Handout Cours 2

Suite des Expressions Rationnelles, Automates Déterministes

1 Expressions Rationnelles : Notations (Suite)

Quand on écrit des expressions rationnelles on se permet d'omettre des parenthèses qui ne sont pas utiles, sachant que les deux opérateurs binaires $+$ et $\cdot$ sont associatifs, et que * lie plus fortement que $\cdot$, qui lie plus fortement que $+$.

Quelques raccourcis ("sucre syntaxique") fréquents :

- $r?$ est une abréviation pour $r + \epsilon$
- r^+ est une abréviation pour rr^*

Un langage L est *rationnel* quand il existe une expression rationnelle r telle que $L = \mathcal{L}(r)$.

On se permet des séquences de définitions d'expressions rationnelles, c-à-d on donne chaque fois un nom à une expression rationnelle, et quand on écrit une autre expression rationnelle dans la suite on peut se servir d'un nom qu'on a déjà défini. Par exemple

$$\begin{aligned} r_1 &= a(a+b)a \\ r_2 &= br_1^*b \end{aligned}$$

Attention on a dans ce type de définitions pas droit à des cycles.

2 Questions faciles et difficiles

2.1 Propriétés de Clôture de la classe des langages rationnels

On a d'abord évidemment :

- La classe des langages rationnels est close sous union, c-à-d quand L_1 et L_2 sont rationnels, alors $L_1 \cup L_2$ est également rationnel.
- La classe des langages rationnels est close sous concaténation, c-à-d quand L_1 et L_2 sont rationnels, alors $L_1 \cdot L_2$ est également rationnel.
- La classe des langages rationnels est close sous l'étoile de Kleene, c-à-d quand L est rationnel, alors L^* l'est également.

Mais on peut imaginer d'autres propriétés de clôture qui sont beaucoup moins évidentes :

- Est-ce que la classe des langages rationnels est close sous intersection ?
- Est-ce que la classe des langages rationnels est close sous complément ?

Nous allons trouver des réponses à ces questions pendant ce semestre.

2.2 Décider des langages rationnels

C'est le problème : étant donné un mot w et une expression rationnelle r, est-ce que $w \in \mathcal{L}(r)$? On peut imaginer un algorithme naïf qui essaye toutes les factorisations possibles de w et puis de faire correspondre les facteurs aux parties de r, mais ça serait énormément inefficace. Comment faire mieux ?

2.3 Des langages qui ne sont pas rationnels ?

Étant donné un alphabet Σ, l'ensemble RAT des expressions rationnelles sur Σ est dénombrable infini, tandis que la classe de tous les langages sur Σ est non-dénombrable. Il y a donc beaucoup plus de langages que de langages rationnels.

Mais à quoi ressemble un langage non-rationnel ? En général, comment peut-on montrer qu'un langage donné n'est *pas* rationnel ?

3 Automates Finis Déterministes

Un *automate fini déterministe* (AFD) est un quintuplet $(\Sigma, Q, q_0, F, \delta)$ tel que

- Σ est un alphabet fini ;
- Q est un ensemble fini, appelé l'ensemble des *états* ;
- $q_0 \in Q$, appelé *l'état initial* ;
- $F \subseteq Q$, appelé *ensemble des états acceptants* ;
- $\delta : Q \times \Sigma \rightsquigarrow Q$ est une fonction partielle, appelée la *fonction de transition*.

Certains auteurs disent *état final* à la place de *état acceptant*.

Un AFD est *complet* quand sa fonction de transition est une fonction totale.

Étant donné un automate $(\Sigma, Q, q_0, F, \delta)$, on définit une fonction partielle $\delta^* : Q \times \Sigma^* \rightsquigarrow Q$ par récurrence sur le deuxième argument :

$$\begin{array}{rcl} \delta^*(q, \epsilon) & = & q \\ \delta^*(q, aw) & = & \left\{ \begin{array}{ll} \text{indéfini} & \text{quand } \delta(q,a) \text{ indéfini} \\ \delta^*(\delta(q,a), w) & \text{quand } \delta(q,a) \in Q \end{array} \right. \end{array}$$

Définition équivalente : Quand $|w| = n$ et il existe $p_0, \ldots, p_n$ tel que pour tous $0 \leq i < n$: $\delta(p_i, w[i]) = p_{i+1}$, alors $\delta^*(p_0, w) = p_n$.

Un automate $A = (\Sigma, Q, q_0, F, \delta)$ *accepte* un mot $w \in \Sigma^*$ quand $\delta^*(q_0, w) \in F$.

Le *langage reconnu* par l'automate $A = (\Sigma, Q, q_0, F, \delta)$ est $\mathcal{L}(A) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$.

Un langage L est *reconnaissable* quand il existe un AFD A tel que $L = \mathcal{L}(A)$. Les anglophones disent : *regular langage*. On note *Rec* la classe des langages reconnaissables.

Attention : il y a a priori trois comportements différents possibles d'un automate $(\Sigma, Q, q_0, F, \delta)$ sur un mot $w \in \Sigma^*$:

- il consomme le mot entier et arrive dans un état acceptant : $\delta^*(q_0, w) \in F$;
- il consomme le mot entier et arrive dans un état qui n'est pas acceptant : $\delta^*(q_0, w) \in Q \backslash F$ i;
- il ne consomme pas le mot entier car il bloque sur une transition manquante : $\delta^*(q_0, w)$ pas défini.

Dans le premier cas l'automate accepte $w \in \Sigma^*$, dans les deux derniers cas il le *refuse*. Quand un automate est complet, le troisième cas ne peut jamais se produire.

Il est facile de montrer, on utilisant un état *puits*, que pour

tout AFD A il y a un AFD complet A' tel que $\mathcal{L}(A) = \mathcal{L}(A')$.

Tout langage fini est reconnaissable (construction d'un arbre préfixe).

Il n'est pour l'instant pas évident si la classe des langages reconnaissable est close sous union, concaténation, ou étoile de Kleene. En revanche, cette classe est close sous complément !

Aussi, on a une méthode évidente pour montrer qu'un langage est reconnaissable (construire l'automate), mais pour l'instant aucune méthode pour montrer qu'un langage n'est *pas* reconnaissable.