TP4 : Minimisation

1 Objectif

L'objectif de ce TP est de programmer l'algorithme de minimisation d'un automate déterministe et d'ajouter à notre interpréteur la commande correspondante.

2 Partition

L'algorithme de minimisation manipule des partitions de l'ensemble des états de l'automate à minimiser.

On représentra une partition d'un ensemble comportant les entiers $\{0, \ldots, n-1\}$ sous la forme d'un tableau d'entiers de taille n. Le principe consiste à mettre dans la case i le numéro de la partie à laquelle i appartient. La partition $\{\{0, 2, 5\}, \{1, 4\}, \{3\}\}$, par exemple, sera représentée de la façon suivante :

0	1	2	3	4	5
0	1	0	2	1	0

Dans l'exemple, nous avons arbitrairement associé l'identifiant 0 à la première partie, l'identifiant 1 à la seconde et ainsi de suite.

3 Algorithme

L'algorithme de minimisation d'un automate déterministe $D = \langle Q, \Sigma, \delta, i, F \rangle$, consiste à construire des partitions successives de l'ensemble Q.

On commence par une partition binaire dans laquelle les états d'acceptation sont regoupés au sein d'une partie et les autres états au sein de l'autre. Puis, les différentes transitions sortantes des états de chaque partie sont examinées. Les états d'une partie P dont les transitions sortantes sur un symbole s ont pour destination des états appartenant à une même partie sont alors regroupés au sein d'un sous-ensemble de P.

La décomposition d'une partie en parties plus petites peut être effectuée de la manière suivante. Reprenons la partition décrite ci-dessus : $\{\{0, 2, 5\}, \{1, 4\}, \{3\}\}$,

que nous appellerons partition précédente et supposons que : $\delta(0, s) = 5$, $\delta(1, s) = 3$, $\delta(2, s) = 0$, $\delta(3, s) = 1$, $\delta(4, s) = 2$ et $\delta(5, s) = 1$. On notera $P(i)$ la partie de l'état i dans la partition précédente.

On construit une nouvelle partition, appelée partition courante de la manière suivante.

On commence par initialiser la partition courante en affectant à chaque élément la partie fictive `-1`

0	1	2	3	4	5
−1	−1	−1	−1	−1	−1

Puis, on associe au plus petit élément de chaque partie, dans la partition précédente, l'indentifiant de sa partie. On obtient :

0	1	2	3	4	5
0	1	−1	2	−1	−1

On parcours alors les états dans l'ordre croissant. Pour tout état i, les états $j < i$ sont inspectés. Si i et j appartiennent à la même partie dans la partition précédente ($P(i) = P(j)$) et si s permet d'aller à partir de j et à partir de i vers des états appartenant à la même partie k dans la partition précédente ($P(\delta(j, s)) = P(\delta(i, s)) = k$), alors on affecte i à la même partie que j. C'est ce qui se passe pour l'état 2 de notre exemple, qui se retouve affecté à la même partie que l'état 0.

0	1	2	3	4	5
0	1	0	2	−1	−1

Si on ne trouve pas d'état j tel que $P(i) = P(j) \wedge P(\delta(j, s)) = P(\delta(i, s))$ alors on affecte i à une nouvelle partition. Ce qui est le cas des états 4 et 5 dans notre exemple. On obtient en fin de compte la partition suivante :

0	1	2	3	4	5
0	1	0	2	3	4

Le processus est alors réitéré sur la partition courante, qui devient partition précédente. Il s'arrête lorsque l'on aboutit à une partition stable pour tous les symboles de l'alphabet. Cette partition est appelée partition finale.

Il ne reste alors plus qu'à construire l'automate minimal M. Pour cela on crée un automate comportant autant d'états qu'il y a de parties dans la partition finale. Puis, pour toute transition (o, s, d) de D, on ajoute à M la transition $(P_f(o), s, P_f(d))$ où $P_f(i)$ est la partie à laquelle appartient l'état i dans la partition finale.

L'état initial de M est l'état qui correspond à la partie de la partition finale qui contient l'état initial de D. Tout état de M qui correspond à une partie contenant un état d'acceptation de D est état d'acceptation de M.