangle} \Rightarrow$$

$$\frac{\langle (F \wedge b) \rangle \ \texttt{prog1} \ \langle G \rangle \qquad \langle (F \wedge (\neg b)) \rangle \ \texttt{prog2} \ \langle G \rangle}{\langle F \rangle \ \texttt{if (}b\texttt{) prog1 else prog2 end if} \ \langle G \rangle} \text{ if then}$$

$$\frac{\{(I \wedge b)\} \ \texttt{prog} \ \{I\}}{\{I\} \ \texttt{while(}b\texttt{) prog end while} \ \{(I \wedge (\neg b))\}} \text{ while partiel}$$

Dans la règle précédente, I est appelé *invariant* de la boucle.

$$\frac{\langle (I \wedge b \wedge (V = n)) \rangle \ \texttt{prog} \ \langle I \wedge (V < n) \wedge (V \geq 0)) \rangle}{\langle I \rangle \ \texttt{while(}b\texttt{) prog end while} \ \langle (I \wedge (\neg b)) \rangle} \text{ while total}$$

Dans cette dernière règle, V est un terme et n une variable libre qui n'est utilisée nulle part ailleurs. Comme dans la règle précédente, I est un invariant, alors que V est un *variant* de la boucle.