LASF – Opérations sur les automates finis

Christophe Mouilleron

ensiie
école nationale supérieure d'informatique
pour l'industrie et l'entreprise

Motivation

La dernière fois, on a vu :

- les opérations sur les mots et les langages
- la définition d'automate fini

Objectif :

- définir des opérations sur les automates finis
- faire le lien avec les opérations sur les langages
- en déduire des procédés algorithmiques pour reconnaitre facilement certains langages

Plan

Plan

Automate transposé

Idée :

1. changer le sens des transitions
2. échanger I et F

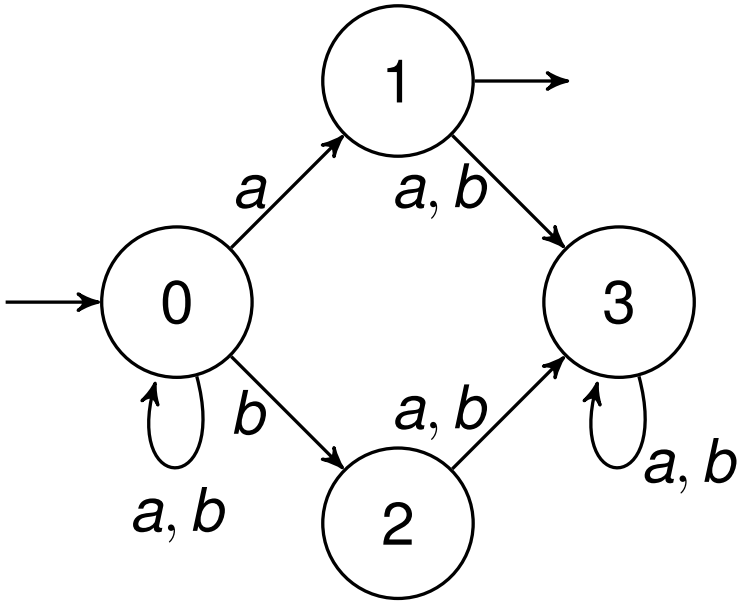
Définition

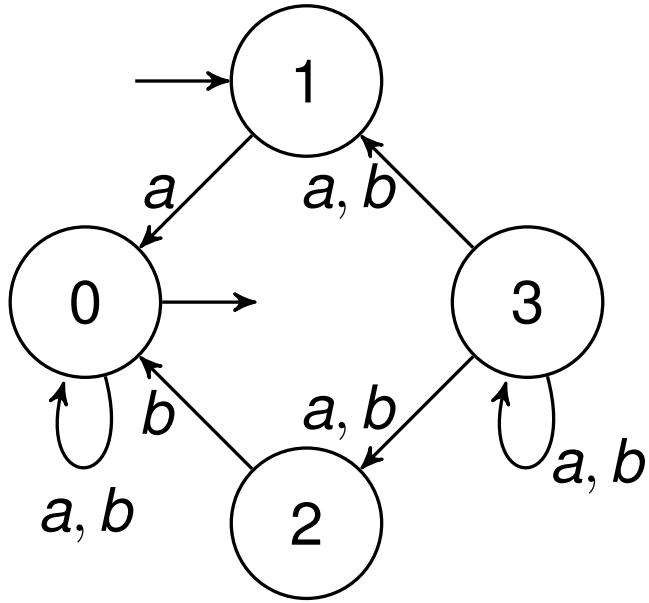
Soit $\mathcal{A} = (A, Q, I, F, E)$. L'automate transposé de $\mathcal{A}$, noté $\mathcal{A}^t$, est défini par $\mathcal{A}^t := (A, Q, I^t, F^t, E^t)$ où :

- $I^t := F$
- $F^t := I$
- $E^t := \{(q, a, p) \in Q \times A \times Q,\ (p, a, q) \in E\}$

Langage reconnu par l'automate transposé

Exemple : $\mathcal{A} \rightsquigarrow \mathcal{A}^t$

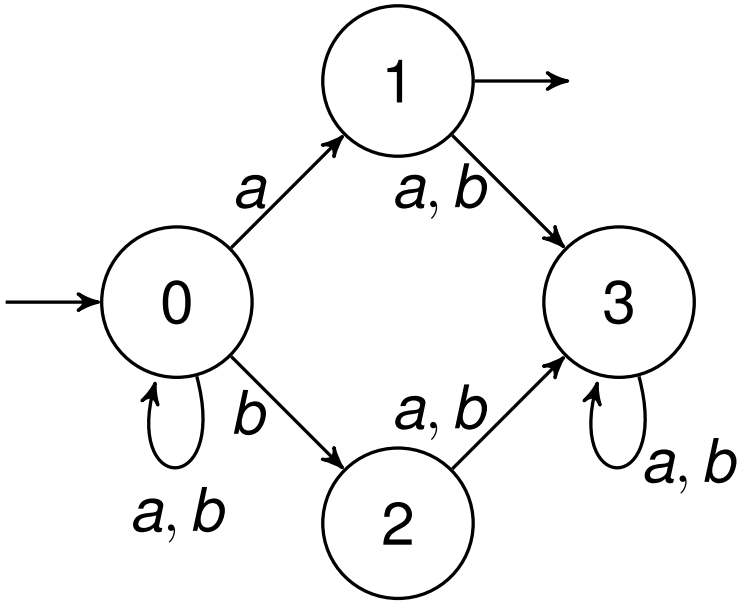


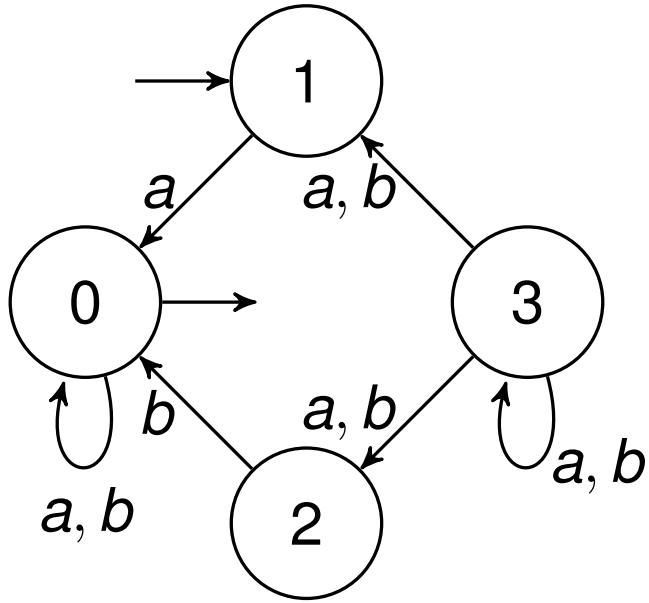


$$\mathrm{L}(\mathcal{A}) = \{ua,\ u \in A^*\} \qquad \mathrm{L}(\mathcal{A}^t) = ?$$

Langage reconnu par l'automate transposé

Exemple : $\mathcal{A} \rightsquigarrow \mathcal{A}^t$





Propriété

Le langage reconnu par $\mathcal{A}^t$ est le transposé du langage reconnu par $\mathcal{A}$.
Autrement dit : $\mathrm{L}(\mathcal{A}^t) = \mathrm{L}(\mathcal{A})^t$.

Plan

Union de deux automates

On se donne deux automates :

- $\mathcal{A}_1 := (A_1, Q_1, I_1, F_1, E_1)$
- $\mathcal{A}_2 := (A_2, Q_2, I_2, F_2, E_2)$

Définition

Si $Q_1 \cap Q_2 = \emptyset$, on définit l'union des automates $\mathcal{A}_1$ et $\mathcal{A}_2$ par

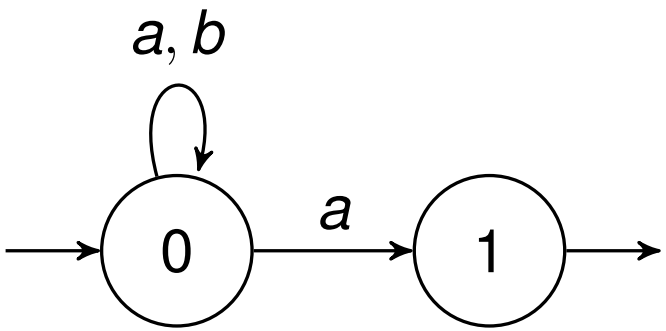
$$\mathcal{A}_1 \cup \mathcal{A}_2 := (A_1 \cup A_2, Q_1 \cup Q_2, I_1 \cup I_2, F_1 \cup F_2, E_1 \cup E_2).$$

Note : Si $Q_1 \cap Q_2 \neq \emptyset$, renuméroter les états de Q_2 qui sont dans Q_1.

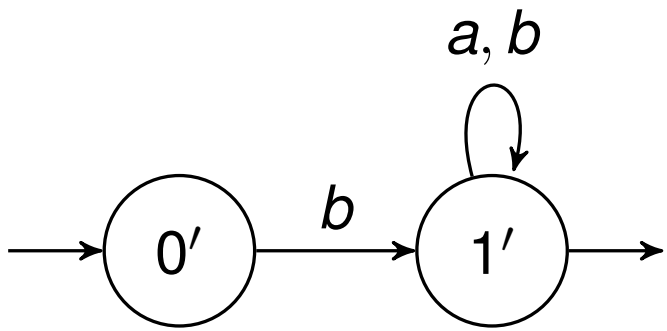
Langage reconnu par l'union d'automates finis

Exemple :

$\mathcal{A}_1$



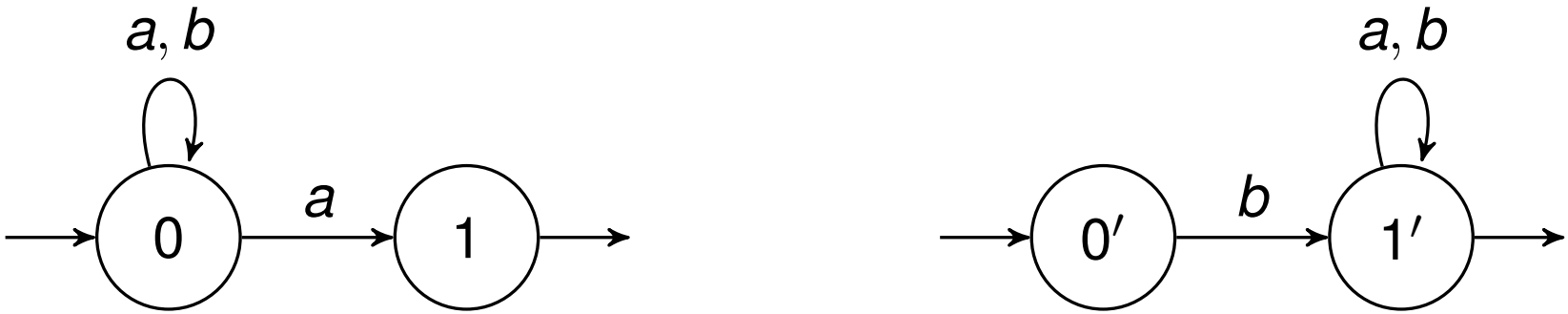
$\mathcal{A}_2$



Langage reconnu par l'union d'automates finis

Exemple :

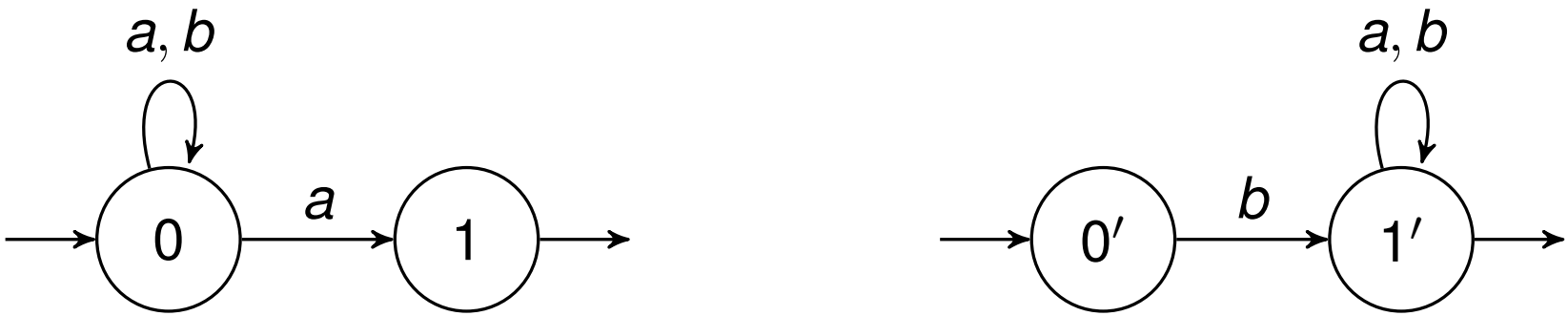
$\mathcal{A}_1 \cup \mathcal{A}_2$



Langage reconnu par l'union d'automates finis

Exemple :

$\mathcal{A}_1 \cup \mathcal{A}_2$



Propriété

Le langage reconnu par $\mathcal{A}_1 \cup \mathcal{A}_2$ est l'union des langages reconnus par $\mathcal{A}_1$ et $\mathcal{A}_2$. Autrement dit : $L(\mathcal{A}_1 \cup \mathcal{A}_2) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

Corollaire

Tout langage fini de A^* est reconnaissable par un automate fini.

Plan

Intersection de deux langages reconnaissables

On se donne toujours deux automates :

- $\mathcal{A}_1 := (A_1, Q_1, I_1, F_1, E_1)$ $\qquad L_1 := \mathrm{L}(\mathcal{A}_1)$
- $\mathcal{A}_2 := (A_2, Q_2, I_2, F_2, E_2)$ $\qquad L_2 := \mathrm{L}(\mathcal{A}_2)$

Comment construire un automate reconnaissant $L_1 \cap L_2$?

Intersection de deux langages reconnaissables

On se donne toujours deux automates :

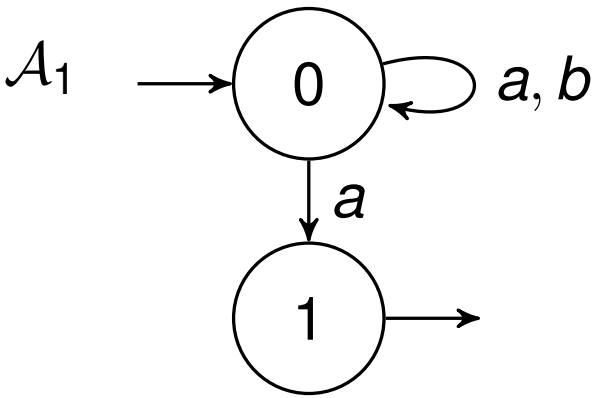
- $\mathcal{A}_1 := (A_1, Q_1, I_1, F_1, E_1)$ $\qquad L_1 := \mathsf{L}(\mathcal{A}_1)$
- $\mathcal{A}_2 := (A_2, Q_2, I_2, F_2, E_2)$ $\qquad L_2 := \mathsf{L}(\mathcal{A}_2)$

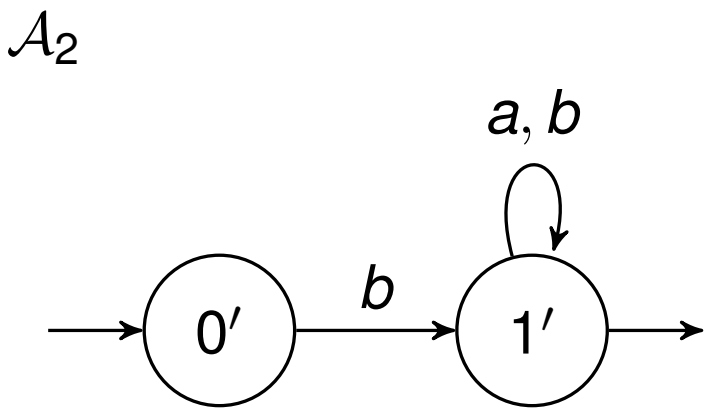
Comment construire un automate reconnaissant $L_1 \cap L_2$?

Idée :

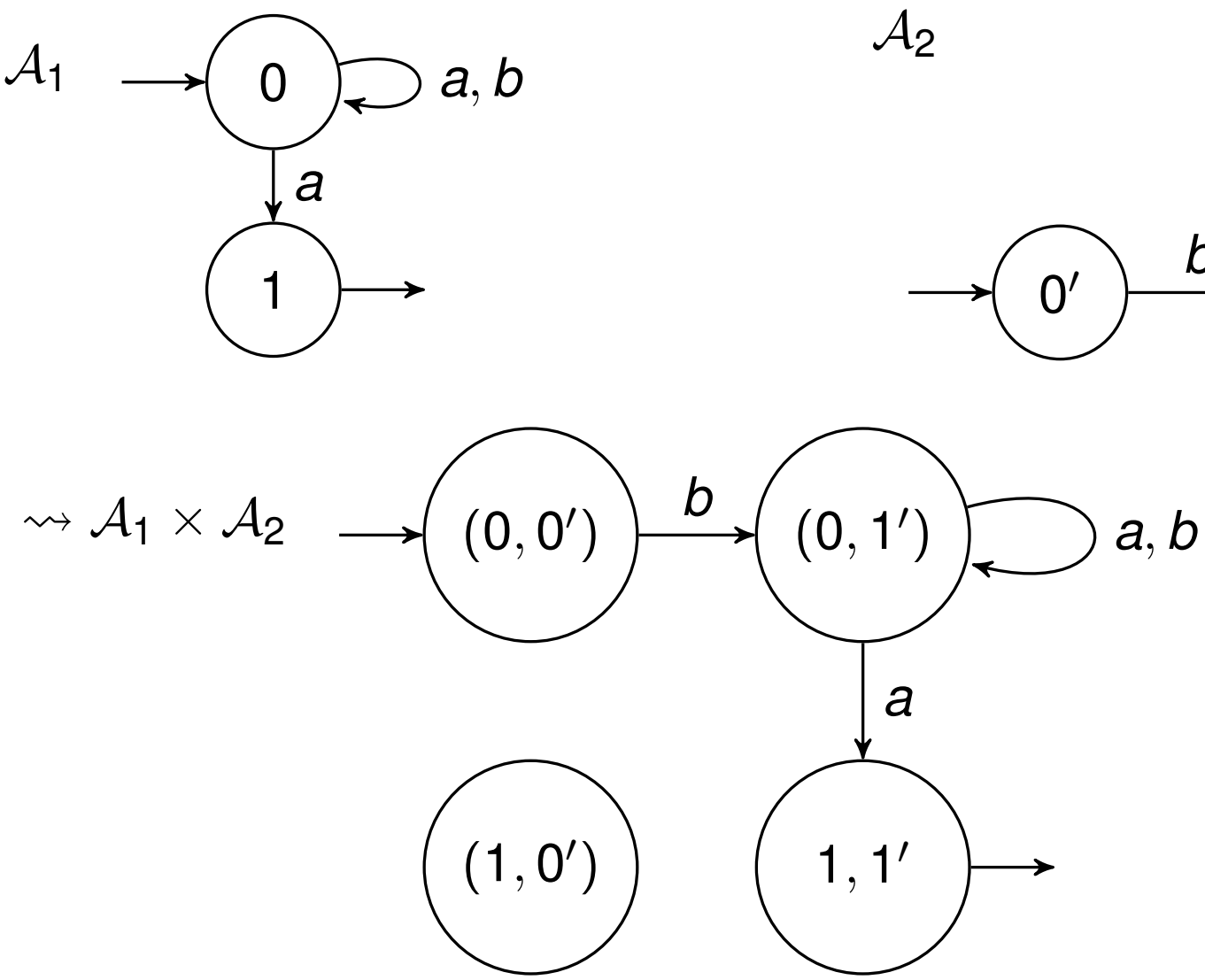
- considérer simultanément un chemin dans $\mathcal{A}_1$ et un dans $\mathcal{A}_2$
- mot u reconnu si et seulement si il existe deux chemins tels que $i_1 \in I_1 \xrightarrow{u} f_1 \in F_1$ **et** $i_2 \in I_2 \xrightarrow{u} f_2 \in F_2$.

Produit cartésien de deux automates – Exemple





Produit cartésien de deux automates – Exemple



Produit cartésien de deux automates

Définition

Le produit cartésien des automates $\mathcal{A}_1$ et $\mathcal{A}_2$ est l'automate $\mathcal{A}_1 \times \mathcal{A}_2 := (A, Q_1 \times Q_2, I_1 \times I_2, F_1 \times F_2, E)$ où :

$$E := \left\{ \Big((p_1, p_2),\ a,\ (q_1, q_2) \Big),\ (p_1, a, q_1) \in E_1 \text{ et } (p_2, a, q_2) \in E_2 \right\}.$$

Propriété

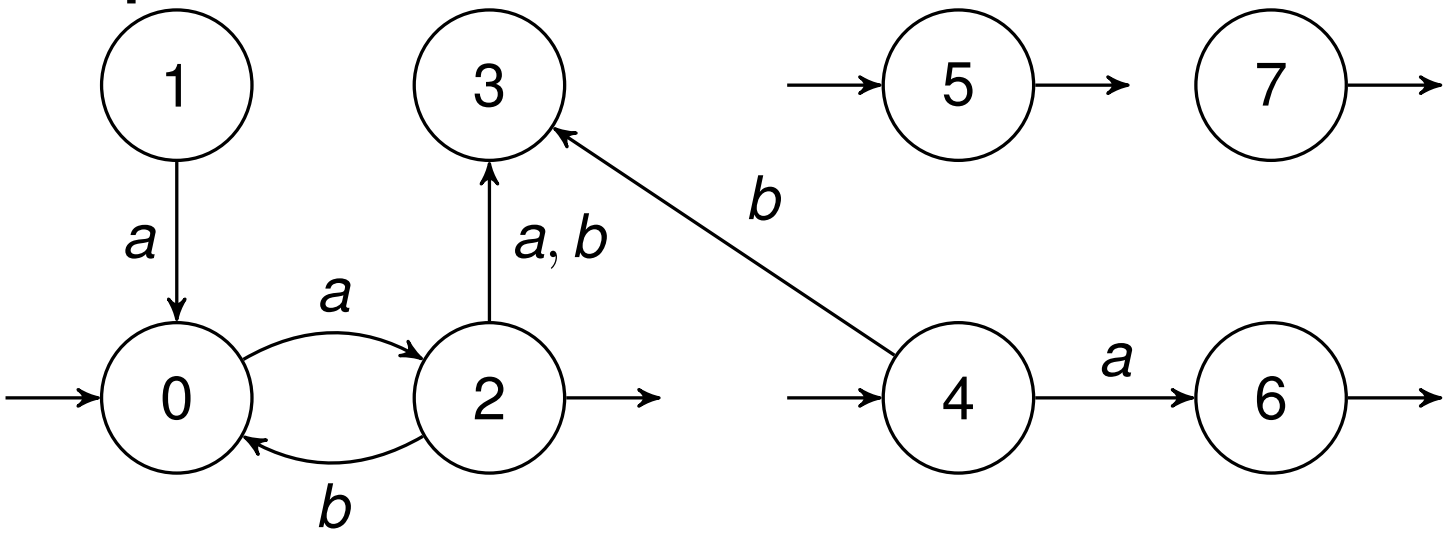
Le langage reconnu par $\mathcal{A}_1 \times \mathcal{A}_2$ est l'intersection des langages reconnus par $\mathcal{A}_1$ et $\mathcal{A}_2$. Autrement dit : $L(\mathcal{A}_1 \times \mathcal{A}_2) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

Plan

Motivation

Idée : supprimer les états inutiles.

Exemple :

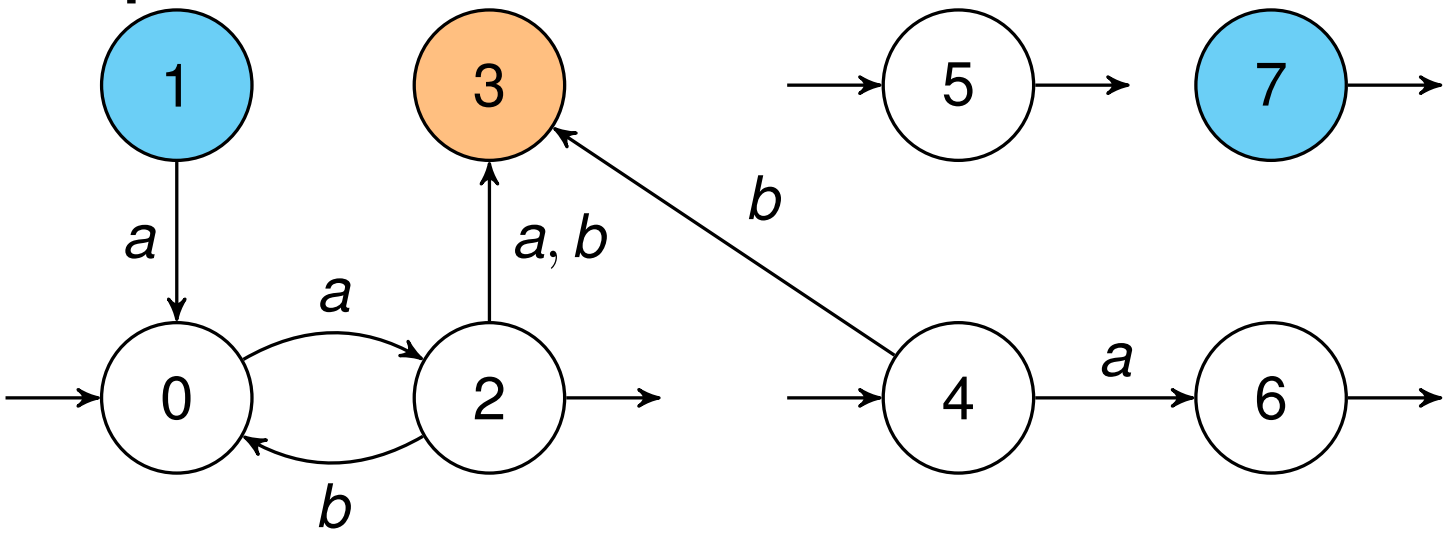


Sont inutiles :

Motivation

Idée : supprimer les états inutiles.

Exemple :



Sont inutiles :

1. les états qu'on n'atteint pas en partant d'un état initial,
2. les états qui ne permettent pas d'atteindre un état final.

États accessibles et co-accessibles

Soit $\mathcal{A} = (A, Q, I, F, E)$ un automate fini.

Définition

Un état q est dit accessible depuis un état p lorsqu'il existe dans $\mathcal{A}$ un chemin $p = s_0 \xrightarrow{u_0} s_1 \xrightarrow{u_1} \dots \xrightarrow{u_n} s_n = q$.

Définition

Un état p est dit accessible lorsqu'il existe dans l'automate $\mathcal{A}$ un chemin $s_0 \xrightarrow{u_0} s_1 \xrightarrow{u_1} \dots \xrightarrow{u_n} s_n$ où $s_0 \in I$ et $s_n = p$.

Définition

Un état p est dit co-accessible lorsqu'il existe dans l'automate $\mathcal{A}$ un chemin $s_0 \xrightarrow{u_0} s_1 \xrightarrow{u_1} \dots \xrightarrow{u_n} s_n$ où $s_0 = p$ et $s_n \in F$.

Algorithme d'accessibilité

Algorithme 1 : `accessibilite`

Entrée : un automate $\mathcal{A} = (A, Q, I, F, E)$, et un état $p \in Q$
Sortie : $S =$ ensemble des états accessibles depuis p dans $\mathcal{A}$

```
1  S ← ∅
2  W ← {p}
3  tant que W ≠ ∅ faire
4      Choisir r dans W
5      W ← W \ {r}
6      S ← S ∪ {r}
7      pour chaque (r, a, q) ∈ E faire
8          si q ∉ S alors
9              W ← W ∪ {q}
10 retourner S
```

Algorithme d'accessibilité (suite)

Exemples avec l'automate précédent :

- `accessibilite`$(1) \rightsquigarrow \{0, 1, 2, 3\}$
- `accessibilite`$(7) \rightsquigarrow \{7\}$

Terminaison de l'algorithme :

Correction de l'algorithme :

Algorithme d'accessibilité (suite)

Exemples avec l'automate précédent :

- $\texttt{accessibilite}(1) \rightsquigarrow \{0, 1, 2, 3\}$
- $\texttt{accessibilite}(7) \rightsquigarrow \{7\}$

Terminaison de l'algorithme :

- Les états vont uniquement de Q à W et de W à S.
- À chaque itération de la boucle, un élément va de W à S.

Correction de l'algorithme :

- Il faut montrer que $q \in \texttt{accessibilite}(p)$ si et seulement si il existe un chemin menant de p à q dans $\mathcal{A}$

preuve par récurrence sur la longueur du chemin

Adaptation et applications

Adaptations possibles :

- calcul de co-accessibilité
- recherche de boucles

chemin $p \xrightarrow{u} p$ avec $u \neq \varepsilon$

Adaptation et applications

Adaptations possibles :

- calcul de co-accessibilité
  - $\rightsquigarrow$ suivre les transitions dans l'autre sens

- recherche de boucles — chemin $p \xrightarrow{u} p$ avec $u \neq \varepsilon$
  - $\rightsquigarrow$ partir avec $W = \{q \mid (p, a, q) \in E$ pour au moins une lettre $a\}$.

Applications :

1. tester si $L(\mathcal{A})$ est non-vide

2. tester si $L(\mathcal{A})$ est infini

Adaptation et applications

Adaptations possibles :

- calcul de co-accessibilité
  - $\rightsquigarrow$ suivre les transitions dans l'autre sens

- recherche de boucles — chemin $p \xrightarrow{u} p$ avec $u \neq \varepsilon$
  - $\rightsquigarrow$ partir avec $W = \{q \mid (p, a, q) \in E$ pour au moins une lettre $a\}$.

Applications :

1. tester si $L(\mathcal{A})$ est non-vide
   - $\rightsquigarrow$ chercher un état final accessible

2. tester si $L(\mathcal{A})$ est infini
   - $\rightsquigarrow$ chercher un état accessible, co-accessible, et sur lequel on boucle.

Plan

Émondé d'un automate fini

Automate émondé : tous les états sont accessibles et co-accessibles.

Procédé d'émondage :

1. calculer l'ensemble Q_1 des états accessibles,
2. calculer l'ensemble Q_2 des états co-accessibles,
3. changer Q en $Q_1 \cap Q_2$, I en $I \cap Q_2$ et F en $F \cap Q_1$
4. supprimer de E toutes les transitions (p, a, q) telles que $p \notin Q_1 \cap Q_2$ ou $q \notin Q_1 \cap Q_2$.

Propriété

Pour tout automate fini $\mathcal{A}$, on peut construire un automate fini émondé $\mathcal{A}^E$ qui est équivalent à $\mathcal{A}$, c'est à dire tel que $L(\mathcal{A}^E) = L(\mathcal{A})$.

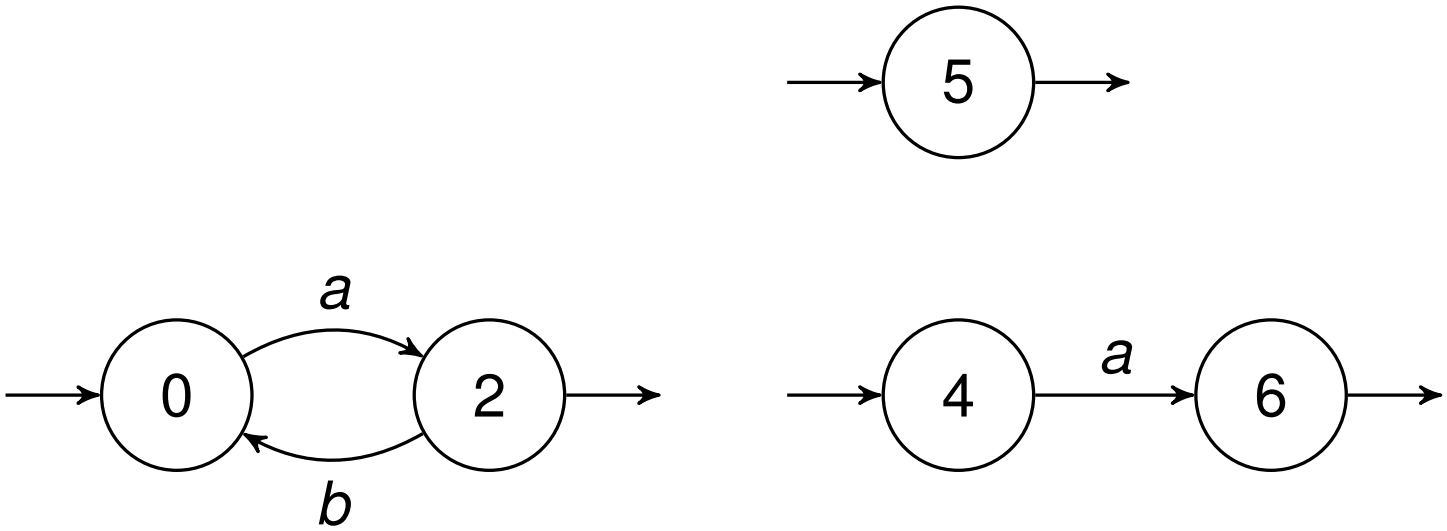
Émondé d'un automate fini – exemple

Sur l'exemple du début :

- états accessibles ? $Q_1 := \{0, 2, 3, 4, 5, 6\}$
- états co-accessibles ? $Q_2 := \{0, 1, 2, 4, 5, 6, 7\}$

Ainsi :

- $Q^E = Q_1 \cap Q_2 = \{0, 2, 4, 5, 6\}$,
- $I^E = I \cap Q_2 = \{0, 4, 5\}$,
- $F^E = F \cap Q_1 = \{2, 5, 6\}$.



Plan

Préfixe, suffixe et facteur d'un langage reconnaissable

Propriété

Si L est un langage reconnaissable, alors Pré(L), Suf(L) et Fact(L) le sont aussi.

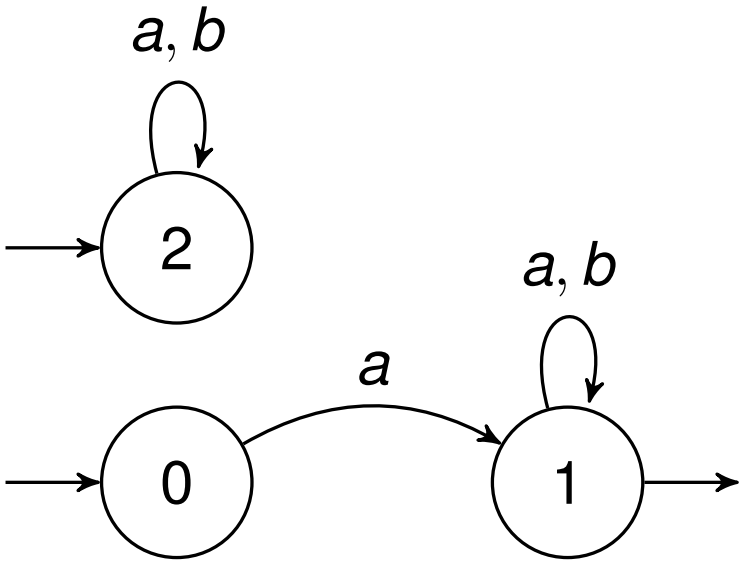
Idée de preuve :

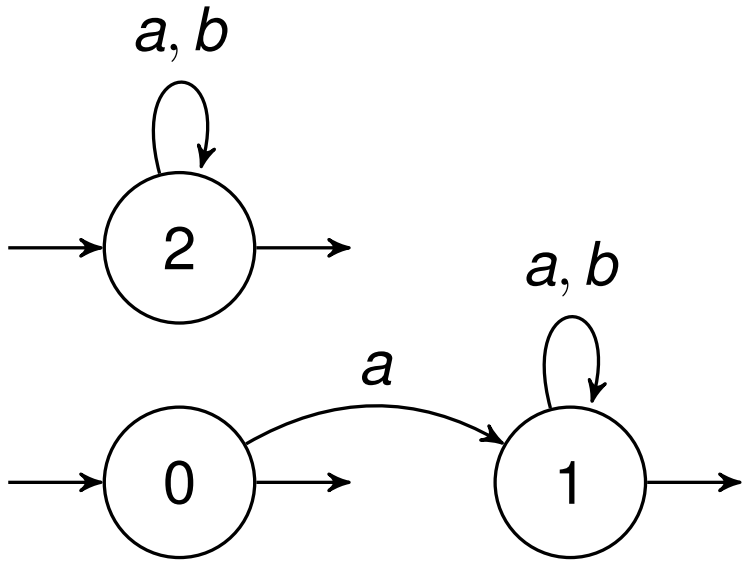
- L reconnaissable $\Rightarrow$ il existe $\mathcal{A}$ tel que $L(\mathcal{A}) = L$
- émonder $\mathcal{A} \Rightarrow \mathcal{A}^E = (A, Q^E, I^E, F^E, E^E)$
- ajuster les entrées/sorties suivant le cas :
  - pour Pré(L), changer F^E en Q^E — on peut s'arrêter n'importe quand
  - pour Suf(L), changer I^E en Q^E — on peut partir d'où on veut
  - pour Fact(L), changer I^E et F^E en Q^E.

Émonder est essentiel !

Attention : Pour reconnaître Pré(L), Suf(L) et Fact(L), il **faut** émonder.

Contre-exemple :





$L := L(\mathcal{A}) = \{au, u \in A^*\}$

$L(\mathcal{A}') = A^* \neq L \cup \{\varepsilon\} = \text{Pré}(L)$

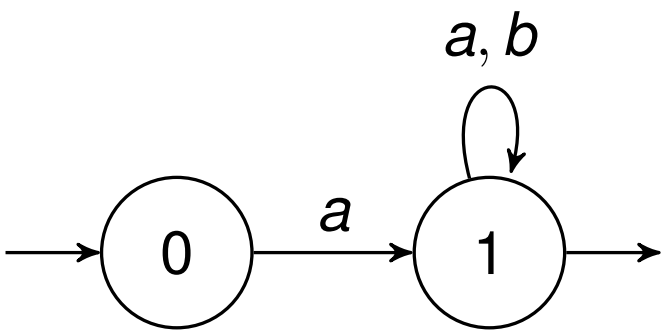
Plan

Automates complets

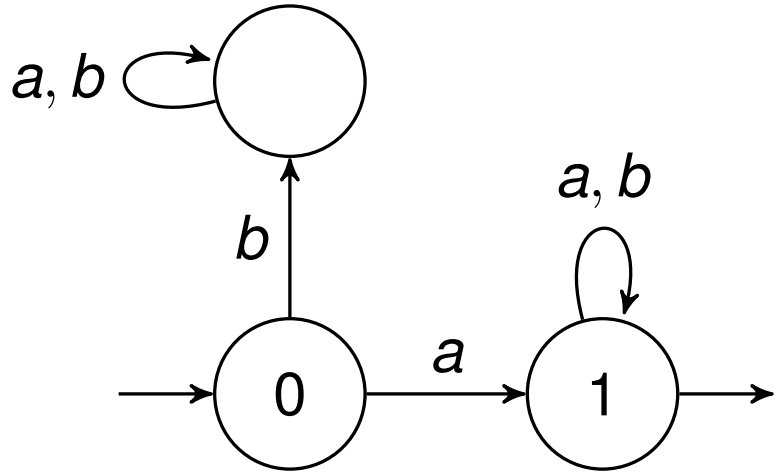
Idée : quelque soit l'état courant et la lettre lu, on doit pouvoir faire une transition vers un nouvel état.

Définition

Un automate fini $\mathcal{A} = (A, Q, I, F, E)$ est dit complet lorsque, pour tout $a \in A$ et tout $p \in Q$, il existe $q \in Q$ tel que $(p, a, q) \in E$.



Pas complet



Complet

Complétion

État puits : état non final sur lequel on boucle quelque soit la lettre lue.

Procédé de complétion :

À partir de $\mathcal{A} = (A, Q, I, F, E)$, construire $\mathcal{A}' := (A, Q \cup \{z\}, I, F, E)$ où

- $z \notin Q$ est un état puits,
- $E' := E \cup \{(p, a, z) \mid \nexists\, q \in Q,\ (p, a, q) \in E\} \cup \{(z, a, z) \mid a \in A\}$.

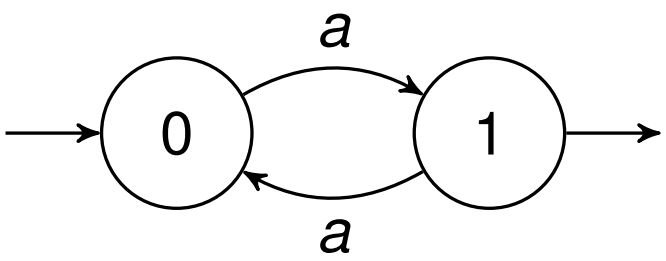
Propriété

Tout automate fini est équivalent à un automate fini accessible complet.

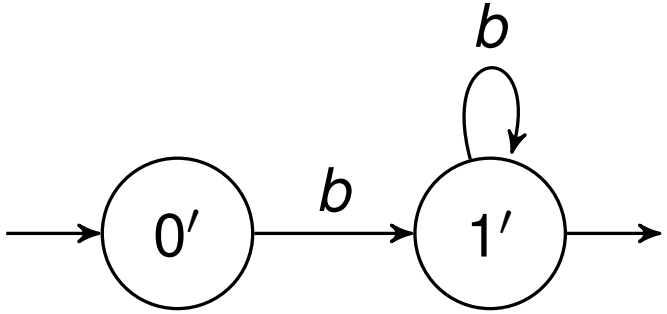
NB : pas émondé à cause de l'état puits. z pas co-accessible

Plan

Reconnaissance de $L_1 \cdot L_2$?



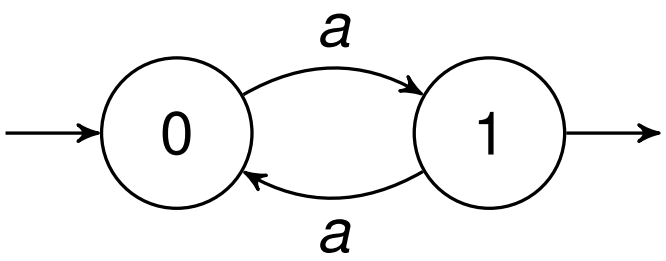
$L_1 := L(\mathcal{A}_1) = \{a^{2n+1}, n \in \mathbb{N}\}$

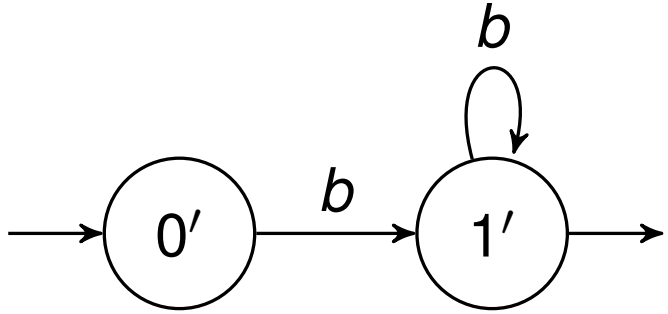


$L_2 := L(\mathcal{A}_2) = \{b^m, m \in \mathbb{N}^*\}$

Automate reconnaissant $L_1 \cdot L_2$?

Reconnaissance de $L_1 \cdot L_2$?

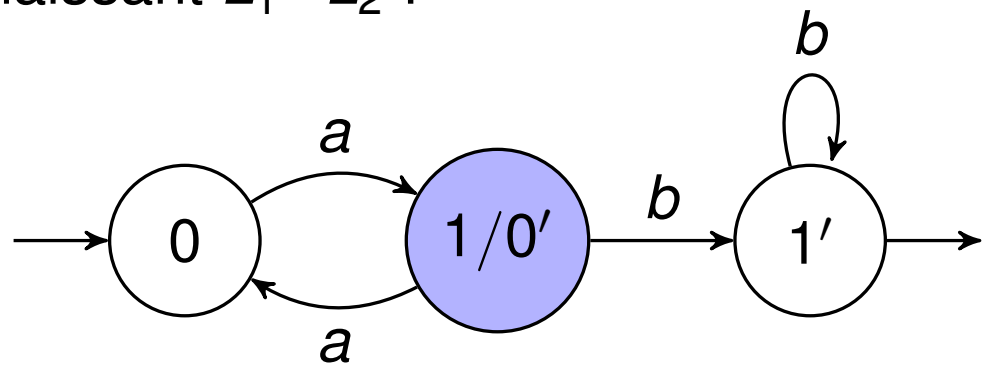




$L_1 := L(\mathcal{A}_1) = \{a^{2n+1}, n \in \mathbb{N}\}$

$L_2 := L(\mathcal{A}_2) = \{b^m, m \in \mathbb{N}^*\}$

Automate reconnaissant $L_1 \cdot L_2$?



Reconnaissance de $L_1 \cdot L_2$? (suite)

Idée : fusionner l'état final de $\mathcal{A}_1$ avec l'état initial de $\mathcal{A}_2$

Problèmes :

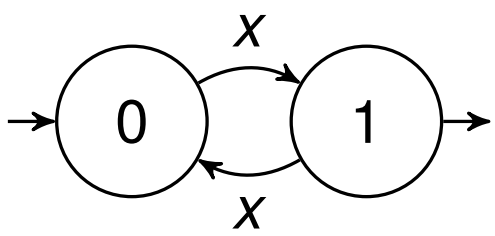
- si $|F_1| > 1$ ou $|I_2| > 1$?

Reconnaissance de $L_1 \cdot L_2$? (suite)

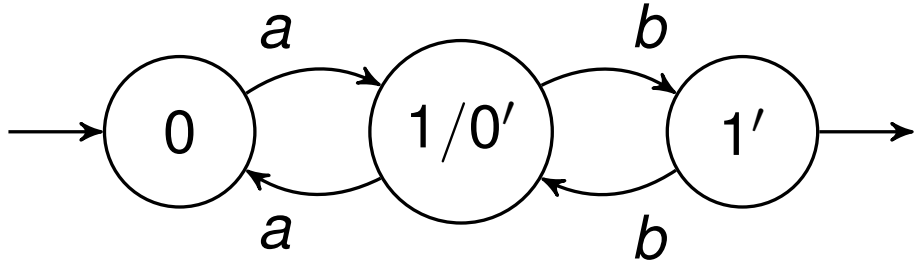
Idée : fusionner l'état final de $\mathcal{A}_1$ avec l'état initial de $\mathcal{A}_2$

Problèmes :

- si $|F_1| > 1$ ou $|I_2| > 1$? se ramener à $|I_2| = 1$
- même si $|I_2| = 1$, cela peut échouer :



donne



$L_1 = \{a^{2n+1}, n \in \mathbb{N}\}$
$L_2 = \{b^{2m+1}, m \in \mathbb{N}\}$

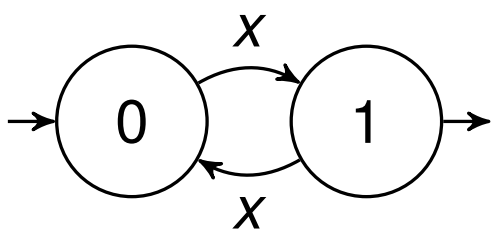
$abbaab \in L(\mathcal{A})$
mais $abbaab \notin L_1 \cdot L_2$

Reconnaissance de $L_1 \cdot L_2$? (suite)

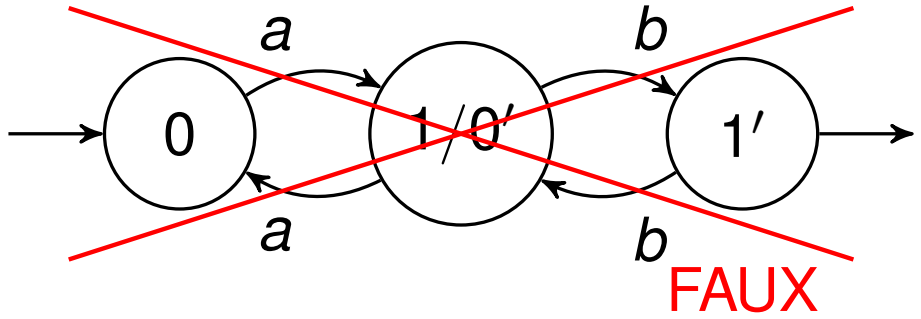
Idée : fusionner l'état final de $\mathcal{A}_1$ avec l'état initial de $\mathcal{A}_2$

Problèmes :

- si $|F_1| > 1$ ou $|I_2| > 1$? se ramener à $|I_2| = 1$
- même si $|I_2| = 1$, cela peut échouer :



donne



$L_1 = \{a^{2n+1}, n \in \mathbb{N}\}$
$L_2 = \{b^{2m+1}, m \in \mathbb{N}\}$

$abbaab \in L(\mathcal{A})$
mais $abbaab \notin L_1 \cdot L_2$

Plan

Notion d'automate standard

Idéalement, pour construire un automate reconnaissant $L_1 \cdot L_2$, il faut :

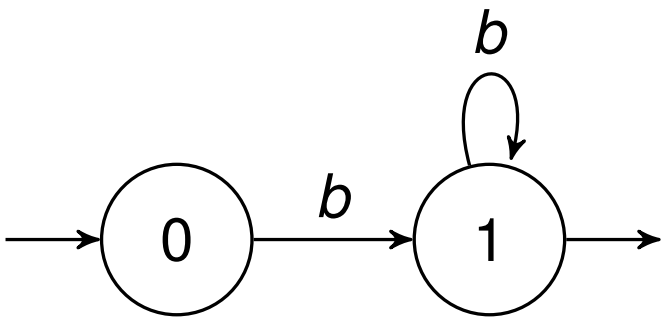
- $|I_2| = 1$ — $\mathcal{A}_2$ n'a qu'un état initial i
- aucune transition n'arrive en i — sinon on risque de repartir dans $\mathcal{A}_1$

Définition

Un automate fini $\mathcal{A} = (A, Q, I, F, E)$ est dit standard lorsque :

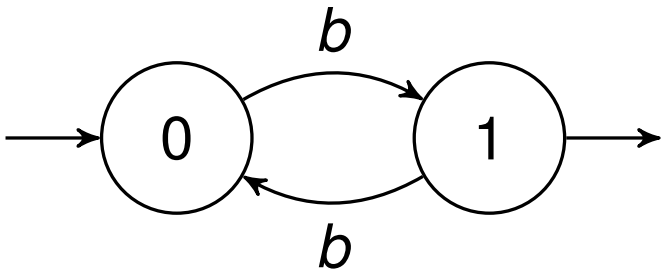
1. I est un singleton $\{i\}$,
2. pour tout $a \in A$ et tout $q \in Q$, on a $(q, a, i) \notin E$.

Exemples d'automates standards



Standard

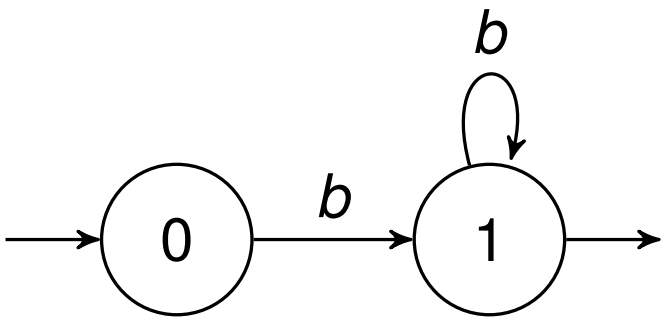
$L(\mathcal{A}_1) = \{b^m,\ m \in \mathbb{N}^*\}$



Pas standard

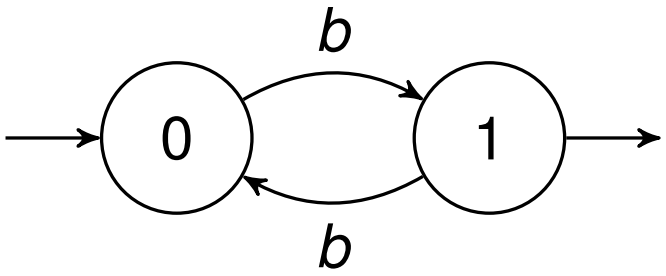
$L(\mathcal{A}_2) = \{b^{2m+1},\ m \in \mathbb{N}\}$

Exemples d'automates standards



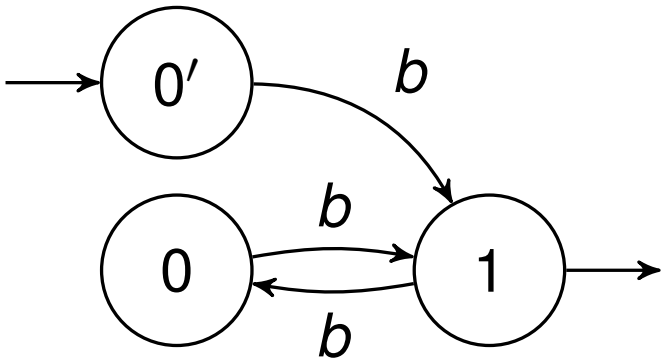
Standard

$L(\mathcal{A}_1) = \{b^m,\ m \in \mathbb{N}^*\}$



Pas standard

$L(\mathcal{A}_2) = \{b^{2m+1},\ m \in \mathbb{N}\}$



Standard

$L(\mathcal{A}_3) = \{b^{2m+1},\ m \in \mathbb{N}\}$

Standardisation

Propriété

Tout automate fini est équivalent à un automate fini standard.

Preuve = Procédé de standardisation

À partir de $\mathcal{A} = (A, Q, I, F, E)$, construire $\mathcal{A}' = (A, Q \cup \{i\}, \{i\}, F', E')$ où

- $i \notin Q$ — i nouvel état
- $F' := \begin{cases} F & \text{si } I \cap F = \emptyset \\ F \cup \{i\} & \text{sinon} \end{cases}$ — $\varepsilon \in L(\mathcal{A}) \Rightarrow i$ final
- $E' := E \cup \{(i, a, q) \mid \exists\, p \in I,\ (p, a, q) \in E\}$
  - $\rightsquigarrow$ Pour chaque $p \xrightarrow{a} q$ dans $\mathcal{A}$ avec p initial, ajouter la transition $i \xrightarrow{a} q$.

Plan

Produit de deux langages reconnaissables

Propriété

Si L_1 et L_2 sont reconnaissables, alors $L_1 \cdot L_2$ aussi.

Preuve :

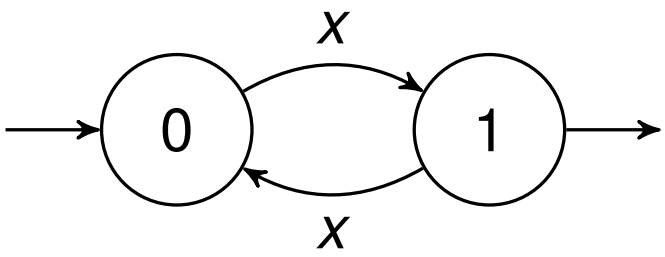
1. Choisir $\mathcal{A}_1$ (resp. $\mathcal{A}_2$) reconnaissant L_1 (resp. L_2)
2. Standardiser $\mathcal{A}_2$ pour n'avoir qu'un seul état initial i_2, $\quad \mathcal{A}_2 \rightsquigarrow \mathcal{A}'_2$
3. Fusionner chaque état final $f \in F_1$ dans $\mathcal{A}_1$ avec une copie de i_2

Ainsi, on définit l'automate $\mathcal{A} = (A, Q_1 \cup Q_2, I_1, F, E)$ où :

- $F := \begin{cases} F_2 & \text{si } I_2 \cap F_2 = \emptyset \\ F_1 \cup F_2 & \text{sinon} \end{cases}$ $\qquad \varepsilon \in L_2 \Rightarrow L_1 \subset L_1 \cdot L_2$
- $E := E_1 \cup E_2 \cup \{(q_1, a, q_2),\ q_1 \in F_1 \text{ et } (i_2, a, q_2) \in E'_2\}$

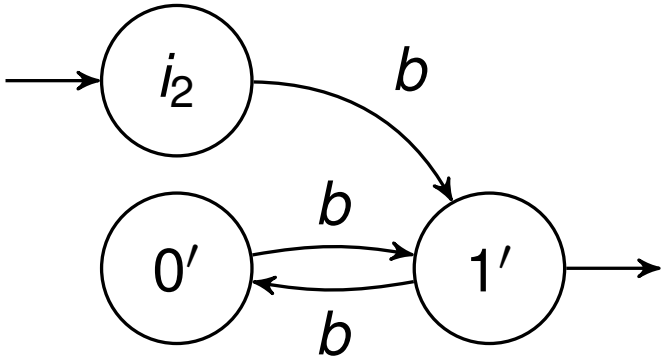
Produit de deux langages reconnaissables – Exemple

On reprend l'exemple :



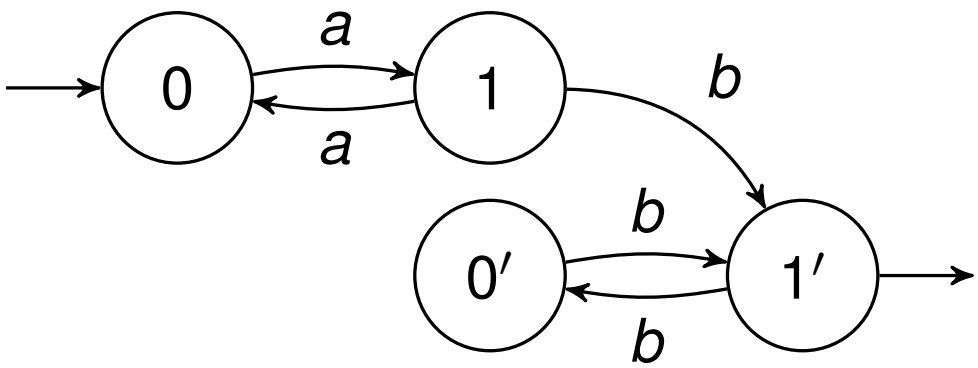
$$L_1 = L(\mathcal{A}_1) = \{a^{2n+1}, n \in \mathbb{N}\}$$
$$L_2 = L(\mathcal{A}_2) = \{b^{2m+1}, m \in \mathbb{N}\}$$

On standardise $\mathcal{A}_2$:



$L(\mathcal{A}'_2) = L_2$

On fusionne 1 et i_2 :



$L(\mathcal{A}) = L_1 \cdot L_2$

Étoile d'un langage reconnaissable

Propriété

Si L est reconnaissables, alors L^* l'est aussi.

Preuve :

1. Prendre un automate standard $\mathcal{A}$ tel que $L(\mathcal{A}) = L$
2. Fusionner chaque $f \in F$ à une copie de i
3. Rendre i final

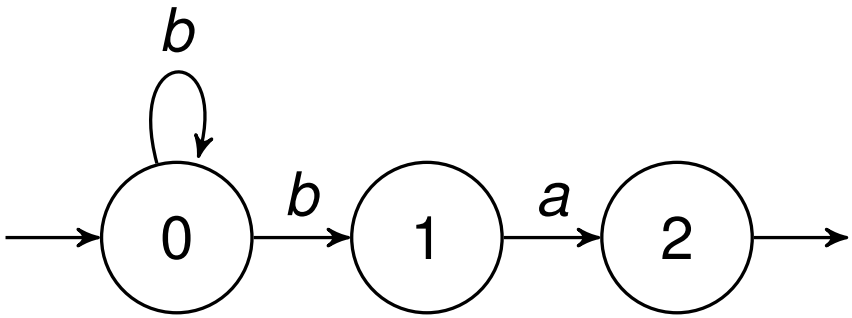
$\varepsilon \in L^*$

Au final, cela revient à construire $\mathcal{A}' := (A, Q, \{i\}, F \cup \{i\}, E')$ où

$$E' := E \cup \{(f, a, q),\ f \in F \text{ et } (i, a, q) \in E\}.$$

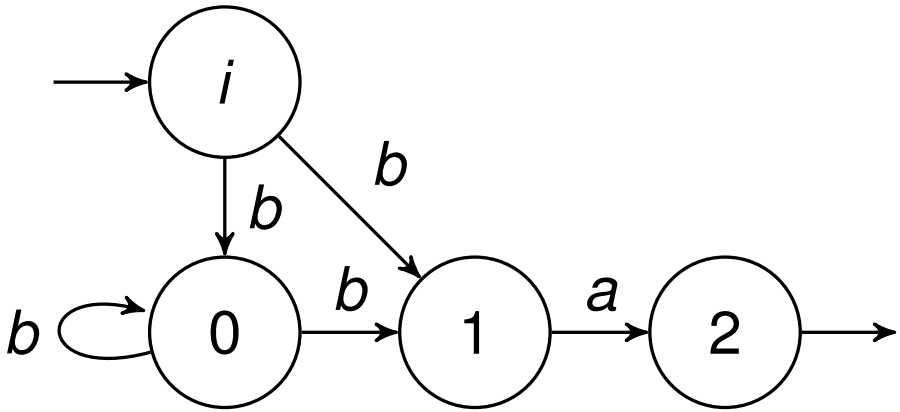
Étoile d'un langage reconnaissable – Exemple

On considère :



$$L = L(\mathcal{A}) = \{b^n a,\ n \in \mathbb{N}^*\}$$

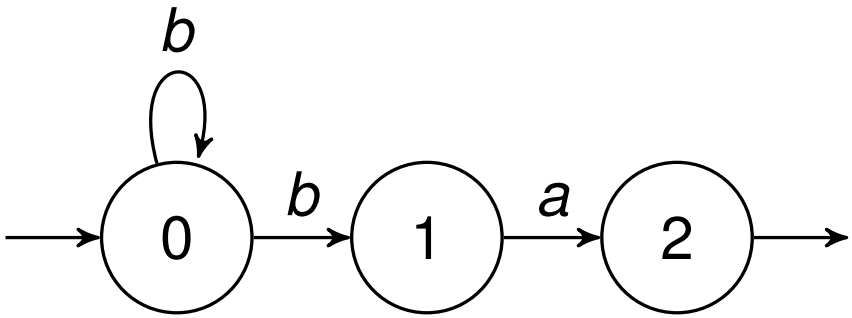
On applique la construction du transparent précédent :



1. Standardisation

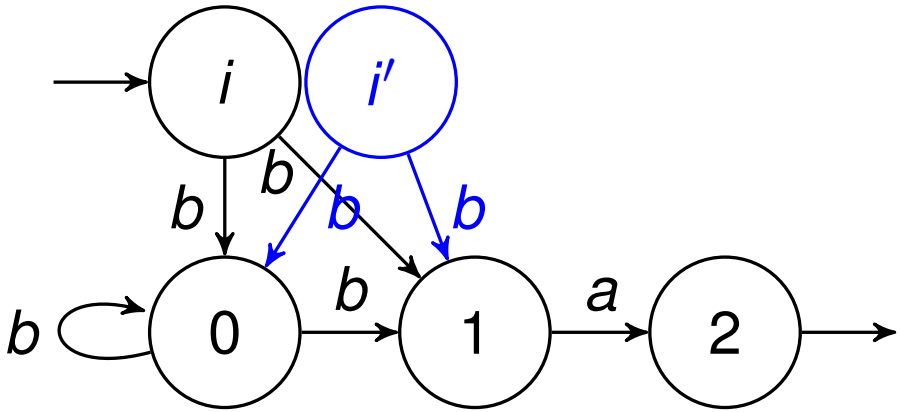
Étoile d'un langage reconnaissable – Exemple

On considère :



$$L = L(\mathcal{A}) = \{b^n a,\ n \in \mathbb{N}^*\}$$

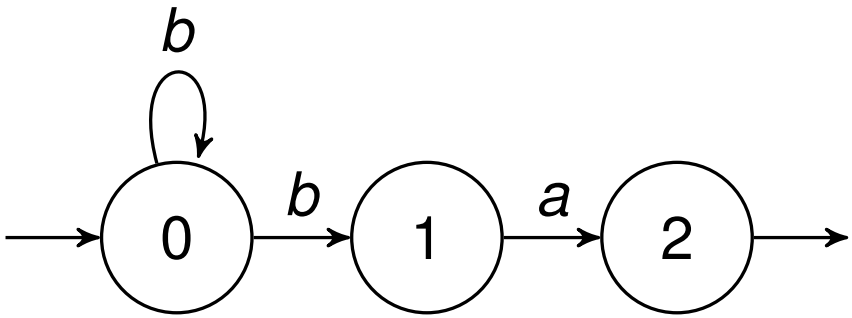
On applique la construction du transparent précédent :



2. Ajouter une copie de i par état final . . .

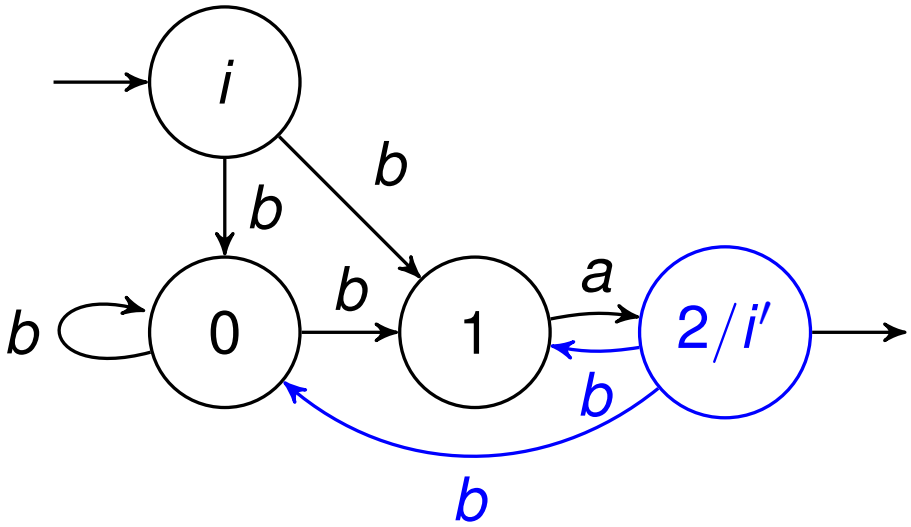
Étoile d'un langage reconnaissable – Exemple

On considère :



$$L = L(\mathcal{A}) = \{b^n a,\ n \in \mathbb{N}^*\}$$

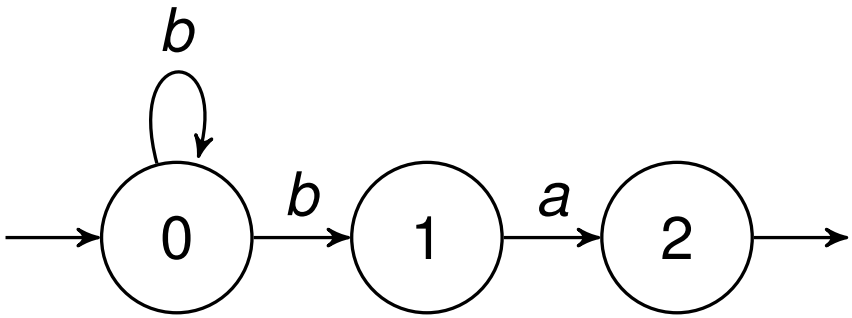
On applique la construction du transparent précédent :



2. . . . et faire les fusions

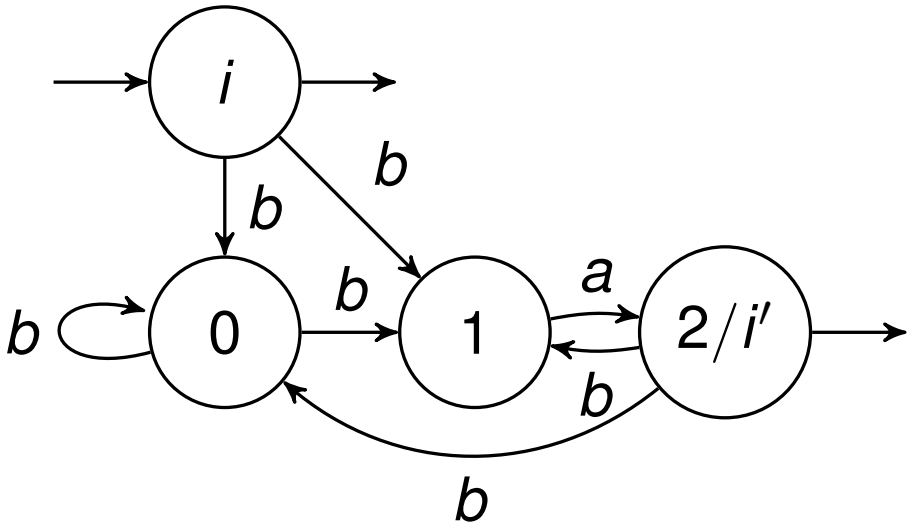
Étoile d'un langage reconnaissable – Exemple

On considère :



$$L = L(\mathcal{A}) = \{b^n a,\ n \in \mathbb{N}^*\}$$

On applique la construction du transparent précédent :



3. Rendre i final