Langages et Automates : LA3
Partie 3 : Clotures - De l'Expression Rationnelle à l'Automate

Propriétés de Cloture

Dans les transparents suivants on va prouver les propriétés de cloture suivantes des langages reconnaissables.

Théoreme

Soient L et L' deux langages reconnaissables sur l'alphabet Σ. Alors :

1. $L' \cup L'$
2. $\overline{L'}$
3. $L \cap L'$
4. $L.L'$
5. L^*

sont tous des langages reconnaissables

En particulier grace à 1,4 et 5 on obtient le corollaire suivant :

Corollaire

Tout langage rationnel est reconnaissable

Opérations ensemblistes

- Pour l'union on peut comme on l'a vu précédemment utiliser la déterminisation :
  Si $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ et $\mathcal{A}' = (\Sigma, Q', I', F', \delta')$ sont deux AFND, Alors $\mathcal{A}'' = (\Sigma, Q \cup Q', I \cup I', F \cup F', \delta \cup \delta')$ est un AFND reconnaissant l'union des langages reconnus par $\mathcal{A}$ et $\mathcal{A}'$.
- Pour le complémentaire on peut faire la construction suivante :
  Si $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ est complet alors $\mathcal{A}' = (\Sigma, Q, q_0, Q \setminus F, \delta)$ est un automate reconnaissant le complémentaire du lagage reconnu par $\mathcal{A}$.
- L'intersection découle des deux précédentes puisque $L \cap L' = \overline{\overline{L} \cup \overline{L'}}$.

Automate Produit

Pour l'union et l'intersection, on peut aussi utiliser la construction dite de l'automate produit

Théoreme

$\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ est un AFD complet reconnaissant L.
$\mathcal{A}' = (\Sigma, Q', q'_0, F', \delta')$ un AFD complet reconnaissant L'.
On définit un AFND $\mathcal{A}''$:

- *Etats : $Q \times Q'$*
- *Etat initial : (q_0, q'_0)*
- $\delta''((q, q'), a) = (\delta(q, a), \delta'(q', a))$

Si les états finaux de $\mathcal{A}''$ sont $F \times F'$, alors $\mathcal{A}''$ reconnait $L \cap L'$.
Si les états finaux de $\mathcal{A}''$ sont $F \times Q' \cup Q \times F'$, alors $\mathcal{A}''$ reconnait $L \cup L'$.

Cet automate simule la lecture "simultanée" dans les deux automates en même temps.
L'hypothèse de complétude n'est pas nécessaire pour l'intersection.
Il n'est pas difficile de voir qu'en fait on obtient exactement le meme automate qu'avec la méthode qui consiste a déterminiser l'union disjointe des deux automates.

Cloture par Concaténation

Théoreme

$\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ est un AFD reconnaissant L.
$\mathcal{A}' = (\Sigma, Q', q'_0, F', \delta')$ un AFD reconnaissant L'.
On suppose Q et Q' disjoints et on définit un AFND $\mathcal{A}''$:

- *Etats : $Q \cup Q'$*
- *Etat initial : q_0*
- *Etats finaux :* $\begin{cases} F' & \text{si } q'_0 \notin F', \text{ (c'est à dire si } \varepsilon \notin L') \\ F' \cup F & \text{sinon} \end{cases}$
- $\forall a \in \Sigma,\ \delta''(q, a) = \begin{cases} \{\delta(q, a)\} & \text{si } q \in (Q \setminus F) \cup Q' \\ \{\delta(q, a)\} \cup \{\delta'(q'_0, a)\} & \text{si } q \in F \end{cases}$

Alors $\mathcal{A}''$ est un AFND reconnaissant $L.L'$

Cloture par étoile

Avant de montrer la construction pour l'étoile, on introduit la définition suivante : un automate est dit standard, si il n'existe aucune transition entrante dans son état initial.

Théoreme

Pour tout langage reconnaissable L, il existe un AFD standard qui le reconnait.

En partant d'un AFD $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ quelconque reconnaissant L, Il suffit d'ajouter un état q'_0 et de modifier δ par δ' avec :

- $\forall q, a$, si $\delta(q, a) = q_0$ alors $\delta'(q, a) = q'_0$
- $\forall a,\ \delta'(q'_0, a) = \delta(q_0, a)$
- toutes les autres transitions sont inchangées.

On prend alors cet état q'_0 comme nouvel état initial (q_0 n'est plus initial). Si q_0 était état final, alors q'_0 le devient.

Cloture par Etoile

Théoreme

$\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ est un AFD standard reconnaissant L. On définit un AFND $\mathcal{A}'$:

- *Etats : Q*
- *Etat initial : q_0*
- *Etats finaux : $Q \cup q_0$*
- $\delta''(q, a) = \begin{cases} \{\delta(q, a)\} & \text{si } q \in Q \setminus F \\ \{\delta(q, a)\} \cup \{\delta(q_0, a)\} & \text{si } q \in F \end{cases}$

Alors $\mathcal{A}''$ est un AFND reconnaissant L^*

(Pourquoi a-t-on besoin d'un AFD standard ?)

Automates à Epsilon Transitions

On va présenter ici un l'algorithme de Thompson pour passer de l'E.R. à l'automate. C'est celui qui est implémenté dans la commande *grep* de UNIX.

Cet algorithme utilise une notion encore plus générale d'automate : les automates à ε-transitions : aux transitions usuelles s'ajoutent des transitions dont l'étiquette n'est pas une lettre mais le mot vide ε :

$$q_1 \xrightarrow{\varepsilon} q_2$$

Lors du calcul d'un mot, si on est dans l'état q_1, on peut "sauter" directement dans l'état q_2. Comme dans le cas non deterministe, un mot est reconnu par un automate avec ε-transitions si il existe au moins un calcul qui ne bloque pas et finit dans un état acceptant.

Automates à Epsilon Transitions

L'algorithme de Thompson procède en deux étapes :

1. On construit un Automate a ε-transitions a partir de l'expression rationnelle.
2. On construit un AFD reconnaissant le meme langage en supprimant les ε-transitions.

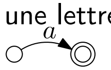
Algorithme de Thompson - Etape 1

Pour construire un automate avec ε-transitions reconnaissant le langage décrit par une E.R, Il suffit de donner une construction pour les trois opérations union, produit, étoile et d'utiliser la structure d'arbre de l'E.R.
A noter, les automates que l'on va construire vont tous vérifier :

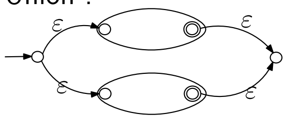
- un seul état initial, un seul état acceptant.
- aucune transition entrante dans l'état initial, aucune sortante de l'état acceptant
- Les automates ont exactement 2n états, où n est le nombre de symboles lettres, * et $+$.

Algorithme de Thompson - Etape 1

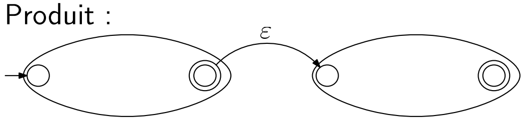
- une lettre :



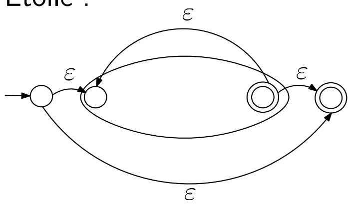
- Union :



- Produit :



- Etoile :



Algorithme de Thompson - Etape 2

On procède en deux étapes pour transformer l'automate de Thompson $\mathcal{A}$ en un AFD $\mathcal{A}' = (\Sigma', Q', {q_0}', F', \delta')$.
On définit la cloture epsilon d'un état q comme l'ensemble des états accessibles à partir de q par un chemin uniquement constitué de transitions ε.

1. Suppression des ε-transitions :
   - Q' = les états qui ont une transition entrante étiquetée par une lettre, plus l'état initial (qui reste l'état initial)
   - F' = les états dont la cloture epsilon contient l'état acceptant de $\mathcal{A}$.
   - On met une transition de q vers q' avec la lettre a si il existe un état q'' appartenant à la cloture epsilon de q et tel que $\delta(q'', a) = q'$.
     Il est souvent plus simple à cette étape de ne pas dessiner l'automate mais juste d'écrire la table de transitions.
2. Déterminisation de l'AFND obtenu.

Algorithme de Glushkov

Pour finir ce chapitre , on décrit maintenant un dernier algorithme pour passer de E.R. à automate.
Démo sur un exemple :

$$E = (ab + b)^*(bb + a^*)$$

On commence par linéariser l'expression rationnelle (une E.R. est dite linéaire si chaque symbole de lettre n'apparaît qu'une seule fois) :

$$E' = (a_1 a_2 + a_3)^*(a_4 a_5 + a_6^*)$$

Deux choses sont importantes a présent

1. On peut facilement trouver un automate pour une expression rationnelle linéaire
2. On peut facilement transformer l'automate reconnaissant $\mathcal{L}(E')$ en un automate reconnaissant $\mathcal{L}(E)$

Algorithme de Glushkov - ER linéaire

Considérons l'E.R. linéaire

$$E' = (a_1 a_2 + a_3)^*(a_4 a_5 + a_6^*)$$

Toute expression de ce type correspond à un langage local, c'est à dire uniquement défini par

1. les lettres qui peuvent être le début
2. les lettre qui peuvent être à la fin
3. les facteurs de taille 2 autorisés

Sur l'exemple ci dessus,

1. on peut commencer par $\{a_1, a_3, a_4, a_6, a_8\}$
2. on peut terminer par $\{a_5, a_7, a_8\}$

Algorithme de Glushkov - ER linéaire

$$E' = (a_1 a_2 + a_3)^*(a_4 a_5 + a_6^*)$$

Calculer les facteurs de taille 2 revient à décider quelle lettre peut suivre une lettre donnée
On écrit la table des successeurs et on en déduit la table de transitions de l'automate (on ajoute l'état initial et on marque les états finaux).

	Succ
a_1	a_2
a_2	a_1, a_3, a_4, a_6
a_3	a_1, a_3, a_4, a_6
a_4	a_5
a_5	$\emptyset$
a_6	a_6

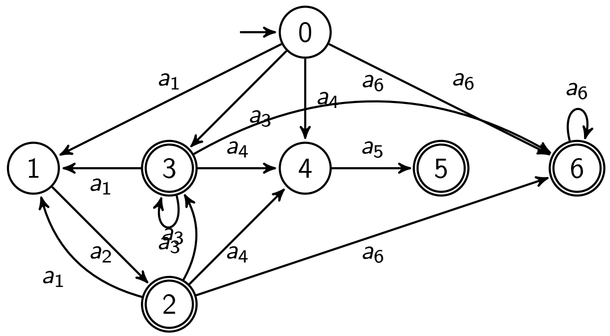
	Succ
0	1, 3, 4, 6
1	2
2	1, 3, 4, 6
3	1, 3, 4, 6
4	5
5	$\emptyset$
6	6

Algorithme de Glushkov - ER linéaire

$$E' = (a_1 a_2 + a_3)^*(a_4 a_5 + a_6^*)$$

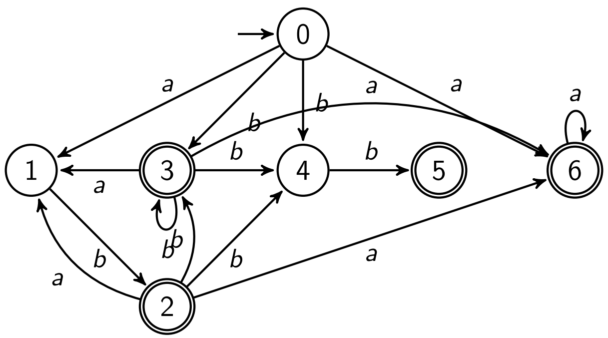
A partir de la table on définit l'automate $\mathcal{A}'$ a l'aide de la table en mettant une transition $\delta(i, a_j, j)$ à chaque fois que j est successeur de i.

	Succ
0	1, 3, 4, 6
1	2
2	1, 3, 4, 6
3	1, 3, 4, 6
4	5
5	$\emptyset$
6	6



Glushkov - Etape finale

Ensuite on n'a plus qu'à remplacer les lettres $a_1, ..., a_p$ par les lettres correspondantes de l'alphabet originel pour obtenir un automate $\mathcal{A}$ reconnaissant le langage $\mathcal{L}(E)$ décrit par l'E.R initiale E.



Glushkov - Preuve

Soit ϕ la fonction de Σ' vers Σ qui à a_i associe a ou b dans la linéarisation initiale.(Dans notre exemple $\phi(a_1) = a$, $\phi(a_2) = b$, etc...)

ϕ s'étend en une fonction de Σ'^* vers Σ^* : pour un mot $w = w_1...w_p$, $\phi(w) = \phi(w_1)...\phi(w_n)$.

Cela définit naturellement une fonction entre les langages, les automates et les expressions rationnelles sur l'alphabet Σ' vers ceux sur l'alphabet Σ.

- $\phi(L') = \{\phi(u),\ u \in L'\}$.
- $\phi(\mathcal{A}')$ est l'automate obtenu en remplaçant chaque lettre etiquette de transition par son image par ϕ
- $\phi(E')$ est l'expression rationnelle où on a remplacé chaque occurence d'une lettre par une occurence de son image par ϕ.

La première étape de Glushkov correspond bien a définir E' telle que $\phi(E') = E$, et la dernière étape correspond bien prendre transformer l'automate $\mathcal{A}'$ en l'automate $\phi(\mathcal{A}')$.

Glushkov - Preuve

L'algorithme de Glushkov marche car avec les notions précédentes on a bien :

- $\phi(\mathcal{L}(\mathcal{A}')) = \mathcal{L}(\phi(\mathcal{A}'))$ pour tout automate $\mathcal{A}'$
- $\phi(\mathcal{L}(E')) = \mathcal{L}(\phi(E'))$ pour toute expression rationnelle E'

Pour les automates cela est facile puisque un mot reconnu correspond à un chemin de l'état initial vers un état final, et ceux ci sont preservés par ϕ.

Pour les expression rationnelles, on peut le prouver par récurrence sur l'expression rationnelle, il suffit alors de montrer que

- $\phi(\mathcal{L}(E_1 + E_2)) = \mathcal{L}(\phi(E_1)) \cup \mathcal{L}(\phi(E_2))$
- $\phi(\mathcal{L}(E_1.E_2)) = \mathcal{L}(\phi(E_1)).\mathcal{L}(\phi(E_2))$
- $\phi(\mathcal{L}(E_1^*)) = \mathcal{L}(\phi(E_1))^*$

Glushkov - Preuve

On a d'abord défini ϕ comme fonction des lettres Σ' vers Σ avant de l'étendre aux mots.

La plupart des choses exposés précédemment fonctionnent de façon identique (ou presque) si on démarre directement avec ϕ fonction de Σ'^* vers Σ^*, à condition qu'elle vérifie

pour tous mots u et v : $\phi(u.v) = \phi(u).\phi(v)$

On dit alors que ϕ est un morphisme de Σ'^* vers Σ^*.

En particulier, on peut montrer

Proposition

Soit ϕ morphisme de Σ_1^* vers Σ_2^*.
Si $L_1 \in Rec(\Sigma_1)$, alors $\phi(L_1) \in Rec(\Sigma_2)$.
Si $L_2 \in Rec(\Sigma_2)$, alors $\phi^{-1}(L_2) \in Rec(\Sigma_1)$.