Handout 7

L'Algorithme de Brzozowski et McCluskey, et le lemme d'Arden

1 L'algorithme de Brzozowski et McCluskey

Il s'agit d'un algorithme pour transformer un automate en expression rationnelle.

1.1 Les automates généralisés

Pendant les transformations on manipule un *automate généralisé*, c'est un automate qui a des expressions rationnelles sur les flèches. Pendant l'exécution, un automate généralisé peut aller d'un état q vers un état p en consommant un *mot* v quand ce mot v est dans le langage de l'expression rationnelle sur une flèche de q vers p.

Formellement on représente les transitions d'un tel automate généralisé par une fonction qui à chaque état q et chaque état p associe une expression rationnelle $E_{q,p}$. Quand une transition est étiquetée par $\emptyset$ on ne la représente pas dans le dessin. Un tel automate généralisé *accepte* un mot $w \in \Sigma^*$ quand il existe des états $q_0, \ldots, q_n$ et une *factorisation* du mot

$$w = w_0 \cdots w_{n-1}$$

où $w_i \in \Sigma^*$, tel que

- q_0 est un état initial ;
- q_n est un état acceptant ;
- pour tout i avec $0 \leq i < n$: $w_i \in \mathcal{L}(E_{q_i,q_{i+1}})$

1.2 L'algorithme de Brzozowski et McCluskey

Cet algorithme transforme un automate (qui peut être non-déterministe avec des ϵ-transitions) en expression rationnelle, par élimination des états. L'invariant maintenu par les transformation est le même que celui de l'algorithme de Thompson:

- il y a un seul état initial ;
- aucune transition entre dans l'état initial ;
- il y a un seul état acceptant ;
- aucune transition sort de l'état acceptant ;
- l'état initial n'est pas l'état acceptant.

Il faut d'abord assurer que l'automate de départ satisfait toutes ces propriétés, si ce n'est pas le cas ajouter des nouveaux uniques états initial et acceptant avec les ϵ-transitions qu'il faut.

Puis on élimine, un après l'autre, tous les états de cet automate *sauf* l'état initial et *sauf* l'état acceptant. Pour éliminer un état p, on remplace pour tous états q, r l'expression $E_{q,r}$ par $E_{q,r} + E_{q,p}E_{p,p}^* E_{p,r}$. En pratique on a quelque chose à faire seulement quand il y a une flèche de q vers p et une de p vers r, car dans le cas contraire on a que $E_{p,q} = \emptyset$ or $E_{q,r} = \emptyset$. Aussi, il convient de simplifier les expressions rationnelles quand c'est possible.

À la fin on obtient un automate généralisé avec seulement 2 états (l'état initial et l'état acceptant), l'expression sur la flèche de l'état initial vers l'état acceptant est l'expression cherchée.

2 Le Lemme d'Arden et application aux automates

2.1 Le lemme d'Arden

1. La solution la plus petite de l'équation

$$L = A \cdot L \cup B$$

où A et B sont des langages et L une variable, est A^*B.

2. Si $\epsilon \notin A$ alors A^*B est même la seule solution de cette équation.

Quand tous les langages sont rationnels on se permet la syntaxe des expressions rationnelles ($+$ à la place de $\cup$ etc.).

2.2 Application

Le lemme d'Arden est utile pour transformer un automate en expression rationnelle. Étant donné un AFD (S, Q, q_0, F, δ), on écrit un système d'équations :

$$L_q = \sum_{a \in S} a\, L_{\delta(q,a)} \qquad \text{si } q \in Q - F$$

$$L_q = \sum_{a \in S} a\, L_{\delta(q,a)} + \epsilon \qquad \text{si } q \in F$$

Ce système décrit les langages $L_q = \{w \in \Sigma^* \mid \delta^*(q, w) \in F\}$. Pour trouver la solution la plus petite on transforme ce système en un système équivalent où l'équation pour L_{q_0} n'a plus de variables sur son côté droit. On se sert de

- remplacer la variable L partout par e quand on a une équation $L = e$ où L ne paraît pas dans e ;
- remplacer la variable L par sa solution selon Arden quand on a une équation $L = e$ où L paraît dans e ;
- Simplifier les expressions rationnelles tant qu'on peut.

3 Conséquence des traductions entre expressions rationnelles et automates

Nous avons vu

- qu'on peut traduire toute expression rationnelle r en un automate A_r tel que $\mathcal{L}(r) = \mathcal{L}(A_r)$, à l'aide de l'algorithme de Thompson (handout 5) ou celui de Glushkov (handout 6).

  Par conséquent, $Rat \subseteq Rec$.

- qu'on peut traduire tout automate A en une expression rationnelle r_A telle que $\mathcal{L}(A) = \mathcal{L}(r_A)$, à l'aide de l'algorithme de Brzozowki-McCluskey, ou la méthode d'Arden (ce handout).

  Par conséquent, $Rec \subseteq Rat$.

Donc : $Rat = Rec$ - tout langage est rationnel si et seulement s'il est reconnaissable.

Ça veut aussi dire que la classe *Rat* "hérite" les propriétés de clôture de *Rec*, et *Rec* hérite les propriétés de clôture de *Rat*. En résumé :

La classe des langages rationnelles, qui est identique à la classe des langages reconnaissables, et close sous :

- union,
- complément,
- intersection,
- concaténation,
- étoile de Kleene,
- miroir,

et encore des autres opérations (voir le TD).