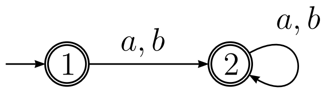
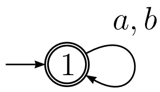
Université de Rouen	Année en cours
Faculté des sciences	Théorie des langages
Licence Informatique 3ème année	Résumé de cours pour le TD n° 6

Minimisation d'un automate

Nous avons vu dans les chapitres précédents, que pour un langage reconnaissable donné, on pouvait construire plusieurs automates le reconnaissant et plus particulièrement plusieurs automates complets.

Exemple :





Les automates complets ci-dessus reconnaissent tous deux le langage Σ^* pour $\Sigma = \{a, b\}$

On se pose alors le problème suivant : « étant donné un langage reconnaissable, peut-on construire un automate complet le reconnaissant et ayant un nombre d'états minimum ? »

On se place dans la classe des automates complets car on a vu précédemment que c'était la classe la plus intéressante. On se restreindra même aux automates complets accessibles. Un automate est accessible si tous ses états sont accessibles par un chemin quelconque à partir de l'état initial.

Soit $M = (\Sigma, Q, \delta, \{i\}, F)$ un automate complet accessible qui reconnait le langage L. Si cet automate n'est pas minimal en le nombre d'états (on dira plus simplement minimal), c'est qu'il a « trop » d'états, c'est à dire que plusieurs états jouent des rôles identiques dans l'automate.

On cherche alors à identifier les états ayant même rôle dans l'automate. On définit sur Q une relation d'équivalence notée E par :

$$\forall p, q \in Q, pEq \Leftrightarrow (\forall w \in \Sigma^*, \delta(p, w) \in F \Leftrightarrow \delta(q, w) \in F)$$

Autrement dit, p et q sont équivalents si partant de p ou de q, on atteint un état terminal en lisant les mêmes mots. Dans le cas contraire, on dit que p et q sont séparables.

On cherche alors à construire sur $Q \times Q$ la table de cette relation. On pourra, à l'aide de cette table, construire un automate équivalent minimal en identifiant tous les états équivalents.

Remarques : p et q sont séparables si il existe au moins un mot w de Σ^* qui les sépare :

$$\exists w \in \Sigma^* \text{ tel que } (\delta(p, w), \delta(q, w)) \in F \times (Q \setminus F) \cup (Q \setminus F) \times T$$

En particulier, si $p \in F$ et $q \in Q \backslash F$ ou $p \in Q \backslash F$ et $q \in F$ alors p et q sont séparés par ε.

E est réflexive : $\forall p \in Q,\ pEp$ E est symétrique : $\forall p, q \in Q,\ pEq \Rightarrow qEp$ La table est donc une matrice symétrique.

On peut donc se contenter de considérer les couples $(p, q) \in F \times F \cup (Q \setminus F) \times (Q \setminus F)$ tels que $p < q$.

Principe de l'algorithme de minimisation :
On marque tous les couples qui sont séparables. A la fin de l'algorithme, les couples non marqués sont les couples équivalents. On associe à chaque couple (p, q) une liste des couples (p', q') des états pour lesquels il existe un chemin menant en (p, q) : $\exists w \in \Sigma^*$ tel que $\delta(p', w) = p$ et $\delta(q', w) = q$. Cette liste ne sera construite par l'algorithme que tant que le couple (p, q) ne sera pas marqué.

Algorithme minimiser
Initialisation
marquer tous les $(p, q) \in F \times (Q \setminus F) \cup (Q \setminus F) \times F$

Itération
pour chaque couple $(p, q) \in F \times F \cup (Q \setminus F) \times (Q \setminus F)$ tel que $p < q$ **faire**
    pour chaque $a \in \Sigma$ **faire**
        calculer $p' = \delta(p, a)$ et $q' = \delta(q, a)$
    fpour
si il existe $a \in \Sigma$ tel que (p', q') est un couple marqué **alors**
        (* $\exists w$ qui sépare p' et q', p et q sont donc séparés par aw *)
        marquer (p, q) et tous les couples de la liste associée récursivement
    sinon
        (* si p' et q' sont séparés par la suite, il faudra également séparer p et q *)
        ajouter (p, q) à la liste associée à (p', q')
    fsi
fpour