Chapitre trois

Automates

MPI/MPI*, lycée Faidherbe

Résumé

Ce chapitre met en place des langages caractérisés par le fait qu'il existe une méthode simple pour tester l'appartenance d'un mot à ce langage. On donnera plusieurs définitions des automates qui, toutes,permettront de reconnaître les mêmes langages.

I Automates déterministes

I.1 Transitions

La première étape va être de donner un cadre aux commandes et états.

Définition 1 : machine

Une machine est un triplet (Σ, S, δ) où

- Σ est un ensemble fini, l'alphabet, ses éléments seront les lettres,
- S est un ensemble fini, l'ensemble des états,
- δ est une fonction de $S \times \Sigma$ vers S, la fonction de transition.

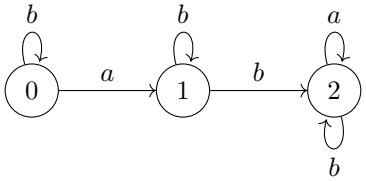
Une machine est caractérisée par la fonction de transition ; en effet l'alphabet et l'ensemble des états sont les composantes de son ensemble de départ. Voici un exemple :

δ	0	1	2
a	1	1	2
b	0	2	2

Définition 2 : transition

Une transition d'une machine (Σ, S, δ) est un triplet (s, a, s') tel que $\delta(s, a) = s'$.
On notera aussi $s.a = s'$ ou $s \xrightarrow{a} s'$.
s est l'origine de la transition, s' en est l'extrémité et a son étiquette.

Une représentation peut être faite par un graphe valué par Σ : S est l'ensemble des sommets et les arêtes sont les transitions.
Contrairement aux graphes vus jusqu'à présent les arêtes d'un sommet vers lui-même sont possibles et il peut y avoir plusieurs arêtes d'un sommet s vers un sommet s' : elles auront alors des étiquettes distinctes.
On peut noter que, si n est le cardinal de Σ, il y a n arêtes sortantes depuis tout sommet, étiquetées par les n lettres de l'alphabet.



Une suite finie de commandes est une suite de lettres donc un mot.
On peut associer une transition à un mot en composant les transitions des lettres : on fait le choix de lire les lettres de la gauche vers la droite.

Définition 3 : action d'un mot

Si $M = (\Sigma, S, \delta)$ est une machine sur le langage Σ on prolonge la fonction de transition à Σ^* tout entier par la fonction δ^* de $S \times \Sigma^*$ vers S avec

- $\delta^*(s, \varepsilon) = s$ pour tout $s \in S$
- $\delta^*(s, x.u) = \delta^*\big(\delta(s, x), u\big)$ pour $u \in \Sigma^*$ et $x \in \Sigma$.

On notera encore $\delta^*(s, u) = s.u$ ou $s \xrightarrow{u} s'$, si $s' = s.u$.
On peut donc écrire $s.(u.x) = (s.u).x$ pour $u \in \Sigma^*$ et $x \in \Sigma$. Pour $u = x_1 x_2 \cdots x_n$, on a $s.u = \big((\cdots((s.x_1).x_2).\cdots.x_{n-1}).x_n\big)$.
Si on pose $s_1 = \delta(s, x_1)$, $s_2 = \delta(s_1, x_2)$, ..., $s' = s_n = \delta(s_{n-1}, x_n)$ alors $(s, s_1, s_2, \ldots, s_n)$ est le **parcours** (ou **calcul**) de u à partir de s.
On notera ce parcours sous la forme

$$s \xrightarrow{x_1} s_1 \xrightarrow{x_2} s_2 \xrightarrow{x_3} \quad \cdots \quad \xrightarrow{x_{n-1}} s_{n-1} \xrightarrow{x_n} s_n$$

Théorème 1 : transition par un produit

$\delta^*(s, u.v) = \delta^*\big(\delta^*(s, u), v\big)$ pour tous $u, v \in \Sigma^*$.

Exercice 1 **Solution page 21**
Prouver le théorème.

I.2 Automates déterministes complets

Pour l'instant les parcours de la machine se font sans condition de point de départ et tous les parcours ont la même valeur. Nous allons enrichir la machine en lui ajoutant un point de départ et des sommets d'arrivée souhaités.

Définition 4 : automate déterministe complet

Un automate fini déterministe ou A.F.D. (DFA en anglais)
est un quintuplet $Q = (\Sigma, S, \delta, s_0, T)$ où

- (Σ, S, δ) est une machine
- s_0 est un élément de S, l'état initial,
- T est une partie de S, l'ensemble des états finaux,
  on dit aussi accepteurs, terminaux voire même finals !

Un automate va servir à tester les mots : a-t-on $\delta^*(s_0, u) \in T$?
On dit alors que u est **reconnu** par l'automate.

Définition 5 : langage reconnu

- Le langage reconnu (ou accepté) par un automate $Q = (\Sigma, S, \delta, s_0, T)$ est l'ensemble $L(Q)$ de tous les mots reconnus par Q.

$$L(Q) = \{u \in \Sigma^* \ ; \ \delta^*(s_0, u) \in T\}$$

- Un langage L est **reconnaissable** s'il existe un a.f.d. complet Q tel que $L = L(Q)$.

Exercice 2 **Solution page 21**

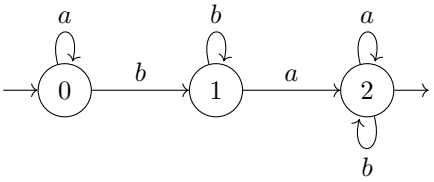
$Q = (\Sigma, S, \delta, s_0, T)$ est un automate.

- Si $T = \emptyset$ alors $L(Q) = \emptyset$.
- Si $T = S$ alors $L(Q) = \Sigma^*$.
- $L(Q)$ contient ε si et seulement si s_0 appartient à T.

Pour représenter graphiquement un automate

- l'état initial est indiqué par une flèche rentrante
- un état terminal est entouré d'un double cercle ou est marqué par une flèche sortante.

Dans l'exemple donné ci-dessus,on pose $s_0 = 0$ et $T = \{2\}$.

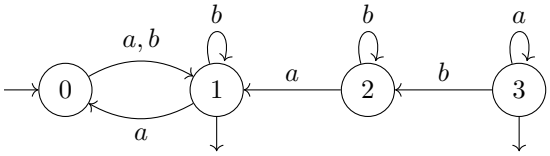


Le langage reconnu est l'ensemble des mots contenant ba.

I.3 Émondage

Nous allons dans la suite déterminer des automates différents en conservant le langage. La première possibilité est d'enlever les états non utilisés.

Dans l'automate ci-dessous on voit qu'aucun parcours ne passera par les états 2 ou 3.



On va essayer de les retirer.

Définition 6 : accessibilité

$Q = (\Sigma, S, \delta, s_0, T)$ est un automate déterministe complet.
Un état $s \in S$ est dit accessible s'il existe $u \in \Sigma^*$ tel que $s = \delta^*(s_0, u)$.

Les états accessibles sont les seuls états visibles depuis s_0.

Théorème 2 : stabilité des états accessibles

Si S' est l'ensemble des états accessibles d'un automate $Q = (\Sigma, S, \delta, s_0, T)$ alors, pour tout $s \in S'$ et pour tout $a \in \Sigma$, $\delta(s, a)$ appartient à S'.

Exercice 3 **Solution page 21**

Prouver que, si S' est l'ensemble des états accessibles d'un automate $Q = (\Sigma, S, \delta, s_0, T)$ alors, pour tout $s \in S'$ et pour tout $a \in \Sigma$, $\delta(s, a)$ appartient à S'.
Ce signifie que la restriction de δ à $S' \times \Sigma$ qu'on note δ', est à valeur dans S'.
Montrer que, pour tous $s \in S'$ et $u \in \Sigma^*$, $\delta'^*(s, u) = \delta^*(s, u)$.

Définition 7 : automate émondé

Si S' est l'ensemble des états accessibles d'un automate $Q = (\Sigma, S, \delta, s_0, T)$, l'automate émondé associé à Q est l'automate déterministe complet

$$Q' = (\Sigma, S', \delta', s_0, T \cap S')$$

Théorème 3 conservation du langage

Avec les notations ci-dessus, on a $L(Q) = L(Q')$.

Exercice 4 **Solution page 21**

Prouver le théorème.

I.4 Automates déterministes incomplets

Définition 8 : automate déterministe incomplet

Un automate fini déterministe incomplet est un quintuplet $Q = (\Sigma, S, \delta, s_0, T)$ où

- Σ est un alphabet (fini),
- S est un ensemble fini d'états,
- δ est une application définie sur une **partie** de $S \times \Sigma$ vers S : la fonction de transition.
- s_0 est un élément de S, l'état initial,
- T est une partie de S, l'ensemble des états finaux,

Définition 9 : mots bloquants

$Q = (\Sigma, S, \delta, s_0, T)$ est un automate déterministe incomplet.
$u = u_1 u_2 \cdots u_p$ est un mot sur Σ .
Si $\delta(s_0, u) = s_1$, $\delta(s_1, u_2) = s_2$, …, $\delta(s_{k-1}, u_k) = s_k$ sont définis mais δ n'est pas défini en (s_k, u_{k+1}) (avec $k < p$) on dit que u est un mot bloquant.

Définition 10 : langage reconnu

Le langage reconnu par un automate incomplet est l'ensemble des mots u non bloquants tels que $\delta^*(s_0, u)$ appartient à T.

On va transformer un automate incomplet en automate complet. L'idée est d'ajouter un état-puits qui va recevoir toutes les transitions manquantes et qui reste stable par toutes les lettres.

Définition 11 : complétio

$Q = (\Sigma, S, \delta, s_0, T)$ est un automate incomplet.
On choisit p n'appartenant pas à S et on définit $S' = S \cup \{p\}$.
On prolonge δ en δ' défini sur $S' \times \Sigma$ par

- $\delta'(s, x) = \delta(s, x)$ pour $s \in S$ si δ est défini en (s, x),
- $\delta'(s, x) = p$ pour $s \in S$ si $\delta(s, x)$ n'est pas défini et
- $\delta'(p, x) = p$

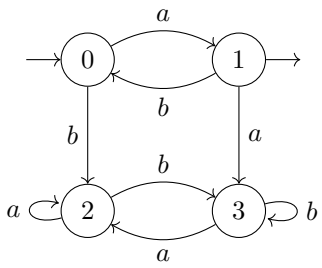
Théorème 4 conservation du langage

$Q' = (\Sigma, S', \delta', s_0, T)$ est un automate déterministe complet tel que $L(Q) = L(Q')$.

Exercice 5 **Solution page 21**

Prouver le théorème

Inversement, on peut souhaiter ne garder que les états utiles, au prix d'un automate qui devient incomplet.



Les états 2 et 3 représentent une sorte de piège : dès qu'on y parvient on ne pourra jamais revenir à un état terminal. La tentation de les enlever se heurte au fait qu'alors il existerait des transitions manquantes.

Définition 12 : co-accessibilité

$Q = (\Sigma, S, \delta, s_0, T)$ est un automate déterministe.
Un état $s \in S$ est dit co-accessible s'il existe $u \in \Sigma^*$ non bloquant tel que $s.u \in T$.

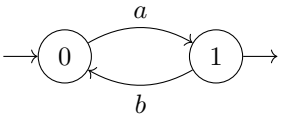
Théorème 5 conservation du langage

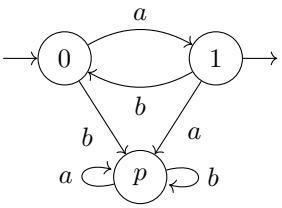
Si S' est l'ensemble des états co-accessibles de Q alors
la restriction de δ à $S' \times \Sigma$, notée δ' est à valeurs dans S' et
$Q' = (\Sigma, S', \delta', s_0, T \cap S')$ est un automate déterministe incomplet tel que $L(Q) = L(Q')$.

Exercice 6 **Solution page 21**

Prouver le théorème

Dans l'exemple ci-dessus on aboutit ainsi à l'automate





On peut le compléter ensuite

I.5 Opérations ensemblistes

Nous allons prouver ici quelques stabilités des langages reconnaissables.

On rappelle que le complémentaire d'un langage L sur Σ est $\overline{L} = \Sigma^* \setminus L$.

Théorème 6

Le complémentaire d'un langage reconnaissable est reconnaissable.

Exercice 7 Solution page 21
Prouver le théorème.

Pour l'union et le produit nous auront besoin d'une construction de machine.

Définition 13 : produit de machines

Si $M_1 = (\Sigma, S_1, \delta_1)$ et $M_2 = (\Sigma, S_2, \delta_2)$ sont deux machines sur le même alphabet leur produit $M_1 \odot M_2$ est la machine
$(\Sigma, S_1 \times S_2, \delta_1 \times \delta_2)$ avec $\delta_1 \times \delta_2\big((s,t),a\big) = \big(\delta_1(s,a), \delta_2(t,a)\big)$.

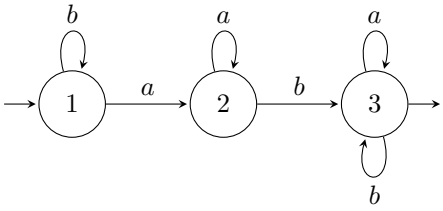
Exercice 8 Solution page 21
Prouver que $(\delta_1 \times \delta_2)^*\big((s,t),u\big) = \big(\delta_1^*(s,u), \delta_2^*(t,u)\big)$.

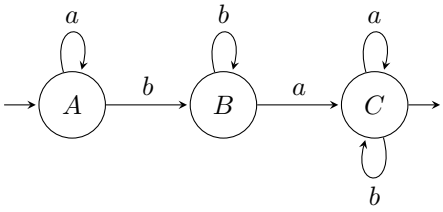
Théorème 7

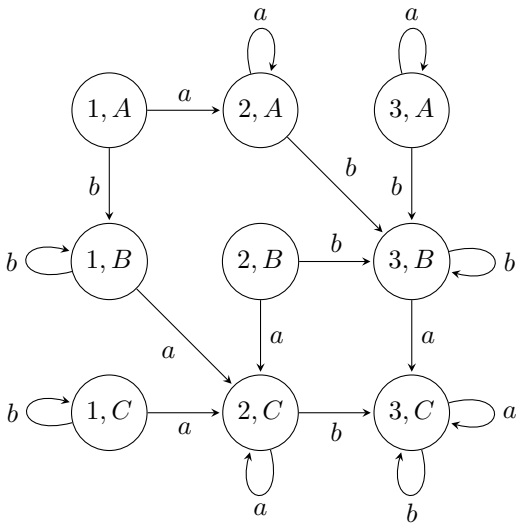
L'intersection de deux langages reconnaissables est reconnaissable.
L'union de deux langages reconnaissables est reconnaissable.

Démonstration Soient L_1 et L_2 reconnus respectivement par $Q_1 = (\Sigma, S_1, \delta_1, s_{0,1}, T_1)$ et $Q_2 = (\Sigma, S_2, \delta_2, s_{0,2}, T_2))$. On choisit $\sigma_0 = (s_{1,0}, s_{2,0})$ pour état initial.
On remarque qu'un mot u est reconnu par Q_1 (resp. Q_2) si et seulement si
$(\delta_1 \times \delta_2)^*(\sigma_0.u \in T_1 \times S_2$ (resp. $S_1 \times T_2)$. Alors
$L_1 \cup L_2$ est reconnu par $(\Sigma, S_1 \times S_2, \delta_1 \times \delta_2, (s_{1,0}, s_{2,0}), T_1 \times S_2 \cup S_1 \times T_2)$ et
$L_1 \cap L_2$ est reconnu par $(\Sigma, S_1 \times S_2, \delta_1 \times \delta_2, (s_{1,0}, s_{2,0}), T_1 \times T_2)$
car $T_1 \times S_2 \cap S_1 \times T_2 = T_1 \times T_2$ . . . . . ■

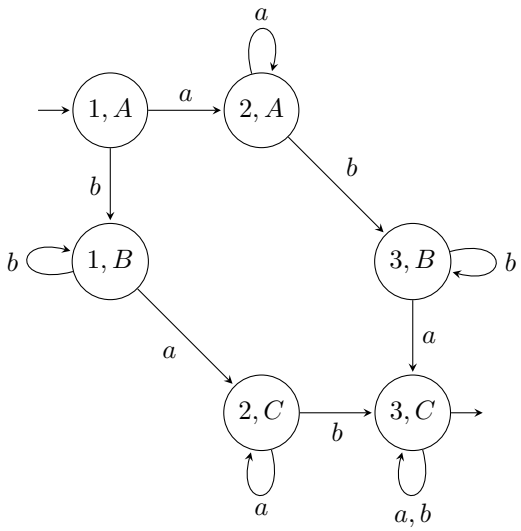
Exemple Les automates suivants reconnaissent Le produit des machines associées est
respectivement L_1, ensemble des mots contenant
ab et L_2, ensemble des mots contenant ba.

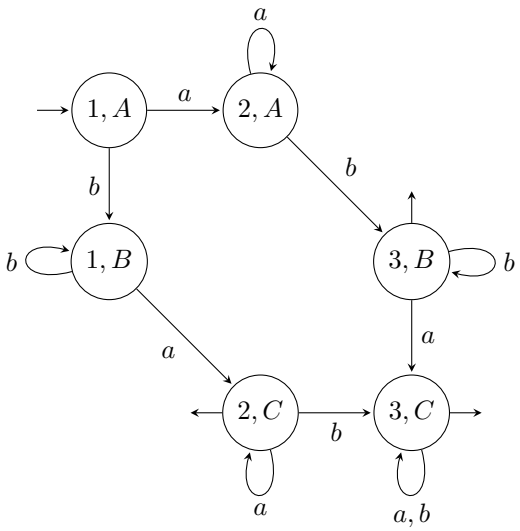






Après élagage, le langage $(L_1 \cap L_2)$ des mots qui contiennent ab et ba et le langage $(L_1 \cup L_2)$ des mots qui contiennent ab ou ba sont reconnus par les automates





II Automates non-déterministes

Les automates vus jusqu'à présent envoyait un état vers un autre état bien déterminé (ou vers aucun état) sous l'action d'une lettre.
Nous allons maintenant lever cette contrainte d'un état unique et considérer qu'une lettre peut déterminer plusieurs transitions depuis un état.
Ce non-déterminisme sera à l'œuvre aussi au départ : on autorisera plusieurs états initiaux.

II.1 Définitions

Définition 14 : automate fini non-déterministe

Un automate fini non-déterministe ou AFND (NFA en anglais)
est un quintuplet $(\Sigma, S, \Delta, I, T)$ composé de :

- Σ, un alphabet,
- S, l'ensemble fini des états de l'automate,
- Δ une application de $S \times \Sigma$ dans $\mathcal{P}(S)$,
- $I \subset S$, l'ensemble des états initiaux,
- $T \subset S$, l'ensemble des états acceptants,

Un automate déterministe peut ainsi être considéré comme un cas particulier d'automate non–déterministe. Dans le cas d'un automate déterministe, pour tout couple $(s, x) \in S \times \Sigma$, $\Delta(s, x)$ admet exactement un élément dans le cas des automates complets ou au plus un élément dans le cas des automates incomplets.

On peut remplacer la fonction de transition par son graphe $G_\Delta = \big\{(s, x, s') \ ; \ s' \in \Delta(s, x)\big\}$.
G_Δ est une partie de $S \times \Sigma \times S$, c'est l'ensemble des transitions.
Δ est alors défini par $\Delta(s, x) = \big(\{s\} \times \{x\} \times S\big) \cap G_\Delta = \big\{s' \in S \ ; \ (s, x, s') \in G_\Delta\big\}$.

On notera encore la transition $(s, x, s') \in G_\Delta$ par $s \xrightarrow{x} s'$.

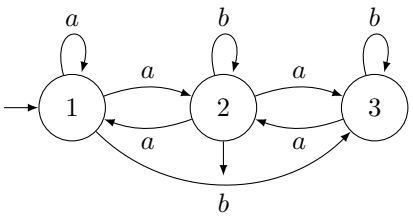
Définition 15 : calcul

Un calcul de l'automate est une suite (finie) de transitions telle que l'origine de chacune est l'extrémité de la précédente. Il sera noté

$$s \xrightarrow{x_1} s_1 \xrightarrow{x_2} s_2 \xrightarrow{x_3} \quad \cdots \quad \xrightarrow{x_{n-1}} s_{n-1} \xrightarrow{x_n} s_n$$

- s est l'origine du calcul,
- s_n est l'extrémité du calcul,
- $u = x_1 x_2 \cdots x_n$ est l'étiquette du calcul.

La principale différence avec les automates déterministe est qu'il y avait une bijection entre les parcours et leur étiquette alors que, pour automate non-déterministe, un même mot peut être associé à plusieurs calculs depuis une même origine.
Exemple : dans l'automate



$1 \xrightarrow{a} 1 \xrightarrow{a} 1 \xrightarrow{b} 3$,
$1 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{b} 2$,
$1 \xrightarrow{a} 2 \xrightarrow{a} 1 \xrightarrow{b} 3$,
$1 \xrightarrow{a} 2 \xrightarrow{a} 3 \xrightarrow{b} 3$
sont les calculs d'étiquette aab.

Définition 16 : langage associé

- Un calcul est **réussi** si son origine appartient à I et son extrémité appartient à T.
- Un mot est **reconnu** (ou accepté) par l'automate s'il est l'étiquette d'au moins un calcul réussi.
- Le **langage reconnu** (ou langage accepté) par un automate M est l'ensemble, $L(M)$, des mots reconnus par l'automate.

II.2 Déterminisation

Les automates non-déterministes sont une généralisation forte des automates déterministes ; cependant ils ne créent pas de nouveaux langages reconnus .

Définition 17 : automate des parties

$Q = (\Sigma, S, \Delta, I, T)$ est un automate non-déterministe.
L'automate des parties associé à Q est $Q' = (\Sigma, \mathcal{P}(S), \delta, I, \mathcal{F})$ avec

- $\delta(E, x) = \bigcup_{s \in E} \Delta(s, x) = \{s' \in S \ ; \ \exists s \in E, \ (s, x, s') \in G_\Delta\}$.
- $\mathcal{F} = \{E \subset S \ ; \ E \cap T \neq \emptyset\}$

On dit aussi que Q' est le déterminisé de Q.

On notera que I était une partie de S mais devient un élément de $\mathcal{P}(S)$.

Théorème 8

Avec les notations ci-dessus

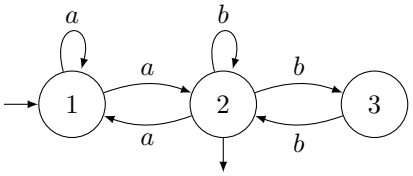
- Q' est un automate déterministe complet.
- $L(Q') = L(Q)$.

Exercice 9 **Solution page 21**

Prouver le théorème.

Si l'automate initial est non-déterministe avec n états, le déterminisé admet 2^n états.
Dans le calcul pratique on ne calculera les images des parties de S que pour les parties accessibles : on part de I on calcule $\delta(I, x)$ pour les différentes lettres de Σ puis les images de ces derniers ensembles et ainsi de suite : on calcule seulement l'automate émondé.

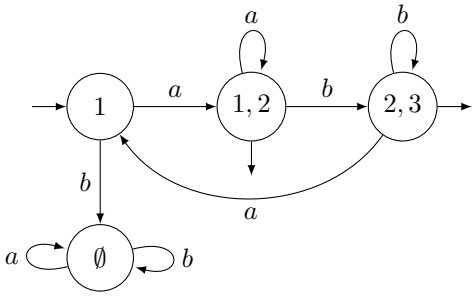
Exemple :
On part de l'automate non déterministe



On calcule seulement les transitions depuis les états utiles (les états accessibles).

Q'	$\{1\}$	$\{1,2\}$	$\{2,3\}$	$\emptyset$
a	$\{1,2\}$	$\{1,2\}$	$\{1\}$	$\emptyset$
b	$\emptyset$	$\{2,3\}$	$\{2,3\}$	$\emptyset$

On obtient l'automate



III Automates asynchrones

Définition 18 : automate asynchrone

Un **automate avec transitions spontanées** (ou asynchrone) sur le langage Σ, avec Σ ne contenant pas ε, est un automate (non déterministe) sur le langage $\Sigma \cup \{\varepsilon\}$.

Autrement dit on ajoute à un automate des transitions de la forme $s \xrightarrow{\varepsilon} s'$ qui se font sans qu'il soit nécessaire d'avoir une lettre qui la commande.
Pour noter la différence entre ε et les lettres, on note $Q = (\Sigma, S, \Delta, I, F, \varphi)$ un tel automate où Δ est toujours la fonction de transition de $S \times \Sigma$ vers $\mathcal{P}(S)$ et $\varphi(s)$ est l'ensemble des extrémités des transitions spontanées d'origine s.
Si $s \xrightarrow{u_1} s_1 \xrightarrow{u_2} \cdots \xrightarrow{u_{n-1}} s_{n-1} \xrightarrow{u_n} s_n$ est un calcul son étiquette est le mot sur Σ,
$u = u_1 u_2 \cdots u_n$, qui peut être de longueur inférieure à n.
Le langage reconnu est toujours l'ensemble des étiquettes des calculs réussis.

Synchronisation

Définition 19 : fermeture

La ε-fermeture d'un état s, $\kappa(s)$, est l'ensemble des états accessibles depuis s par une suite finie de transitions spontanées : s' appartient à $\kappa(s)$ si et seulement si il existe un calcul de la forme $s \xrightarrow{\varepsilon} s_1 \xrightarrow{\varepsilon} s_2 \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} s_{n-1} \xrightarrow{\varepsilon} s'$.
Si $K_0 = \{s\}$ et $K_{n+1} = \bigcup_{q \in K_n} \varphi(q)$ alors la suite (K_n) est une suite de parties de S et $\kappa(s) = \bigcup_{n \in \mathbb{N}} K_n$.

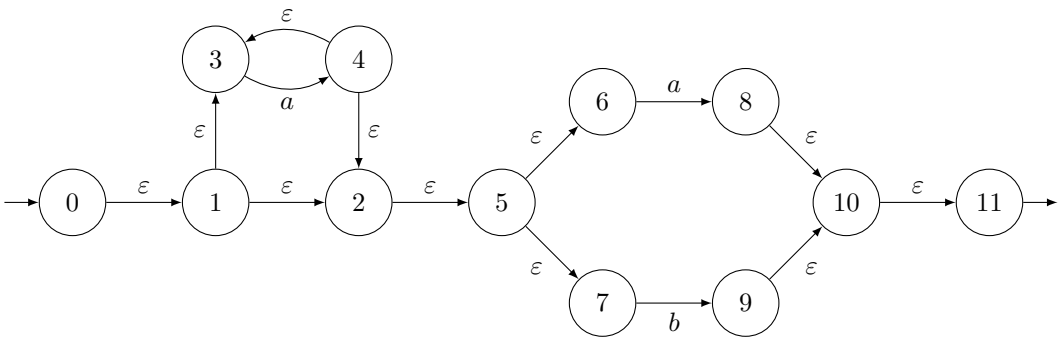
On généralise la ε-fermeture à un sous-ensemble : pour tout partie $S' \subset S$, $\kappa(S') = \bigcup_{s \in S'} \kappa(s)$.

Théorème 9

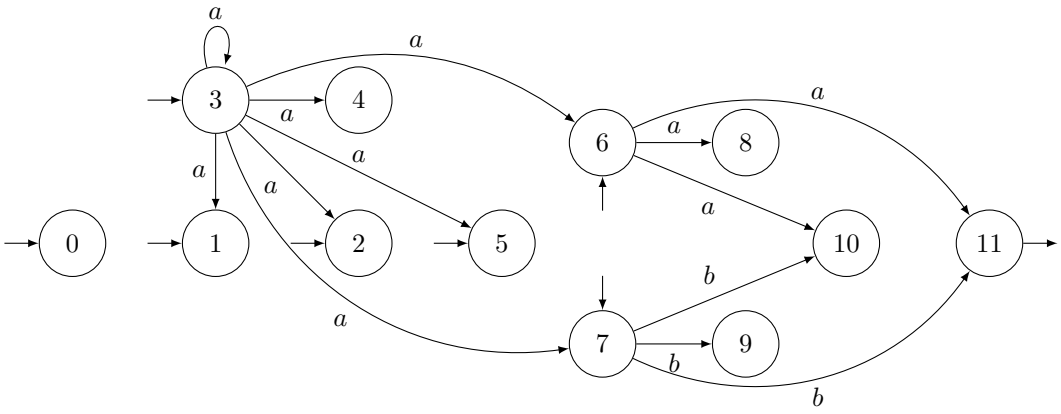
Si $Q = (\Sigma, S, \Delta, I, T, \varphi)$ un automate avec transitions spontanées et si $I' = \kappa(I)$ et $\Delta'(s,x) = \kappa\big(\Delta(s,x)\big)$ alors $Q' = (\Sigma, S, \Delta', I', T)$ est un automate (non déterministe) sans transition spontanée qui reconnaît le même langage que Q.

Exercice 10 **Solution page 22**
Prouver le théorème.

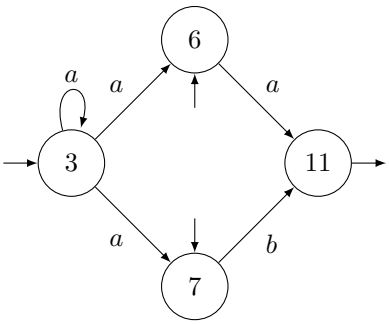
Exemple

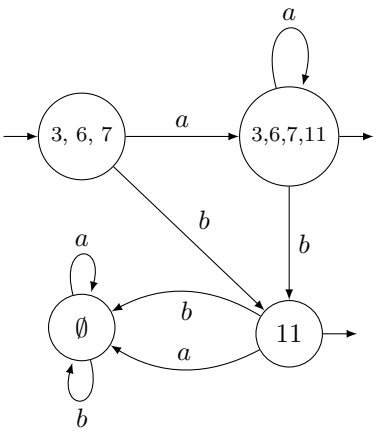


Le processus ci-dessus donne alors



On peut émonder et déterminiser.





IV Construction d'automates

IV.1 Automates de Glushkov

L est un langage local sur Σ défini par (P, F, S) et un booléen b_ε marquant si L contient ε ou non.

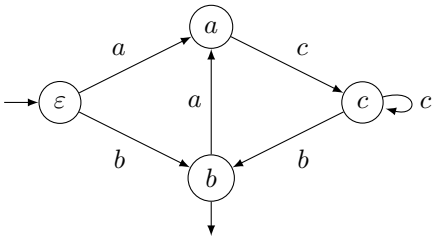
Définition 20

Automate de Glushkov L'automate de Glushkov associé à L est l'automate incomplet $Q = (\Sigma, \Sigma \cup \{\varepsilon\}, \delta, \varepsilon, S')$ où δ est définie par

- $\delta(\varepsilon, x)$ existe si et seulement si $x \in P$, alors $\delta(\varepsilon, x) = x$,
- $\delta(x, y)$ existe si et seulement si $xy \in F$, alors $\delta(x, y) = y$.

et $S' = S \cup \{\varepsilon\}$ si b_ε vaut vrai, $S' = S$ sinon.

Exemple L est le langage local sur $\Sigma = \{a, b, c\}$ défini par $P = \{a, b\}$, $S = \{b\}$, $F = \{ac, ba, cb, cc\}$ et b vaut faux.



On remarquera que toutes les transitions de même étiquette ont la même extrémité.

Théorème 10 de Glushkov

Si Q est l'automate de Glushkov associé à L alors Q reconnaît L.

Exercice 11 **Solution page 22**

Prouver le théorème

V Construction d'automates

V.1 Construction de Thomson

On va construire un automate $Q_{Th}(r)$ non déterministe avec transitions spontanées associé à une expression rationnelle $\mathtt{r}$: $L(\mathtt{r}) = L\big(Q_{Th}(\mathtt{r})\big)$.
À chaque étape l'automate sera **normalisé** :

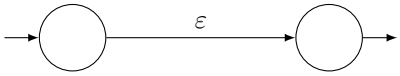
- il y a un seul état initial
- il y a un seul état final
- l'état initial n'est l'extrémité d'aucune transition
- l'état final n'est l'origine d'aucune transition

Par contre il y aura souvent beaucoup de transitions spontanées.

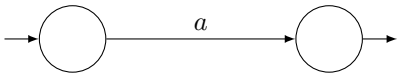
Langages élémentaires

Les automates suivant reconnaissent les langages élémentaires


$Q_{Th}(\varnothing)$



$Q_{Th}(\epsilon)$



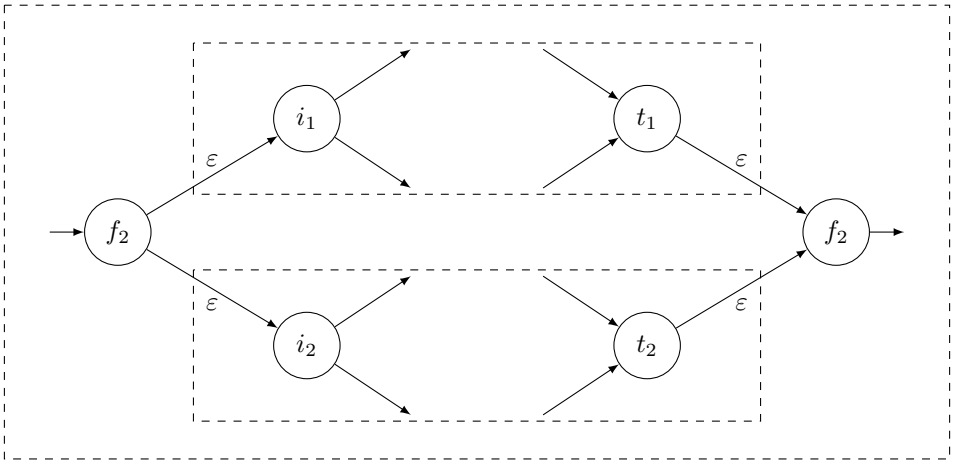
$Q_{Th}(\mathtt{a})$

Union

On suppose définis les automates de Thomson $Q_{Th}(\nabla_1) = \big(\Sigma, S_1, \Delta_1, \{i_1\}, \{t_1\}, \varphi_1\big)$ et $Q_{Th}(\nabla_2) = \big(\Sigma, S_2, \Delta_2, \{i_2\}, \{t_2\}, \varphi_2\big)$ reconnaissant respectivement $L[\mathtt{r}_1]$ et $L[\mathtt{r}_2]$.
On pose $Q_{Th}(\mathtt{r}_1|\mathtt{r}2) = \big(\Sigma, S, \Delta, \{i\}, \{t\}, \varphi\big)$ avec

- $S = S_1 \cup S_2 \cup \{i, f\}$ (supposé disjoints)
- $\Delta(s,x) = \Delta_1(s,x)$ pour $s \in S_1$, $\Delta(s,x) = \Delta_2(s,x)$ pour $s \in S_2$, $\Delta(i,x) = \emptyset$ et $\Delta(f,x) = \emptyset$
- $\varphi(s) = \varphi_1(s)$ pour $s \in S_1 \setminus \{t_1\}$, $\varphi(t_1) = \{t\}$,
  $\varphi(s) = \varphi_2(s)$ pour $s \in S_2 \setminus \{t_2\}$, $\varphi(t_2) = \{t\}$,
  $\varphi(i) = \{i_1, i_2\}$ et $\varphi(f) = \emptyset$

Exercice 12
Prouver que $Q_{Th}(\mathtt{r}_1|\mathtt{r}_2)$ reconnaît $L[\mathtt{r}_1|\mathtt{r}_2]$.

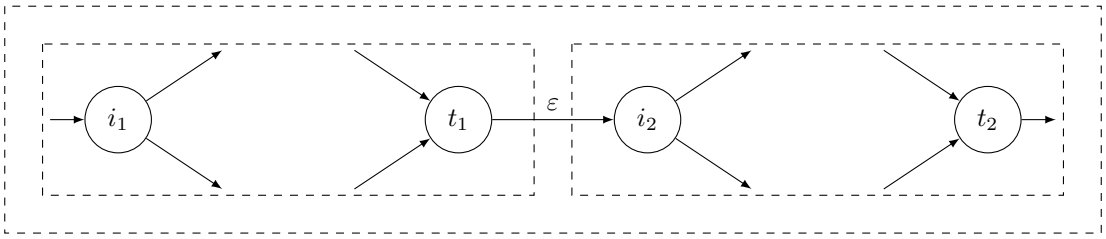


Produit

On garde les mêmes notations que ci-dessus.
On pose $Q_{Th}(\mathtt{r}_1.\mathtt{r}2) = \big(\Sigma, S, \Delta, \{i\}, \{t\}, \varphi\big)$ avec

- $S = S_1 \cup S_2$ (supposé disjoints),
- $\Delta(s, x) = \Delta_1(s, x)$ pour $s \in S_1$, $\Delta(s, x) = \Delta_2(s, x)$ pour $s \in S_2$,
- $\varphi(s) = \varphi_1(s)$ pour $s \in S_1 \setminus \{t_1\}$, $\varphi(t_1) = \{i_2\}$ et $\varphi_2(s)$ pour $s \in S_2$.

Exercice 13
Prouver que $Q_{Th}(\mathtt{r}_1.\mathtt{r}_2)$ reconnaît $L[\mathtt{r}_1.\mathtt{r}_2]$.

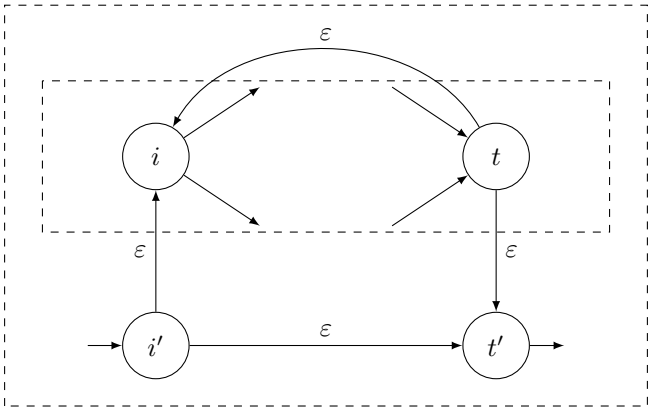


Étoile

On suppose définis l'automate de Thomson $Q_{Th}(\nabla) = \big(\Sigma, S, \Delta, \{i\}, \{t\}, \varphi\big)$ reconnaissant respectivement $L[\mathtt{r}]$.
On pose $Q_{Th}(\mathtt{r}*) = \big(\Sigma, S', \Delta', \{i'\}, \{t'\}, \varphi'\big)$ avec

- $S = S \cup \{i', f'\}$ (supposé disjoints)
- $\Delta'(s, x) = \Delta(s, x)$ pour $s \in S$, $\Delta'(i, x) = \Delta(t, x) = \emptyset$,
- $\varphi(s) = \varphi_1(s)$ pour $s \in S \setminus \{t\}$, $\varphi(t) = \{i, t'\}$, $\varphi(i') = \{i, t'\}$ et $\varphi(t') = \emptyset$

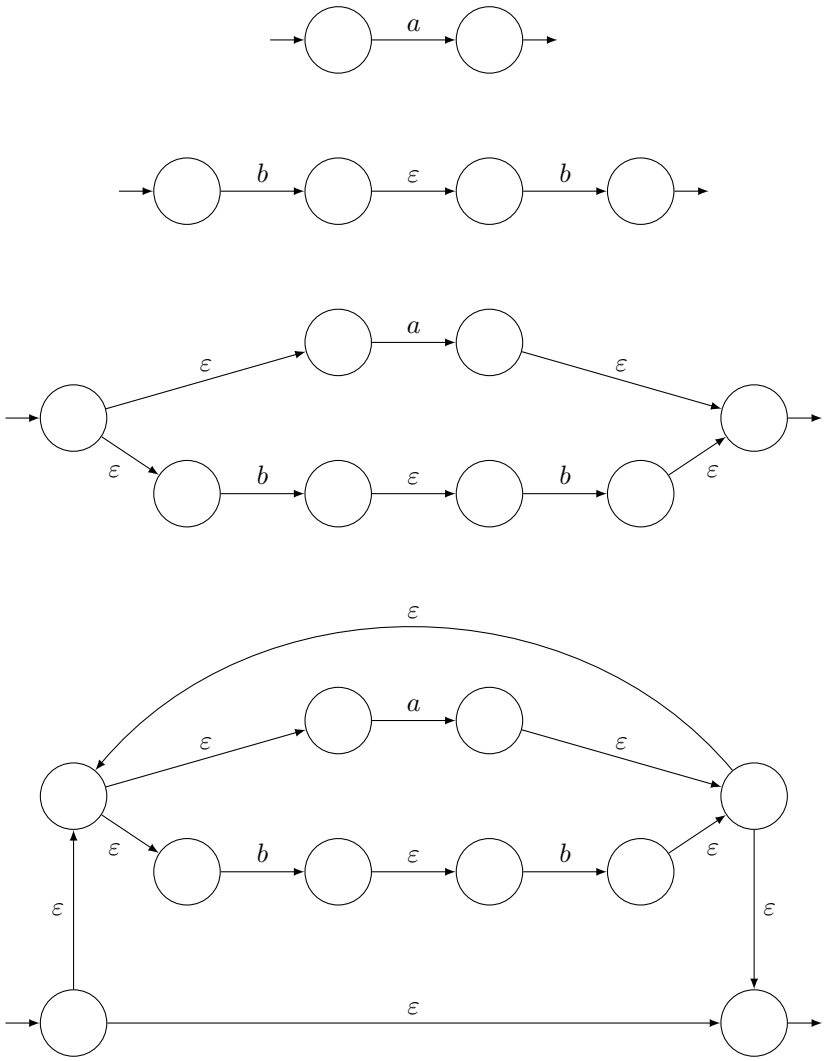
Exercice 14
Prouver que $Q_{Th}(\mathtt{r}*)$ reconnaît $L[\mathtt{r}*]$.



La démonstration de la propriété $L(\mathtt{r}) = L\big(Q_{Th}(\mathtt{r})\big)$ est donc démontrée par induction structurelle sur les expressions régulières.
On a ainsi la propriété : **tout langage rationnel est reconnaissable.**

Exemple

On va construire $Q_{Th}\big((\mathtt{a}|\mathtt{b}.\mathtt{b})*\big)$.



V.2 L'algorithme de Berry-Sethi

Morphismes

On considère une fonction f d'un alphabet Σ' vers un alphabet Σ.
Les mots sont des produits de lettres : on peut donc définir une fonction f^* de Σ'^* vers Σ^* par $f^*(\varepsilon) = \varepsilon$ (le premier ε est le mot vide de Σ'^*, le second celui de Σ^*)
et $f^*(u.x) = f^*(u).f(x)$ pour $u \in \Sigma'^*$ et $x \in \Sigma'$.

Exercice 15 **Solution page 22**
Prouver que si f est une fonction de Σ' vers Σ alors $f^*(u.v) = f^*(u).f^*(v)$ pour tous mots u et v de Σ'^*.

On prolonge f^* sur les langages sans changer le nom :
si L est un langage sur Σ', $f^*(L) = \{f^*(u)\ ;\ u \in L\}$ est un langage sur Σ.

Exercice 16 - Opération rationnelles **Solution page 22**
Prouver que si L_1 et L_2 sont des langages sur Σ' alors

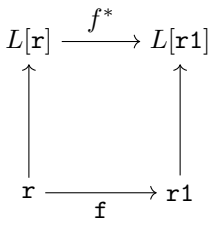
1. $f^*(L_1 \cup L_2) = f^*(L_1) \cup f^*(L_2)$,
2. $f^*(L_1.L_2) = f^*(L_1).f^*(L_2)$
3. $f^*(L_1^*) = \big(f^*(L_1)\big)^*$.
4. A-t-on toujours $f^*(L_1 \cap L_2) = f^*(L_1) \cap f^*(L_2)$?

De même on prolonge f de manière inductive sur l'ensemble des expressions régulières sur Σ' :
`f(∅ ) = ∅` , `f(ϵ) = ϵ`, `f(a) = b` avec $b = f(a)$, $\texttt{f}(\texttt{r}_1|\texttt{r}_2) = (\texttt{f}(\texttt{r}_1)|\texttt{f}(\texttt{r}_2)$,
$\texttt{f}(\texttt{r}_1.\texttt{r}_2) = (\texttt{f}(\texttt{r}_1).\texttt{f}(\texttt{r}_2))$ et $\texttt{f}(\texttt{r}*) = (\texttt{f}(\texttt{r}))*$.

Théorème 11

Si f est une fonction de Σ' vers Σ alors $L\big[\texttt{f(r)}\big] = f^*\big(L[\texttt{r}]\big)$
pour toute expression régulière `r` sur Σ'.

On a donc un transport depuis les expressions régulières vers les langages.



Dans ce diagramme les deux chemins de `r` à L[`r1`] donnent le même résultat.

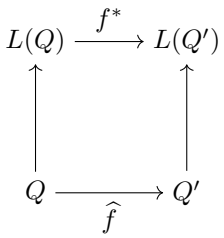
Exercice 17 **Solution page 23**
Prouver le théorème

f peut servir aussi à transformer les automates (non déterministes) sur l'alphabet Σ'.
Si $Q = (\Sigma', S, \Delta, I, T)$ on note $\widehat{f}(Q) = (\Sigma, S, \Delta', I, T)$ où le graphe de Δ'
est défini par $G_{\Delta'} = \{(s, f(x), s')\ ;\ (s, x, s') \in G_\Delta\}$.
C'est l'automate obtenu en changeant les lettres par leur image par f.

Théorème 12

Si f est une fonction de Σ' vers Σ alors $L\big(\widehat{f}(Q)\big) = f^*\big(L(Q)\big)$.
pour toute expression régulière r sur Σ'.

On a donc un transport depuis les automates vers les langages.



Dans ce diagramme les deux chemins de Q à $L(Q')$ donnent le même résultat.

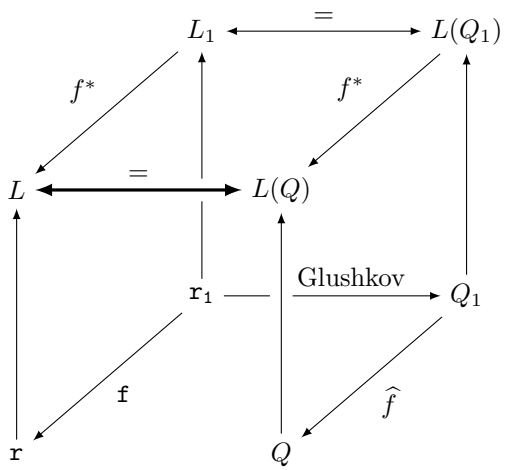
Exercice 18 **Solution page 23**
Prouver le théorème

Construction de l'automate

- On part d'un langage rationnel non vide L.
  dénoté par une expression régulière $\mathtt{r}$, $L = L[\mathtt{r}]$.
- On transforme $\mathtt{r}$ en une expression linéaire $\mathtt{r_1}$ en changeant uniquement les lettres par des lettres **distinctes**.
  Si on note Σ_1 l'alphabet utilisé dans $\mathtt{r_1}$, on note f la fonction de Σ_1 vers Σ qui donne, pour chaque lettre de Σ_1, la lettre de Σ qu'elle remplace.
  Avec la notation ci-dessus on a $r = f(\mathtt{r_1})$.
- On note $L_1 = L[\mathtt{r_1}]$, L_1 est local car $\mathtt{r_1}$ est linéaire.
- On associe à $\mathtt{r_1}$ son automate de Glushkov, Q_1.
- On transforme Q_1 en Q en remplaçant les étiquettes à l'aide de f : $Q = \widehat{f}(Q_1)$.
- On peut, si besoin, déterminiser Q.

Les résultats prouvés ci-dessus donnent alors :

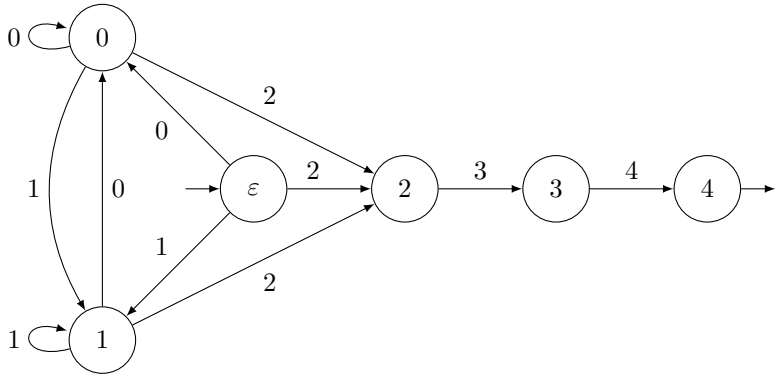
- $L = L[\mathtt{r}] = L[\mathtt{r_1}] = f^*\big(L[\mathtt{r_1}]\big) = f^*\big(L_1\big)$ (transport des expressions régulières)
- $L_1 = L(Q_1)$ (théorème de Glushkov)
- $L(Q) = L\big(\widehat{f}(Q_1)\big) = f^*\big(L(Q_1)\big)$ (transport des automates)
- Ainsi $L = f^*\big(L_1\big) = f^*\big(L(Q_1)\big) = L(Q)$.



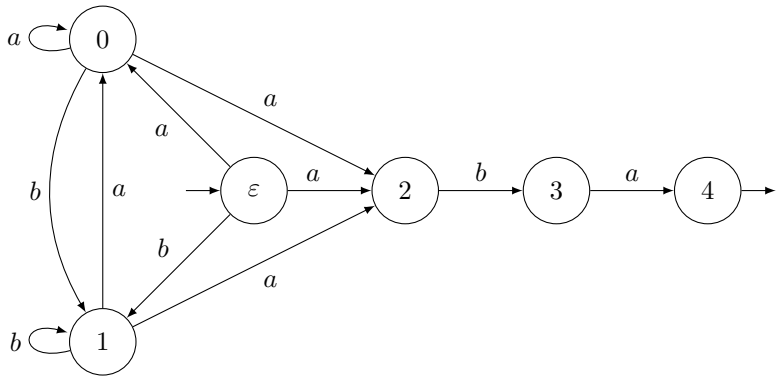
La construction de Q à partir de $\mathtt{r}$ est l'algorithme de Berry-Sethy.

Un exemple

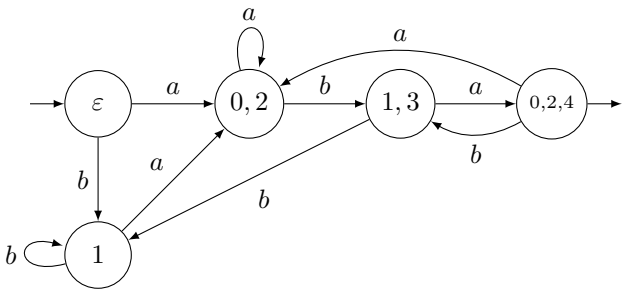
- Soit L dénoté par `r = (a|b)*.a.b.a`.
- On pose $\mathtt{r}_1 = (\mathtt{0}|\mathtt{1}) * .\mathtt{2}.\mathtt{3}.\mathtt{4}$: on définit donc f par le tableau `[|a; b; a; b; a|]`.
- $L_1 = L[\mathtt{r}_1]$ ne content pas ε.
  On calcule $P = \{0, 1, 2\}$, $S = \{4\}$, $F = \{00, 01, 10, 11, 02, 12, 23, 34\}$.
- L'automate Q_1 est calculé.



- On traduit en l'automate Q, non déterministe.



- On peut le déterminiser.



VI Automates et rationalité

VI.1 Lemme de l'étoile

Il est difficile de prouver directement qu'un langage n'est pas reconnaissable ; il faudrait prouver qu'il n'existe aucun automate le reconnaissant.
Un moyen indirect pour nier une propriété est de tirer une conséquence de celle-ci qui est plus facilement contredite ; le lemme de l'étoile fournit une telle propriété.

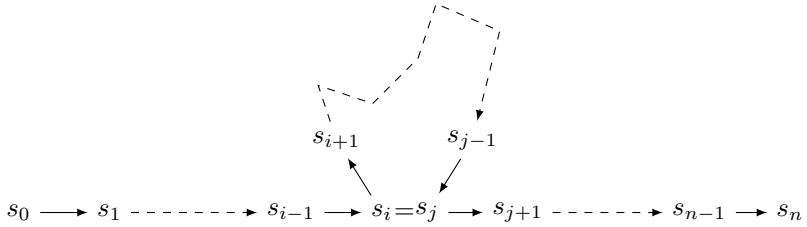
Théorème 13 : lemme de l'étoile ou de pompage (pumping lemma)

Si L est un langage reconnaissable alors il existe un entier N tel que tout mot $u \in L$ de longueur supérieure à N peut se décomposer en $u = u_1 \cdot u_2 \cdot u_3$ avec $|u_1 \cdot u_2| \leqslant N$, $|u_2| \geqslant 1$ et $\forall k \in \mathbb{N},\ u_1 \cdot u_2^k \cdot u_3 \in L$.

Démonstration
On suppose que le langage L est reconnu par l'automate $Q = (\Sigma, S, \Delta, I, T)$.
Le cardinal de S, le nombre d'états, aussi appelé taille de Q, est noté N.
Pour tout mot u de L de longueur $n \geqslant N$ on considère un calcul réussi de u dans Q :

$$s_0 \xrightarrow{x_1} s_1 \xrightarrow{x_2} \cdots \xrightarrow{x_{N-1}} s_{N-1} \xrightarrow{x_N} s_N \xrightarrow{x_{N+1}} \cdots \xrightarrow{x_{n-1}} s_{n-1} \xrightarrow{x_N} s_n$$

L'ensemble de $N+1$ éléments $\{s_0, s_1, \ldots, s_N\}$ est inclus dans S de cardinal N donc il existe deux indices i et j distincts tels que $0 \leqslant i < j \leqslant N$ et $s_i = s_j$.



Si on définit $u_1 = x_1 \cdots x_i$, $u_2 = x_{i+1} \cdots x_j$ et $u_3 = x_{j+1} \cdots x_n$ on a
$u = u_1 \cdot u_2 \cdot u_3$, $|u_1 \cdot u_2| = j \leqslant N$ et $|u_2| = j - i \geqslant 1$.
De plus u_1 est l'étiquette d'un calcul de s_0 à s_i, u_2 est l'étiquette d'un calcul de s_i à s_i et u_3 est l'étiquette d'un calcul de s_i à s_n donc, pour tout $k \in \mathbb{N}$, $u_1 \cdot u_2^k \cdot u_3$ est l'étiquette d'un calcul de s_0 à s_n qui est un calcul réussi : tous les mots $u_1.u_2^k.u_3$ sont reconnus. . . . . ∎

Exemples

- Soit $L_0 = \{a^n b^n \ ;\ n \in \mathbb{N}\}$.
  Si L était reconnaissable on pourrait, pour l'entier N du lemme de l'étoile, décomposer $u = a^N b^N \in L$ en $u_1 \cdot u_2 \cdot u_3$ avec $|u_1 \cdot u_2| \leqslant N$ et $|u_2| \geqslant 1$.
  On en déduit que $u_1 = a^p$, $u_2 = a^q$ avec $q \geqslant 1$ et $u_3 = a^r b^N$ avec $p + q + r = N$.
  On devrait avoir $u_1 \cdot u_2^k \cdot u_3 = a^{N+(k-1)q} b^N \in L$ ce qui est impossible pour $k \neq 1$: L_0 n'est pas reconnaissable.
- On peut en déduire que $L_0' = \{u \in \{a, b\}^* \ ;\ |u|_a = |u|_b\}$ n'est pas reconnaissable.
  En effet son intersection avec $L[\mathtt{a} * \mathtt{b}*]$ est L_0 : si L_0' était reconnaissable, comme $L[\mathtt{a} * \mathtt{b}*]$ est rationnel donc reconnaissable, l'intersection L_0 serait reconnaissable.
- Si L est reconnaissable et s'il existe au moins un mot auquel on peut appliquer la décomposition alors le langage contient $\{u_1 \cdot u_2^n \cdot u_3 \ ;\ n \in \mathbb{N}\}$ avec $u_2 \neq \varepsilon$; tous ces mots sont distincts donc le langage est infini.
- Inversement si le langage L est fini (il est alors rationnel) on ne peut appliquer la décomposition à aucun mot. On en déduit qu'aucun mot n'est de longueur supérieure à N.
  Ainsi un automate qui reconnaît un langage fini a un nombre d'états strictement supérieur à la longueur du plus long mot de L.

VI.2 Théorème de Kleene

Théorème 14 : de Kleene

Les langages rationnels sur Σ sont les langages reconnaissables sur Σ.

Démonstration On a déjà vu que tout langage rationnel est reconnaissable.
Il reste à prouver que tout langage reconnaissable est rationnel.
Si L est reconnaissable, on considère un automate qui peut être non déterministe et comportant des transitions spontanées $Q = (\Sigma, S, \Delta, I, T, \varphi)$ qui reconnaît L.
L'idée de l'algorithme de Mc Naughton et Yamada est de calculer des langages particulier afin de parvenir à L. On note $L(s,t)$ le langage reconnu par l'automate $(\Sigma, S, \Delta, \{s\}, \{t\}, \varphi)$ pour $s \in S$ et $t \in T$; on a $L = \bigcup_{s\in I}\bigcup_{t\in T} = L(s,t)$.
On écrit $S = \{s_0, s_1, \ldots, s_{N-1}\}$, $S_0 = \emptyset$ et $S_k = \{s_0, s_1, \ldots, s_{k-1}\}$ pour $1 \leqslant k \leqslant N$.
$L^k(s,t)$ est l'ensemble des étiquettes des calculs entre s et t ne passant que par des états de S_k en dehors de l'origine et de l'extrémité ; $L(s,t) = L^N(s,t)$.

- $L_0(s,t)$ est le langage des mots d'une lettre qui est l'étiquette d'une transition $s \xrightarrow{x} t$ auquel on ajoute ε si $s = t$ ou s'il existe une transition spontanée entre s et t ($t \in \varphi(s)$).
- On suppose qu'on connaît les langages $L_k(s,t)$.
  Si on a $u \in L^{k+1}(s,t)$, on isole les états s_k dans un calcul de s vers t qui passe par s_k et d'étiquette u : $u = u_1 \cdot u_2 \cdots u_{p-1} \cdot u_p$ où
  — u_1 est l'étiquette d'un calcul de s vers s_k ne passant que par des états de S_k,
  — u_2, …, u_{p-1} sont les étiquette de calculs de s_k vers s_k ne passant que par S_k,
  — u_p est l'étiquette d'un calcul de s_k vers t ne passant que par des états de S_k.
  Inversement un tel produit donne un mot de $L^{k+1}(s,t)$.
  Si un calcul d'étiquette u ne passe pas par s_k alors $u \in L^k(s,t)$.
  Ainsi $L^{k+1}(s,t) = L^k(s,t) \cup L^k(s,s_k) \cdot \left(L^k(s_k,s_k)\right)^* \cdot L^k(s_k,t)$.

Les langages $L_0(s,t)$ sont finis donc rationnels.
On passe des langages $L_k(s,t)$ aux langages $L^{k+1}(s,t)$ par des opérations rationnelles donc, par récurrence, tous les langages $L_k(s,t)$ sont rationnels.
En particulier $L(s,t) = L^N(s,t)$ est rationnel puis $L = \bigcup_{s\in I}\bigcup_{t\in T} = L(s,t)$ est rationnel. . . . . ■

Calcul pratique : élimination des états

On généralise la notion d'automate.

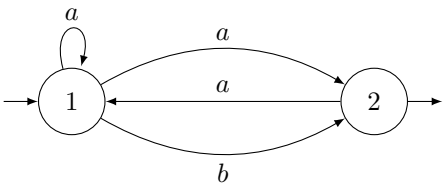
Définition 21

Un *automate généralisé* est un automate fini (déterministe ou non) dont les transitions sont étiquetées par des expressions régulières.

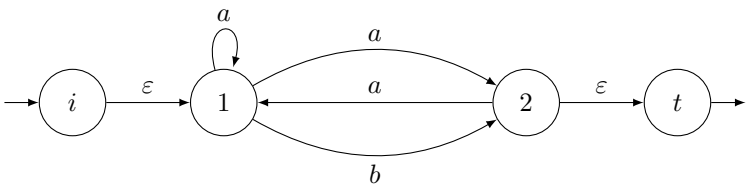
On note qu'un automate généralisé n'est pas un automate classique car les étiquettes appartiennent à un ensemble infini. Pour ces automates les transitions spontanées deviennent de vraies transitions.
Un **calcul** d'un automate généralisé est une suite (finie) de transitions telle que l'origine de chacune est l'extrémité de la précédente.
On le note $s \xrightarrow{\mathtt{r_1}} s_1 \xrightarrow{\mathtt{r_2}} \cdots \xrightarrow{\mathtt{r_{n-1}}} s_{n-1} \xrightarrow{\mathtt{r_n}} s_n$

- L'étiquette du calcul est alors $\mathtt{r_1} \cdot \mathtt{r_2} \cdots \mathtt{r_{n-1}} \cdot \mathtt{r_n}$.
- Un calcul est réussi si son origine est un élément de I et son extrémité un élément de F.
- Le langage reconnu est l'union des étiquettes des calculs reconnus par l'automate.

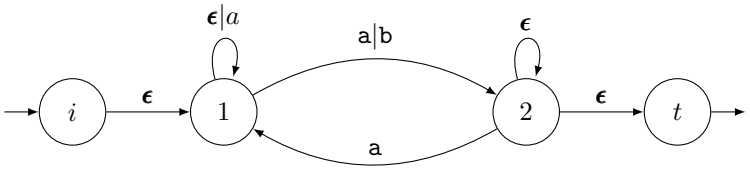
Si L est reconnaissable, on considère un automate qui peut être non déterministe et comportant des transitions spontanées $Q = (\Sigma, S, \Delta, I, T, \varphi)$.
On illustre par l'automate suivant.



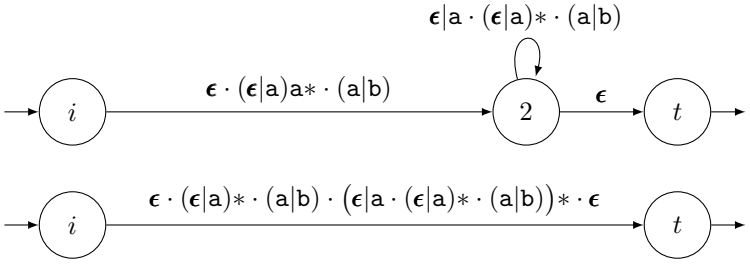
On ajoute deux états, i_0 et t_0 et des transitions spontanées entre i_0 et tous les éléments de I et entre tous les éléments de T et t_0. En remplaçant I par $\{i_0\}$ et T par $\{t_0\}$, on obtient un automate normalisé qui reconnaît le même langage que Q.



On transforme en automate généralisé q_0 en regroupant les étiquettes de mêmes origine et extrémité. Il y a alors au plus une transition entre 2 états, on ajoute une étiquette $\boldsymbol{\epsilon}$ pour les boucles entre un état et lui-même s'il n'y avait pas de transition et une étiquette $\boldsymbol{\varnothing}$ pour les transition entre 2 états entre lesquels il n'y avait pas de transition. Ces transitions ajoutées ne sont pas dessinées.
L'automate reconnaît le même langage que Q.
On note $\mathtt{r}(s, s')$ l'expression régulière de la transition entre s et s' dans Q_0.



On retire un par un les états autres que i et t ; si on retire l'état s_0 on remplace, pour s et s' distincts de s_0, $\mathtt{r}(s, s')$ par $\mathtt{r}(s,t)|\mathtt{r}(s,s_0) \cdot \big(\mathtt{r}(s_0,s_0)\big)* \cdot \mathtt{r}(s_0,s')$.
On admet encore que le langage reconnu est toujours celui que reconnaît Q.



Dans les calculs, on n'a pas écrit les étiquettes $\boldsymbol{\varnothing}$.

Solutions

Exercice 1
La propriété se démontre par récurrence sur $|u|$.

- Elle est triviale pour $|u| = 0$ car alors $u = \varepsilon$ et $\delta^*(s, \varepsilon.v) = delta^*(s, v) = \delta^*\big(\delta^*(s, \varepsilon), v\big)$
- On suppose qu'elle est vraie pour les mots de longueur n.
  Si u est de longueur $n+1$ on écrit $u = x.u'$ avec u' de longueur n.
  $\delta^*(s, u.v) = \delta^*\big(s, x.(u'.v)\big) = \delta^*\big(\delta(s, x), u'.v\big)$; on note $s' = \delta(s, x)$.
  D'après l'hypothèse de récurrence, $\delta^*(s', u'.v) = \delta^*\big(\delta^*(s', u'), v\big)$ or
  $\delta^*(s', u') = \delta^*\big(\delta(s, x), u'\big) = \delta^*(s, x.u') = \delta^*(s, u)$ donc on a bien
  $\delta^*(s, u.v) = \delta^*\big(\delta^*(s', u'), v\big) = \delta^*\big(\delta^*(s, u), v\big)$.
- La propriété est donc vraie pour tout n.

Exercice 2

- Si $T = \emptyset$ alors $\delta^*(s_0, u)$ ne peut jamais appartenir à T donc $L(Q) = \emptyset$.
- Si $T = S$ alors $\delta^*(s_0, u) \in S = T$ pour tout u donc $L(Q) = \Sigma^*$.
- Si $L(Q)$ contient ε alors $s_0 = \delta^*(s_0, \varepsilon) \in T$: s_0 appartient à T.
  Inversement, si $s_0 \in T$ alors $\delta^*(s_0, \varepsilon) = s_0 \in T$ donc $L(Q)$ contient ε.

Exercice 3
Tout élément s de S' est de la forme $s = s_0.v$ donc
$\delta(s, a) = \delta\big(\delta^*(s_0, u)), a\big) = \delta^*\big(\delta^*(s_0, u)), a\big) = \delta^*(s_0, u.a) \in S'$.
Les sommets atteints depuis s_0 appartiennent à S' donc le chemin de $\delta^*(s, u)$ est un chemin dans S', à chaque étape, on peut remplacer δ par δ' d'où $\delta^*(s, u) = \delta''^*(s, u)$.

Exercice 4

- Si $u \in L(Q')$ alors $\delta^*(s_0, u) = \delta'^*(s_0, u) \in T \cap S' \subset T$ donc $u \in L(Q) : L(Q') \subset L(Q)$.
- Si $u \in L(Q)$ alors $\delta^*(s_0, u) \in T$ donc $\delta'^*(s_0, u) \in T$. Or $\delta'^*(s_0, u) \in S'$ donc
  $\delta'^*(s_0, u) \in T \cap S' = T' : u \in L(Q') : L(Q') \subset L(Q)$.

Ainsi $L(Q') = L(Q)$.

Exercice 5
Les mots bloquants deviennent des mots qui arrivent à l'état-puits et qui y restent donc ne sont pas reconnus alors que les mots non-bloquants gardent leur parcours et sont reconnus ou non en même temps dans les deux automates.

Exercice 6
Tout mot reconnu par Q' est reconnu par Q.
De plus tout mot reconnu par Q est non bloquant et ne passe que par des états co-accessibles donc son chemin est dans S', il est donc reconnu par Q'.

Exercice 7
Soit L reconnaissable. On considère un automate $Q = (\Sigma, S, \delta, s_0, T)$ qui le reconnaît. On a a alors $u \in \overline{L} = S \setminus T$ équivalent à $\delta^*(s_0, u) \notin T$, c'est-à-dire à $\delta^*(s_0, u) \in \overline{T} = S \setminus T$.
Ainsi $\overline{L}$ est reconnu par l'automate $(\Sigma, S, s_0, \delta, \overline{T})$.

Exercice 8
Par récurrence sur $|u|$.

Exercice 9

$\delta(E,x) \in \mathcal{P}(S)$ pour tout $E \subset S$ et pour tout $x \in \Sigma$ donc δ est bien une fonction définie sur $\mathcal{P}(S) \times \Sigma$: Q' est bien un automate déterministe complet.

1. Si $u = x_1 x_2 \cdots x_n$ appartient à $L(Q)$ alors il existe un calcul réussi
   $s_0 \xrightarrow{x_1} s_1 \xrightarrow{x_2} s_2 \xrightarrow{x_3} \quad \cdots \quad \xrightarrow{x_{n-1}} s_{n-1} \xrightarrow{x_n} s_n$ avec $s_0 \in I$ et $s_n \in T$.
   On a alors $s_1 \in \Delta(s_0, x_1)$ avec $s_0 \in I$ donc $s_1 \in \delta(I, x_1) = E_1$.
   De même $s_2 \in \Delta(s_0, x_1)$ avec $s_1 \in E_1$ d'où $s_2 \in \delta(E_1, x_2) = E_2$.
   On construit ainsi le parcours de u dans Q' : $I \xrightarrow{x_1} E_1 \xrightarrow{x_2} E_2 \xrightarrow{x_3} \quad \cdots \quad \xrightarrow{x_{n-1}} E_{n-1} \xrightarrow{x_n} E_n$
   Or $s_n \in E_n$ et $s_n \in T$ donc $S_n \cap T \neq \emptyset$ d'où $S_n \in \mathcal{T}$.
   Ainsi $\delta^*(I, u) \in \mathcal{T}$ dans Q' donc $u \in L(Q')$: on a prouvé $L(Q) \subset L(Q')$.
2. Si $\varepsilon \in L(Q)$ alors il existe un état s de S qui est initial et final à la fois. Ainsi $I \cap T \neq \emptyset$. Dans Q' on a donc $I \in \mathcal{T}$ donc $\varepsilon \in L(Q')$.
3. Inversement si $u \neq \varepsilon \in L(Q')$ alors son parcours est de la forme
   $I = E_0 \xrightarrow{x_1} E_1 \xrightarrow{x_2} \cdots \xrightarrow{x_{n-1}} E_{n-1} \xrightarrow{x_n} E_n$ avec $E_n \in \mathcal{T}$ donc il existe $s_n \in E_n \cap T$.
   La transition $E_{n-1} \xrightarrow{x_n} E_n$ implique qu'il existe un élément s_{n-1} dans E_{n-1} et une transition $s_{n-1} \xrightarrow{x_n} s_n$
   On définit ainsi, en descendant, un élément $s_i \in E_i$ pour tout i avec un calcul
   $s_0 \xrightarrow{x_1} s_1 \xrightarrow{x_2} \cdots \xrightarrow{x_{n-1}} s_{n-1} \xrightarrow{x_n} s_n$
   $s_0 \in E_0 = I$, $s_n \in T$ donc u est reconnu par Q.
   On a alors $L(Q') \subset L(Q)$ d'où l'égalité.

Exercice 10

Soit $u \in L(Q)$, $u = u_1 u_2 \cdots u_p$ dans Σ^*, il existe un calcul réussi qu'on écrit, en isolant les ε,
$s_{0,1} \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} s_{0,n_0} \xrightarrow{u_1} s_{1,1} \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} s_{p-1,n_{p-1}} \xrightarrow{u_p} s_{p,1} \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} s_{p,n_p}$ avec $s_{0,1} \in T$ et $s_{p,n_p} \in T$.
On a $s_{0,n_0} \in \kappa(s_{0,1})$ donc $s_{0,n_0} \in I'$ et
$s_{1,1} \in \Delta(s_{0,n_0}, u_1)$ et $s_{1,n_i} \in \kappa(s_{1,1})$ donc $s_{1,n_0} \in \Delta'(s_{0,n_0}, u_1)$.
De même, pour chaque $i \geqslant 2$, $s_{i,n_i} \in \Delta'(s_{i-1,n_{i-1}}, u_i)$ donc on a un calcul réussi pour u dans Q' :
$s_{0,n_0} \xrightarrow{u_1} s_{1,n_1} \xrightarrow{u_2} \cdots s_{p-1,n_{p-1}} \xrightarrow{u_p} s_{p,n_p}$: $u \in L(Q')$.

Inversement, si $u \in L(Q')$, on considère un calcul réussi pour u dans Q' :
$s'_0 \xrightarrow{u_1} s'_1 \xrightarrow{u_2} \cdots s'_{p-1} \xrightarrow{u_p} s'_p$ avec $s'_0 \in I'$ et $s'_p \in T$.
On a $s'_0 \in I' = \kappa(I)$ donc il existe un calcul dans Q d'étiquette ε entre un sommet $s_0 \in I$ et s'_0.
Pour $1 \leqslant i \leqslant p$, on a $s'_{i-1} \xrightarrow{u_i} s'_i$ donc il existe un état s_i tel que $s_i \in \Delta(s'_{i-1}, u_i)$ et $s'_i \in \kappa(s_i)$ donc il existe un calcul dans Q d'étiquette u_i entre s'_{i-1} et s'_i.
En combinant ces calculs, on obtient un calcul dans Q d'étiquette $u_1 u_2 \cdots u_p = u$ entre $s_0 \in I$ et $s'_p \in T$, c'est donc un calcul réussi et $u \in L(Q)$.
On a bien égalité des langages..

Exercice 11

Les mots non vides de L, $u = x_1 x_2 \cdots x_n$, sont associés aux mots reconnus par Q selon le chemin
$\varepsilon \xrightarrow{x_1} x_1 \xrightarrow{x_2} x_2 \xrightarrow{x_3} \quad \cdots \quad \xrightarrow{x_{n-1}} x_{n-1} \xrightarrow{x_n} x_n$.
Inversement un tel chemin a une étiquette u dans L car la transition $\varepsilon \xrightarrow{x_1} x_1$ implique $x_1 \in P$, les $x_{i-1} \xrightarrow{x_i} x_i \xrightarrow{x_{i+1}} x_{i+1}$ impliquent $x_i x_{i+1} \in F$ et x_n terminal implique $x_n \in T$.
Le mot vide est reconnu si et seulement ε est un état final.

Exercice 15

On montre par récurrence sur n que, pour tout mot $u \in \Sigma'^*$ et pour tout mot $v \in \Sigma'^*$ de longueur n, $f^*(u.v) = f^*(u).f^*(v)$.

Exercice 16

1. On a $L_1 \subset L_1 \cup L_2$ donc $f^*(L_1) \subset f^*(L_1 \cup L_2)$.
   De même $f^*(L_2) \subset f^*(L_1 \cup L_2)$ donc $f^*(L_1) \cup f^*(L_2) \subset f^*(L_1 \cup L_2)$.
   Inversement si $v \in f^*(L_1 \cup L_2)$ alors il existe $u \in L_1 \cup L_2$ tel que $v = f^*(u)$. Si $u \in L_i$ alors $v \in f^*(L_i) \subset f^*(L_1) \cup f^*(L_2)$ d'où l'égalité.
2. $f^*(L_1.L_2) = \{f^*(u.v)\ ;\ u \in L_1, v \in L_2\} = \{f^*(u).f^*(v)\ ;\ u \in L_1, v \in L_2\}$ puis
   $f^*(L_1.L_2) = \{f^*(u)\ ;\ u \in L_1\}.\{f^*(v)\ ; v \in L_2\} = f^*(L_1).f^*(L_2)$.
3. On en déduit par récurrence sur n que $f^*(L_1^n) = \big(f^*(L_1)\big)^n$ puis
   $$f\left(L_1^*\right) = f\left(\bigcup_{n\in\mathbb{N}} L_1^n\right) = \bigcup_{n\in\mathbb{N}} f\left(L_1^n\right) = \bigcup_{n\in\mathbb{N}} \big(f^*(L_1)\big)^n = \big(f^*(L_1)\big)^*.$$
4. On a $L_1 \cap L_2 \subset L_1$ donne $f^*(L_1 \cap L_2) \subset f^*(L_1)$ et, de même, $f^*(L_1 \cap L_2) \subset f^*(L_2)$ donc $f^*(L_1 \cap L_2) \subset f^*(L_1) \cap f^*(L_2)$.
   Cependant l'inclusion peut être stricte. Pour $\Sigma' = \Sigma = \{a, b\}$, $f(a) = f(b) = a$, $L_1 = \{a\}^*$ et $L_2 = \{b\}^*$ on a $L_1 \cap L_2 = \{\varepsilon\}$ mais $f^*(L_1) = f^*(L_2) = \{a\}^*$ donc
   $f^*(L_1 \cap L_2) = \{\varepsilon\} \neq \{a\}^* = f^*(L_1) \cap f^*(L_2)$.

Exercice 17
La démonstration se fait par induction structurelle.

1. Les cas de base sont immédiats.
2. Pour $r = x$, $L\big[\widehat{\varphi}(x)\big] = L\big[\varphi(x)\big] = \{\varphi(x)\} = \varphi^*\big(\{x\}\big) = \varphi^*\big(L[x]\big)$.
3. On suppose que $L\big[\widehat{\varphi}(e)\big] = \varphi^*\big(L[e]\big)$ et $L\big[\widehat{\varphi}(f)\big] = \varphi^*\big(L[f]\big)$.
   $L\big[\widehat{\varphi}(e+f)\big] = L\big[\widehat{\varphi}(e)+\widehat{\varphi}(f)\big] = L\big[\widehat{\varphi}(e)\big] \cup L\big[\widehat{\varphi}(f)\big] = \varphi^*\big(L[e] \cup \varphi^*\big(L[f]\big) = \varphi^*\big(L[e] \cup L_[f]\big) = \varphi^*\big(L[(e+f)]\big)$
   De même pour le produit et l'étoile.

Exercice 18
Tout mot de $L(Q)$ est l'étiquette d'un calcul réussi dans Q ; on note $u = x_1x_2\cdots x_n$,
$s_0 \xrightarrow{x_1} s_1 \xrightarrow{x_2} s_2 \xrightarrow{x_3} \quad \cdots \quad \xrightarrow{x_{n-1}} s_{n-1} \xrightarrow{x_n} x_n$.
Dans $\widehat{f}(Q)$ le chemin devient
$s_0 \xrightarrow{f(x_1)} s_1 \xrightarrow{f(x_2)} s_2 \xrightarrow{f(x_3)} \quad \cdots \quad \xrightarrow{f(x_{n-1})} s_{n-1} \xrightarrow{f(x_n)} x_n$
il reste réussi car les ensembles initiaux et terminaux sont inchangés ;
ainsi $f(x_1)f(x_2)\cdots f(x_n) = f^*(u)$ appartient à $L\big(\widehat{f}(Q)\big)$ d'où $f^*\big(L(Q)\big) \subset L\big(\widehat{f}(Q)\big)$.

De même tout calcul réussi dans $\widehat{f}(Q)$ peut se "remonter" en un calcul réussi dans Q car les étiquettes sont des $f(x_i)$. Le mot associé est donc l'image par f^* d'un mot de $L(Q)$ d'où $L\big(\widehat{f}(Q)\big) \subset f^*\big(L(Q)\big)$
On a ainsi $L\big(\widehat{f}(Q)\big) = f^*\big(L(Q)\big)$.