Automates, langages et compilation

Automates finis

Isabelle Ryl

2024 – 2025

Cours de L3 - Université Paris Dauphine-PSL

1. Automates finis

1.1 Présentation informelle

1.2 Automates finis déterministes

1.3 Langages reconnaissables

1.4 Automates non complets

1.5 Opérations sur les reconnaissables

2. Minimisation

3. Automates finis non-déterministes

3.1 Opérations sur les reconnaissables

4. Théorème de Kleene

4.1 Système d'équations

4.2 Algorithme de Brzozowski et McCluskey

Automates finis

Automates finis

Présentation informelle

De quoi parlons-nous ?

Jusqu'ici, nous avons défini la notion de langage et nous avons introduit les expressions rationnelles qui permettent de les décrire de manière compacte

Il est utile de pouvoir « reconnaître » les langages, *i.e.* de décider si un mot appartient ou non à un langage, la décision est un problème important en informatique

Nous allons donc introduire un outil : les automates finis

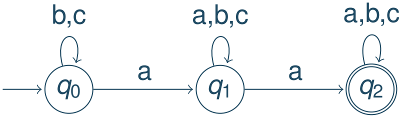
Principe de base

Un automate est un « algorithme » qui permet de décider si un mot appartient ou non à un langage

- partant d'un état initial
- les lettres du mot candidat vont être lues une à une de gauche à droite
- en suivant un « chemin » ou un « calcul » dans l'automate
- chaque lettre permettant de passer d'un état de contrôle au suivant

Si un tel chemin existe et mène à un état final, alors le mot est « reconnu », il appartient donc au langage, dans le cas contraire, le mot n'appartient pas au langage

Exemple

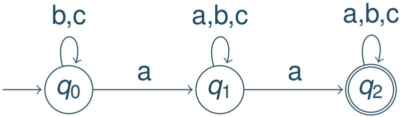


- q_0, q_1, q_2 sont les états
- q_0 est l'état initial
- q_2 est l'état final initial

Le mot *bacaa* est-il un mot reconnu ?

Chemin $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{c} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_2$ ➥Oui !

Exemple



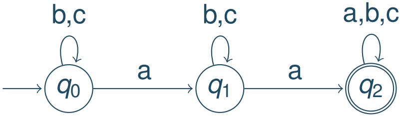
Le mot $u = bacaa$ est reconnu car il existe un chemin d'un état initial à un état final étiqueté par les lettres de u :

$$q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{c} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_2$$

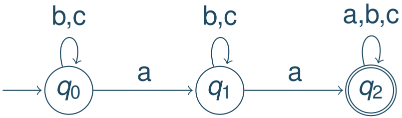
Ce chemin est-il unique ? Pas dans ce cas, autre exemple de chemin :

$$q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{c} q_1 \xrightarrow{a} q_1 \xrightarrow{a} q_2$$

Modification de l'automate :



Exemple



Le mot $v = ab$ est-il reconnu ? Non car :

$$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_1$$

ne mène pas à un état terminal (et aucun autre chemin étiqueté par les lettres du mot v)

Automates finis

Automates finis déterministes

Automates finis déterministes complets

Définition

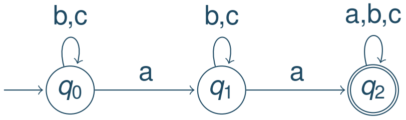
Un **automate fini déterministe complet** est un quintuplet $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ tel que :

- Σ est un alphabet fini
- Q est un ensemble fini d'états
- $q_0 \in Q$ est l'état initial
- $F \subseteq Q$ est l'ensemble des états finaux
- δ est une fonction de $Q \times \Sigma$ dans Q appelée fonction de transition

Notation. Une notation utilisée pour $\delta(q, x) = q'$ est $q \xrightarrow{x} q'$

Exemple de description d’automate et de représentation

- $\Sigma = \{a, b, c\}$
- $Q = \{q_0, q_1, q_2\}$
- q_0 est l’état initial
- $F = \{q_2\}$
- $$\begin{array}{rccc} \delta : & Q \times \Sigma & \longrightarrow & Q \\ & (q_0, a) & \longmapsto & q_1 \\ & (q_0, b) & \longmapsto & q_0 \\ & (q_0, c) & \longmapsto & q_0 \\ & (q_1, a) & \longmapsto & q_2 \\ & (q_1, b) & \longmapsto & q_1 \\ & (q_1, c) & \longmapsto & q_1 \\ & (q_2, a) & \longmapsto & q_2 \\ & (q_2, b) & \longmapsto & q_2 \\ & (q_2, c) & \longmapsto & q_2 \end{array}$$



Fonctionnement d'un automate

Informellement : un mot u est reconnu (ou accepté) par un automate s'il existe dans l'automate un chemin de l'état initial à un état final tel que la concaténation des étiquettes des transitions soit égale à u.

Soit $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ un automate

- Une **configuration** de l'automate est un couple (q_i, u) avec $q_i \in Q$ l'état courant de l'automate et $u \in \Sigma^*$ le mot restant à lire
- Un pas de calcul de l'automate est un changement de configuration d'une configuration (q_i, au) à une configuration (q_j, u) avec $q_i, q_j \in Q$, $u \in \Sigma^*$, $a \in \Sigma$ et $\delta(q_i, a) = q_j$, on note $(q_i, au) \rightarrow (q_j, u)$

Calcul

Un **calcul** de l'automate $\mathcal{A}$ sur un mot $u = x_0x_1\ldots x_nv$ à partir d'un état q_i est une suite de pas de calcul :

$$(q_i, u) \rightarrow (r_1, x_1 \ldots x_n v) \rightarrow \cdots \rightarrow (r_{n-1}, x_{n-1}x_nv) \rightarrow (r_n, x_nv) \rightarrow (q_j, v)$$

- Le résultat du calcul est q_j et on note aussi $(q_i, u) \xrightarrow{*} (q_j, v)$
- Si $q_i = q_0$, on l'appelle simplement calcul de $\mathcal{A}$ sur u

Notations alternatives. Si $v = \varepsilon$, on note également :

$$q_i \xrightarrow{x_0} r_1 \xrightarrow{x_1} r_2 \ldots r_n \xrightarrow{x_n} q_j$$

ou

$$q_i \xrightarrow[*]{u} q \text{ ou } \delta^*(q_i, u) = q$$

Automates finis

Langages reconnaissables

Langage reconnu

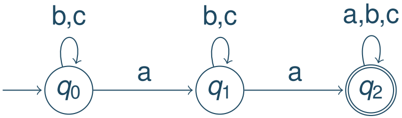
Soit $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ un automate

Un mot $u \in \Sigma^*$ est **reconnu** ou **accepté** par $\mathcal{A}$ s'il existe un calcul de $\mathcal{A}$ sur u, $(q_0, u) \xrightarrow{*} (q, \varepsilon)$ avec $q \in F$, appelé calcul réussi

Le **langage reconnu** (ou accepté) par $\mathcal{A}$ est :

$$\mathcal{L}(\mathcal{A}) = \{u \in \Sigma^* \mid (q_0, u) \xrightarrow{*} (q, \varepsilon) \text{ avec } q \in F\}$$

Exemple de calcul



Calcul réussi : $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{c} q_2$

aussi noté $q_0 \xrightarrow[*]{baac} q_2$

➥*baac* est reconnu par l'automate

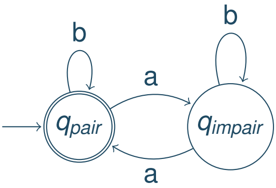
➥L'automate reconnait les mots sur $\{a, b, c\}$ qui contiennent au moins 2 a

Exemple - Nombre pair de *a*

<u>Question</u> : trouver un automate qui reconnait le langage sur $\Sigma = \{a, b\}$ des mots qui contiennent un nombre pair de *a*

<u>Raisonnement</u> : essayer de trouver des classes de mots qui vont permettre de différencier les états :

- pour les mots qui contiennent un nombre impair de *a*, il faut en lire encore au moins 1
- pour les mots qui contiennent un nombre pair de *a*, on peut s'arrêter si on veut ou continuer, mais si on relit un *a* on revient dans le premier cas

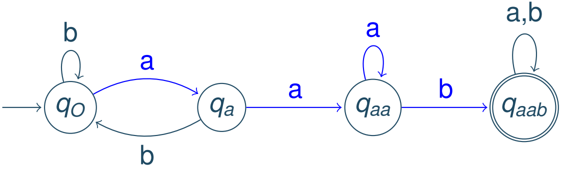


Exemple - *aab* facteur

<u>Question</u> : trouver un automate qui reconnait le langage des mots sur $\Sigma = \{a, b\}$ dont *aab* est un facteur

<u>Raisonnement</u> :

- Construction des états en fonction de l'avancée dans la lecture du facteur recherché

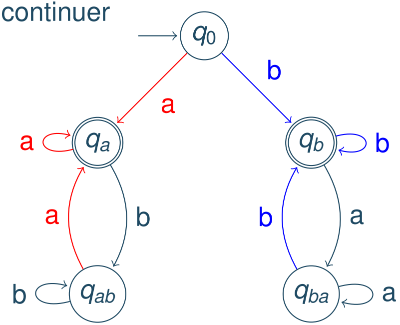


Exemple - 1ère et dernière lettre identiques

Question : trouver un automate qui reconnait le langage des mots sur $\Sigma = \{a, b\}$ dont la première et la dernière lettre sont identiques

Raisonnement :

- il faut savoir quelle est la première lettre, il va donc y avoir un branchement dès le départ
- ensuite chaque fois que nous aurons lu la même lettre nous pourrons nous arrêter ou continuer



* Figure de Sipser, 3ème édition

Langages reconnaissables

Un langage est **reconnaissable** s'il existe un automate fini qui le reconnait

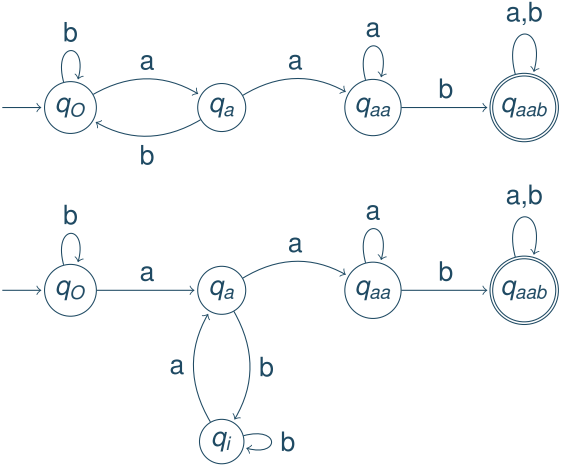
La famille des langages sur Σ^* reconnaissables par un automate fini déterministe est notée $Rec(\Sigma^*)$

Tout automate fini reconnait un et un seul langage, mais un langage peut être reconnu par plusieurs automates

Deux automates finis sont **équivalents** s'ils reconnaissent le même langage

Exemple de 2 automates reconnaissant le même langage

Automates reconnaissant le langage des mots dont *aab* est un facteur



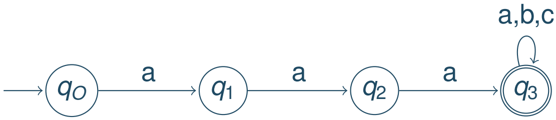
➥L'état q_i n'apporte rien mais ne change pas le langage reconnu

Automates finis

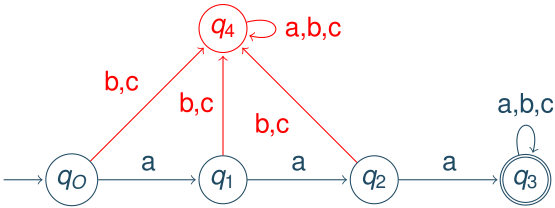
Automates non complets

État puits et spécification partielle

Soit l'automate qui reconnait les mots sur $\{a, b, c\}$ commençant par *aaa* :



☛Cet automate n'est pas complet ! Un automate fini déterministe complet :



➥q_4 est un **état puits**

Automates finis déterministes

Définition

Un **automate fini déterministe** (ou DFA) est un quintuplet $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ tel que :

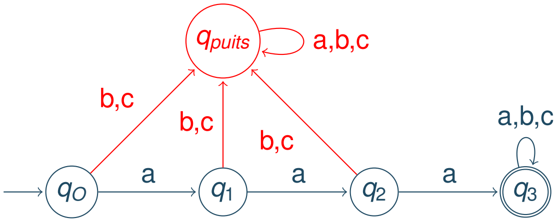
- Σ est un alphabet fini
- Q est un ensemble fini d'états
- $q_0 \in Q$ est l'état initial
- $F \subseteq Q$ est l'ensemble des états finaux
- δ est une fonction **partielle** de $Q \times \Sigma$ dans Q appelée fonction de transition

Les automates finis déterministes et les automates finis déterministes complets sont équivalents

Automates finis déterministes complets ou non

Soit un **automate fini déterministe** $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$, on peut toujours construire un **automate fini déterministe complet** $\mathcal{A}' = (\Sigma, Q', q_0, F, \delta')$ qui reconnait le même langage, avec :

- $Q' = Q \cup \{q_{puits}\}$
- $\forall x \in \Sigma, \forall q \in Q$
  - $\delta'(q, x) = \delta(q, x)$ si $\delta(q, x)$ existe
  - $\delta'(q, x) = q_{puits}$ sinon
- $\forall x \in \Sigma$, $\delta'(q_{puits}, x) = q_{puits}$



Automates finis

Opérations sur les reconnaissables

Clôture par complémentaire

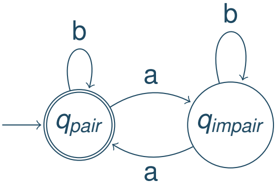
Proposition. Les langages reconnaissables sont clos par complémentaire

Soit $\mathcal{L}$ un langage tel que $\mathcal{L} = \mathcal{L}(\mathcal{A})$ avec $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ un automate fini déterministe complet. Posons $\mathcal{A}' = (\Sigma, Q, q_0, F', \delta)$ avec $F' = Q \setminus F$, alors $\mathcal{L}(\mathcal{A}') = \bar{\mathcal{L}}$.

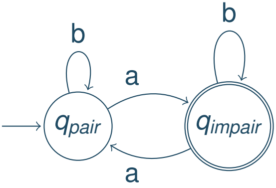
En effet, tout calcul réussi de $\mathcal{A}$ termine dans un état de F, et est donc un calcul qui échoue de $\mathcal{A}'$ et vice versa

Clôture par complémentaire - Exemples 1/2

Langage des mots sur $\{a, b\}$ qui contiennent un nombre pair de a



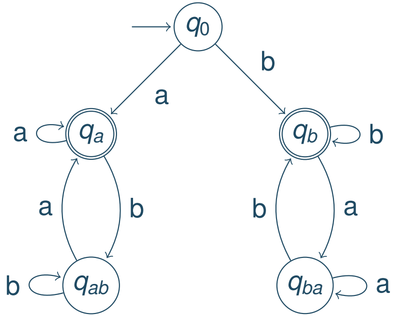
Complémentaire



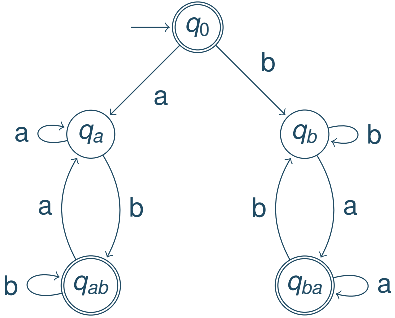
➥ Langage des mots sur $\{a, b\}$ qui contiennent un nombre impair de a

Clôture par complémentaire - Exemples 2/2

Langage des mots sur $\Sigma = \{a, b\}$ dont la première et la dernière lettre sont identiques



Complémentaire



➥ Langage des mots sur $\Sigma = \{a, b\}$ dont la première et la dernière lettre sont différentes

Clôture par union

Proposition. Les langages reconnaissables sont clos par union

Soient $\mathcal{L}_1$ et $\mathcal{L}_2$ deux langages reconnus par $\mathcal{A}_1 = (\Sigma, Q_1, q_0^1, F_1, \delta_1)$ et $\mathcal{A}_2 = (\Sigma, Q_2, q_0^2, F_2, \delta_2)$ des automates finis déterministes complets

Construisons $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ qui reconnait $\mathcal{L}_1 \cup \mathcal{L}_2$ avec

- $Q = Q_1 \times Q_2$
- $q_0 = (q_0^1, q_0^2)$
- $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$
- $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$

➥ Il est fini déterministe

Idée. Pour tout chemin de $\mathcal{A}_1$

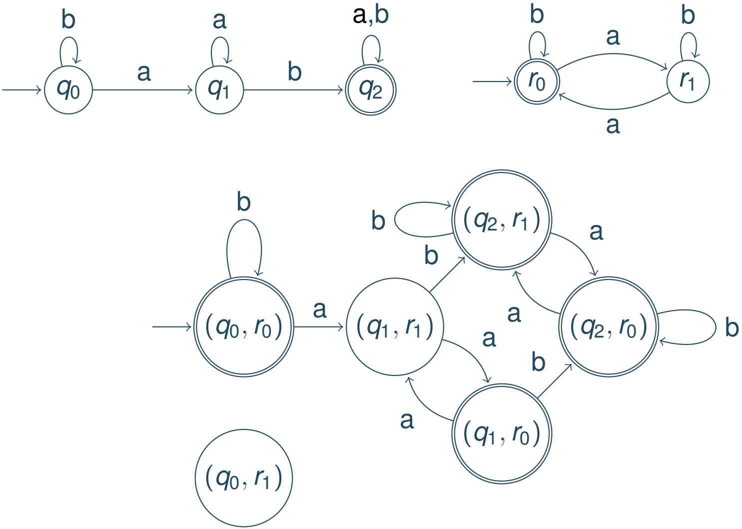
$$q_0^1 \xrightarrow{x_0} q_1^1 \xrightarrow{x_1} q_2^1 \dots \xrightarrow{x_{n-1}} q_n^1$$

avec $q_n^1 \in F_1$, il existe un chemin de $\mathcal{A}$

$$(q_0^1, q_0^2) \xrightarrow{x_0} (q_1^1, \dots) \xrightarrow{x_1} (q_2^1, \dots) \dots \xrightarrow{x_{n-1}} (q_n^1, \dots)$$

avec $(q_n^1, \dots) \in F$

Clôture par union - Exemple



Clôture par différence

Proposition. Les langages reconnaissables sont clos par différence

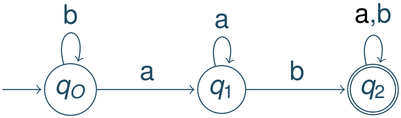
Soient $\mathcal{L}_1$ et $\mathcal{L}_2$ deux langages reconnus par $\mathcal{A}_1 = (\Sigma, Q_1, q_0^1, F_1, \delta_1)$ et $\mathcal{A}_2 = (\Sigma, Q_2, q_0^2, F_2, \delta_2)$ des automates finis déterministes complets

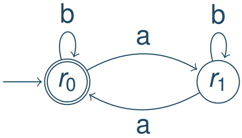
Construisons $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ qui reconnait $\mathcal{L}_1 - \mathcal{L}_2$ avec

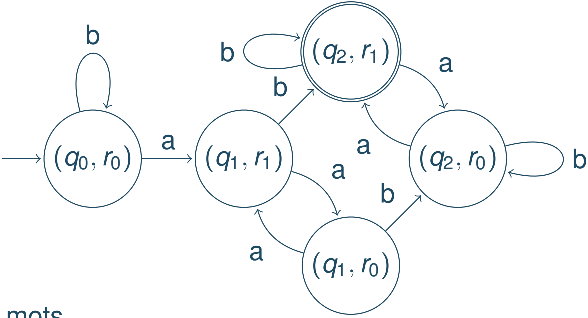
- $Q = Q_1 \times Q_2$ ➥ idem union
- $q_0 = (q_0^1, q_0^2)$ ➥ idem union
- $F = F_1 \times (Q_2 \setminus F_2)$ ➥ !!!!
- $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$ ➥ idem union

Idée. Construction identique à celle de l'union, *i.e.* on vérifie l'appartenance aux 2 langages en faisant fonctionner les automates « en parallèle » mais on discrimine les mots reconnus de manière différente en excluant les mots reconnus par $\mathcal{A}_2$

Clôture par différence - Exemple







<u>Exemples de mots</u>

$aabb \in \mathcal{L}_1, aabb \in \mathcal{L}_2 \Rightarrow aabb \notin \mathcal{L}_1 - \mathcal{L}_2$

$babb \in \mathcal{L}_1, babb \notin \mathcal{L}_2 \Rightarrow aabb \in \mathcal{L}_1 - \mathcal{L}_2$

Clôture par intersection

Proposition. Les langages reconnaissables sont clos par intersection

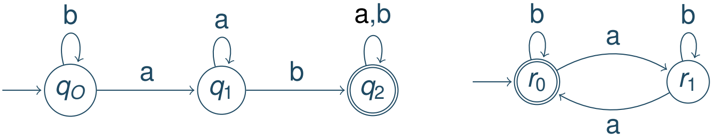
Soient $\mathcal{L}_1$ et $\mathcal{L}_2$ deux langages reconnus par $\mathcal{A}_1 = (\Sigma, Q_1, q_0^1, F_1, \delta_1)$ et $\mathcal{A}_2 = (\Sigma, Q_2, q_0^2, F_2, \delta_2)$ des automates finis déterministes complets

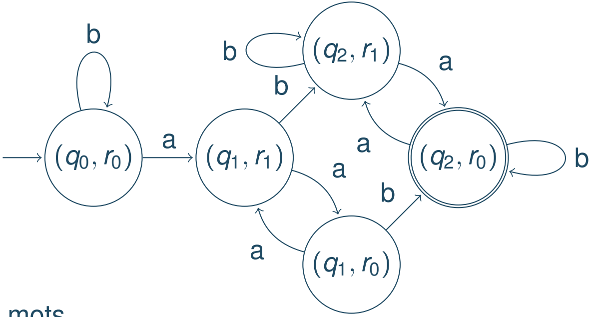
Construisons $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ qui reconnait $\mathcal{L}_1 \cap \mathcal{L}_2$ avec

- $Q = Q_1 \times Q_2$ ➥ idem union
- $q_0 = (q_0^1, q_0^2)$ ➥ idem union
- $F = F_1 \times F_2$ ➥ !!!!
- $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$ ➥ idem union

Idée. Construction identique à celle de l'union, *i.e.* on vérifie l'appartenance aux 2 langages en faisant fonctionner les automates « en parallèle » mais on discrimine les mots reconnus de manière différente en exigeant qu'ils soient reconnus par chacun des automates

Clôture par Intersection - Exemple





Exemples de mots

$aabb \in \mathcal{L}_1, aabb \in \mathcal{L}_2 \Rightarrow aabb \in \mathcal{L}_1 \cap \mathcal{L}_2$

$babb \in \mathcal{L}_1, babb \notin \mathcal{L}_2 \Rightarrow aabb \notin \mathcal{L}_1 \cap \mathcal{L}_2$

Clôtures

Nous avons donc vu que les langages recconnaissables sont clos par :

- complémentaire
- union ➥ opérations rationnelles
- différence
- intersection

avec chaque fois la construction de l'automate déterministe associé

Minimisation

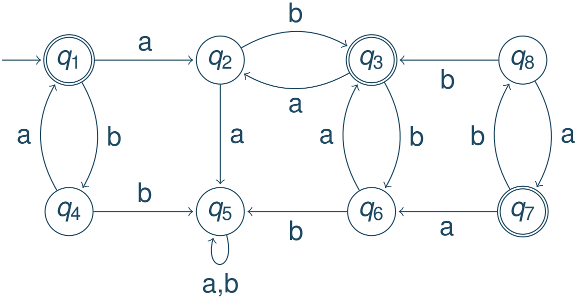
Minimiser un automate

Nous avons vu que tout automate fini reconnait un unique langage, mais qu'un langage peut être reconnu par plusieurs automates, parfois ceux-ci peuvent avoir un nombre important d'états et être lourds à implanter / manipuler

Tout langage reconnaissable est reconnu par un unique automate fini déterministe complet qui minimise le nombre d'états, *i.e.* tout automate fini déterministe complet reconnaissant le même langage possède au moins autant d'états, il est appelé **automate minimal**

Exemple conducteur*

Automate du langage $(ab + ba)^*$, tiré de (*)



* Figure de « Elements of the Theory of computation » Lewis&Papadimitriou

États accessibles

Soit $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ un automate fini déterministe

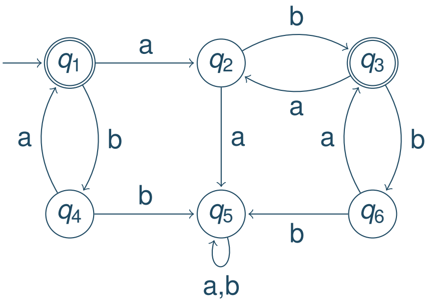
Un état $q \in Q$ est dit **accessible** s'il existe $u \in \Sigma^*$ tel que $\delta^*(q_0, u) = q$, il est dit **co-accessible** s'il existe $u \in \Sigma^*$ tel que $\delta^*(q, u) \in F$

L'automate accessible équivalent à $\mathcal{A}$ est $\mathcal{A}' = (\Sigma, Q', q_0, F', \delta')$ tel que :

- Q' est l'ensemble des états accessibles de $\mathcal{A}$
- $F' = F \cap Q'$
- δ' est la restriction de δ à Q'

➥ Ceci conduit à une première diminution du nombre d'états de l'automate

Exemple conducteur



Langage résiduel

Le résiduel d'un langage $\mathcal{L} \subseteq \Sigma^*$ par un mot $u \in \Sigma^*$, est

$$u^{-1}\mathcal{L} = \{v \in \Sigma^* \mid uv \in \mathcal{L}\}$$

<u>Exemples.</u>

$\mathcal{L} = \{aa, aab, acb, daa\}$ et $u = a$ alors $u^{-1}\mathcal{L} = \{a, ab, cb\}$
$\mathcal{L} = \{aa, aab, acb, daa\}$ et $u = aa$ alors $u^{-1}\mathcal{L} = \{\varepsilon, b\}$
$\mathcal{L} = \{aa, aab, acb, daa\}$ et $u = cb$ alors $u^{-1}\mathcal{L} = \emptyset$

Exemple conducteur - Calcul des résiduels

Pour le langage $L = (ab + ba)^*$

- $a^{-1}L = \; b(ab + ba)^* \; = aba^{-1}L = baa^{-1}L$
- $b^{-1}L = a(ab + ba)^* \; = abb^{-1}L = bab^{-1}L$
- $ab^{-1}L = L$
- $ba^{-1}L = L$
- $aa^{-1}L = \emptyset \; = bb^{-1}L$

➥on trouve 4 résiduels

Résiduel et automates

Soit $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ un automate fini déterministe. Pour chaque état $q \in Q$, on définit $L_q = \{v \in \Sigma^* \mid \delta^*(q, v) \in F\}$

➥ Intuitivement : le langage que l'on peut reconnaître à partir de q

Clairement si $\delta^*(q_0, u) = q$ alors $u^{-1}\mathcal{L}(\mathcal{A}) = L_q$, donc

$$\{u^{-1}\mathcal{L}(\mathcal{A}) \mid u \in \Sigma^*\} = \{L_q \mid q \in Q\}$$

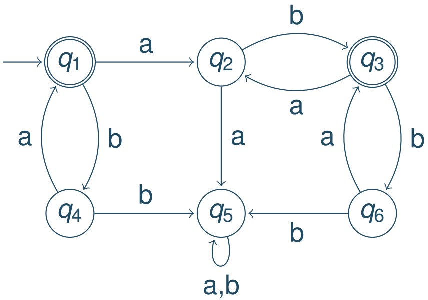
Cette remarque permet d'établir un fait important pour la suite : le nombre de résiduels de $\mathcal{L}(\mathcal{A})$ est fini (et inférieur au nombre d'états de $\mathcal{A}$)

Construction de l'automate des résiduels

Soit $\mathcal{L} \in \Sigma^*$ un langage dont le nombre de résiduels est fini. L'automate des résiduels de $\mathcal{L}$ est $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ avec :

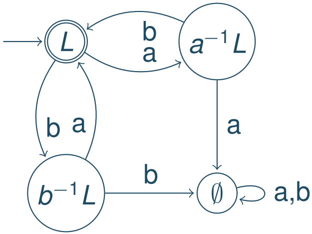
- $Q = \{u^{-1}\mathcal{L} \mid u \in \Sigma^*\}$ ➥ chaque état correspond à un résiduel
- $q_0 = \mathcal{L}$ ➥ l'état initial correspond au résiduel par ε
- $F = \{u^{-1}\mathcal{L} \mid u \in \mathcal{L}\}$ ➥ un état est final s'il contient ε
- $\forall a \in \Sigma, u \in \Sigma^*, \delta(u^{-1}\mathcal{L}, a) = ua^{-1}\mathcal{L}$
  ➥ il existe une transition entre un q_i et un q_j étiquetée par a si q_j est le résiduel de q_i par a

Exemple conducteur



$\forall a \in \Sigma, u \in \Sigma^*,$
$\delta(u^{-1}\mathcal{L}, a) = ua^{-1}\mathcal{L}$

➥ il existe une transition entre un q_i et un q_j étiquetée par a si q_j est le résiduel de q_i par a



Automate minimal

Proposition

L'automate fini déterministe complet minimal d'un langage $\mathcal{L}$ est son automate des résiduels

Preuve

Soit $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ des résiduels de $\mathcal{L}$. Nous devons montrer que

(1) $\mathcal{A}$ est fini déterministe complet ➥évident par construction

(2) $\mathcal{A}$ reconnait $\mathcal{L}$ ➥à la suite

(3)$\mathcal{A}$ est minimal ➥Par définition $|Q|$ est le nombre de résiduels de $\mathcal{L}$. Or ce nombre est inférieur au nombre d'états de tout DFA complet qui reconnaît $\mathcal{L}$, il est donc minimal (cf remarque « *le nombre de résiduels de* $\mathcal{L}(\mathcal{A})$ *est fini (et inférieur au nombre d'états de* $\mathcal{A}$*)* »)

Langage reconnu par l'automate des résiduels

Proposition Soit $\mathcal{L} \in \Sigma^*$ un langage et $\mathcal{A}$ son automate des résiduels, alors $\mathcal{L}(\mathcal{A}) = \mathcal{L}$

Preuve

Soit $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$, montrons que $\mathcal{A}$ reconnait $\mathcal{L}$. En utilisant la construction de la fonction δ, il est facile de montrer par récurrence sur la longueur de u que pour tout mot $u \in \mathcal{L}$, $\delta(q_0, u) = u^{-1}\mathcal{L}$. Donc par construction de F, $u \in \mathcal{L}(\mathcal{A})$.

Soit $u \in \mathcal{L}(\mathcal{A})$, alors $\delta(q_0, u) = q$ avec $q \in F$, *i.e.* $q = v^{-1}\mathcal{L}$ avec $v \in \mathcal{L}$, mais nous avons également $q = u^{-1}\mathcal{L}$ (cf ci-dessus)

Par définition du résiduel, $(u^{-1}\mathcal{L} = v^{-1}\mathcal{L}) \Leftrightarrow (\forall w \in \Sigma^*, uw \in \mathcal{L} \Leftrightarrow vw \in \mathcal{L})$, ce qui est en particulier vrai pour $w = \varepsilon$, donc $u \in \mathcal{L}$

Remarques

Algorithmes pour minimiser un automate

Il existe plusieurs méthodes pour construire l'automate minimal

➥ à voir en TD

Équivalence de 2 automates

Pour savoir si deux automates reconnaissent le même langage, il est à présent possible de construire leur automate minimal respectif et de les comparer

Automates finis non-déterministes

De quoi parle-t-on ?

Nous allons considérer des automates finis qui « ressemblent » aux automates finis déterministes, mais avec quelques contraintes en moins :

- l'automate peut avoir plusieurs états initiaux
- des transitions peuvent être étiquetées par le mot vide
- plusieurs transitions sortantes d'un état peuvent être étiquetées par la même lettre

ε-NFA

La définition la « plus large » d'un automate fini non-déterministe (NFA) admet des transitions étiquetées par le mot vide, *i.e.* des changements d'états « spontannés »

Un **ε-NFA** est un quintuplet $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ tel que :

- Σ est un alphabet fini
- Q est un ensemble fini d'états
- $I \subseteq Q$ est l'ensemble des états initiaux
- $F \subseteq Q$ est l'ensemble des états finaux
- $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ est une <u>relation</u> définissant les transitions

<u>Notation</u>. Si $(q, x, q') \in \delta$, on note $q \xrightarrow{x} q'$

<u>Remarque</u>. Il est possible de définir δ comme une fonction de $Q \times (\Sigma \cup \{\varepsilon\})$ dans 2^Q

Automates finis non-déterministes

Un **automate fini non-déterministe** est un quintuplet $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ tel que :

- Σ est un alphabet fini
- Q est un ensemble fini d'états
- $I \subseteq Q$ est l'ensemble des états initiaux
- $F \subseteq Q$ est l'ensemble des états finaux
- $\delta \subseteq Q \times \Sigma \times Q$ est une relation définissant les transitions

Calcul

- La **configuration** d'un NFA est définie comme celle d'un DFA : un couple (q_i, u) avec $q_i \in Q$ l'état courant de l'automate et $u \in \Sigma^*$ le mot restant à lire
- Un pas de calcul de l'automate est un changement de configuration d'une configuration (q_i, au) à une configuration (q_j, u) avec $q_i, q_j \in Q$, $u \in \Sigma^*$, $a \in \Sigma$ et $(q_i, a, q_j) \in \delta$, on note $(q_i, au) \to (q_i, u)$

➥Le **non-déterminisme** vient du fait que lorsque l'automate est dans une configuration (q_i, au), il peut exister plusieurs état $q_j^1, \ldots, q_j^n$ tels que $(q_i, a, q_j^1) \in \delta, \ldots, (q_i, a, q_j^n) \in \delta$, on ne sait donc pas quelle configuration suivante l'automate va « choisir »

- Un **calcul** de l'automate $\mathcal{A}$ sur un mot $u = x_0 x_1 \ldots x_n$ à partir d'un état q est défini de la même manière que pour un DFA
- Un mot u est **reconnu** s'il existe un calcul de $\mathcal{A}$ sur u terminant dans un état final (il peut en exister plusieurs)

Langage

Soit $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un automate fini non-déterministe, le langage reconnu par $\mathcal{A}$, noté

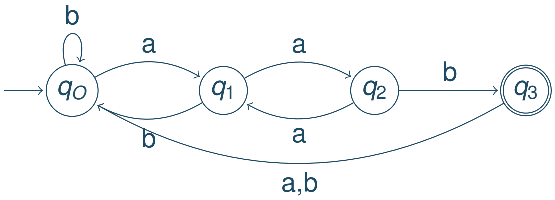
$$\mathcal{L}(\mathcal{A}) = \{u \in \Sigma^* \mid \exists q_i \in I, \exists q_f \in F, \text{ such that } q_i \xrightarrow[*]{u} q_f\}$$

➥Pour certains problèmes un automate non-déterministe peut être plus simple

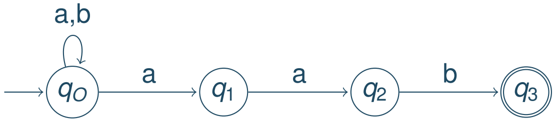
Exemple

Question Construire un automate qui reconnait les mots sur $\{a, b\}$ se terminant par *aab*

Automate déterministe



Automate non-déterministe



ε-NFA et NFA

Soit $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un ε-NFA.

La **ε-clôture** d'un état $q \in Q$ est l'ensemble des états que l'on peut atteindre à partir de q par des transitions étiquetées par ε :

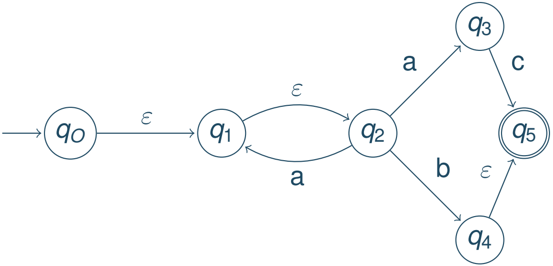
$$\mathcal{E}(q) = \{q' \in Q \mid q \xrightarrow[*]{\varepsilon} q'\}$$

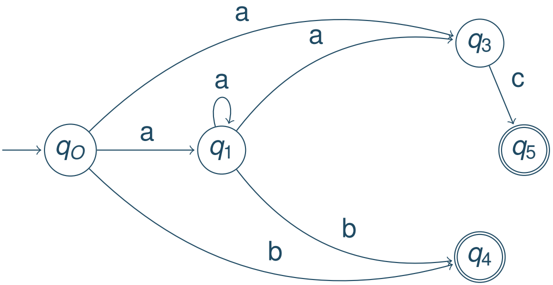
Théorème Pour tout ε-NFA, $\mathcal{A}$ il existe un NFA $\mathcal{A}'$ tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$

Construction Posons pour $\mathcal{A}$, $\mathcal{A}' = (\Sigma, Q, I, F', \delta')$ avec

- $F' = \{q \in Q \mid \mathcal{E}(q) \cap F \neq \emptyset\}$
- $\delta' = \{(q, x, q'') \mid q' \in \mathcal{E}(q) \wedge (q', x, q'') \in \delta\}$

ε-NFA et NFA - Exemple





Équivalence des classes d'automates

Nous avons vu que les ε-NFA sont équivalents aux NFA. On peut également sans perte de généralité considérer que les NFA ont un seul état initial

Par ailleurs, les automates finis déterministes sont clairement un cas particulier des automates finis non-déterministes avec :

- $|I| = 1$
- δ une fonction

Mais, plus étonnant :

Théorème de Rabin-Scott Tout langage reconnu par un NFA peut être reconnu par un DFA

Déterminisation

Soit $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ un NFA, nous pouvons considérer qu'il est sans ε-transition.

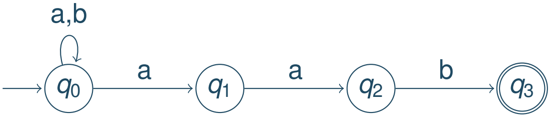
Construisons $\mathcal{A}' = (\Sigma, 2^Q, q_0, F', \delta')$ un DFA tel que :

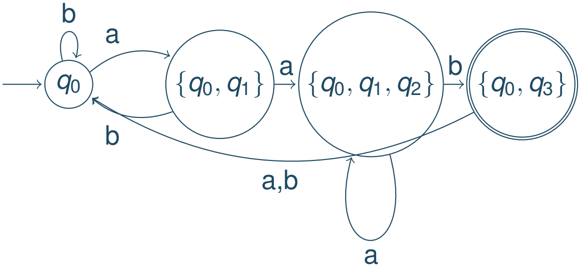
- $F' = \{q \in 2^Q \mid q \cap F \neq \emptyset\}$
- $\forall q' \in 2^Q, \forall x \in \Sigma, \delta'(q', x) = \bigcup_{q \in q'} \{q'' \mid (q, x, q'') \in \delta\}$

➥Par construction de la fonction δ', $\mathcal{A}'$ est déterministe

Exemple

Automate non-déterministe du langage des mots sur $\{a, b\}$ se terminant par *aab*





Déterminisation - Rappel

Soit $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ un NFA, nous pouvons considérer qu'il est sans ε-transition.

Construisons $\mathcal{A}' = (\Sigma, 2^Q, q_0, F', \delta')$ un DFA tel que :

- $F' = \{q \in 2^Q \mid q \cap F \neq \emptyset\}$
- $\forall q' \in 2^Q, \forall x \in \Sigma, \delta'(q', x) = \bigcup_{q \in q'} \{q'' \mid (q, x, q'') \in \delta\}$

➥Par construction de la fonction δ', $\mathcal{A}'$ est déterministe

... avant l'exemple, nous en étions là...

➥ Montrons que $\mathcal{A}'$ reconnait $\mathcal{L}(\mathcal{A})$

Preuve (1/2)

Soit $u \in \mathcal{L}(\mathcal{A})$ alors il existe un calcul $q_0 \xrightarrow[*]{u} q_F$ avec $q_F \in F$

Montrons par récurrence sur la longueur de u que pour tout calcul de $\mathcal{A}$, tel que $q_0 \xrightarrow[*]{u} q_i$ alors $\exists q' \in 2^Q$ avec $q_i \in q'$ tel que $q_0 \xrightarrow[*]{u} q'$

- pour $|u| = 0$, la propriété est vraie par définition de q_0 dans $\mathcal{A}$ et $\mathcal{A}'$
- supposons que la propriété est vraie pour $|u| \leq n$

Preuve (2/2)

Supposons que $u = va$ avec $a \in \Sigma$ et $|v| = n$,

$$\exists q_j \in Q \text{ tel que } q_0 \xrightarrow[*]{v} q_j \xrightarrow{a} q_i$$

Par hypothèse de récurrence, il existe un calcul dans $\mathcal{A}'$ tel que

$$\exists q' \in 2^Q \text{ avec } q_j \in q' \text{ et } q_0 \xrightarrow[*]{v} q'$$

Comme $q_j \in q'$ et $q_j \xrightarrow{a} q_i$, par définition de δ', $\delta'(q', a) = q''$ avec $q_i \in q''$

Par construction de F' si $q_i \in F$,$q'' \in F'$.

Exercice : montrer que réciproquement $\mathcal{A}$ reconnait $\mathcal{L}(\mathcal{A}')$ en utilisant la même technique

Automates finis non-déterministes

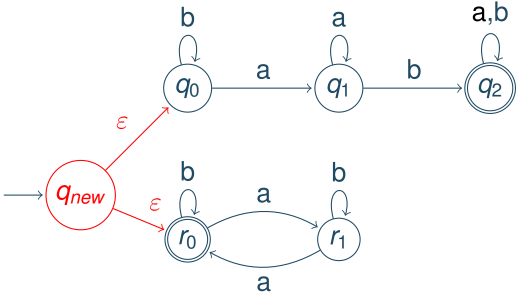
Opérations sur les reconnaissables

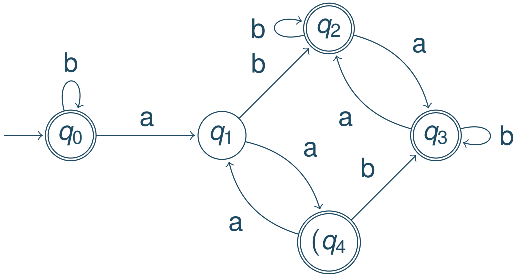
État initial unique, état final unique

Soit $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un automate fini, on peut toujours ramener $\mathcal{A}$ à un automate $\mathcal{A}' = (\Sigma, Q', \{q_0\}, \{q_F\}, \delta')$ avec

- $Q' = Q \cup \{q_0, q_F\}$, q_0 et q_F étant de nouveaux états
- $\delta' = \delta \cup \{(q_0, \varepsilon, q \mid q \in I\} \cup \{(q, \varepsilon, q_F \mid q \in F\}$
- Il n'existe pas de transition entrante dans q_0, et il n'existe pas de transition sortante de q_F

Clôture par union - Exemple





Clôture par concaténation

Proposition. Les langages reconnaissables sont clos par concaténation

Soient $\mathcal{A}_1 = (\Sigma_1, Q_1, q_1, F_1, \delta_1)$ et $\mathcal{A}_2 = (\Sigma_2, q_2, I_2, F_2, \delta_2)$ des automates finis, avec $Q_1 \cap Q_2 = \emptyset$

Construction. $\mathcal{A}' = (\Sigma_1 \cup \Sigma_2, Q_1 \cup Q_2, q_1, F_2, \delta')$ avec

- $\delta' = \delta_1 \cup \delta_2 \cup \{(q_f, \varepsilon, q_2) \mid q_f \in F_1\}$

Clôture par étoile

Proposition. Les langages reconnaissables sont clos par étoile

Construction. Soit $\mathcal{A} = (\Sigma, Q, q_0, \{q_F\}, \delta)$ un automate fini.

On construit $\mathcal{A}' = (\Sigma, Q, q_0, \{q_F\}, \delta')$ tel que

$$\delta' = \delta \cup \{(q_F, \varepsilon, q_0), (q_0, \varepsilon, q_F)\}$$

Nous avons vu que les langages reconnaissables sont clos par union, concaténation, étoile soit par les opérations rationnelles. Donc

$$Rat(\Sigma^*) \subseteq Rec(\Sigma^*)$$

➥Ceci nous permet également de proposer un algorithme pour construire l'automate associé à une expression rationnelle

Algorithme de Thompson

Cet algorithme va produire un automates :

- par induction sur la construction des expressions rationnelles
- ayant un unique état initial sans transition entrante
- un unique état final sans transition sortante

Mais

- avec un résultat non optimal en termes de taille

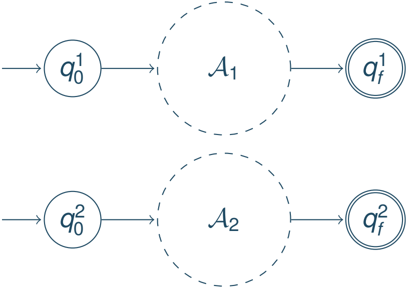
Briques élémentaires

Pour le langage $\emptyset$

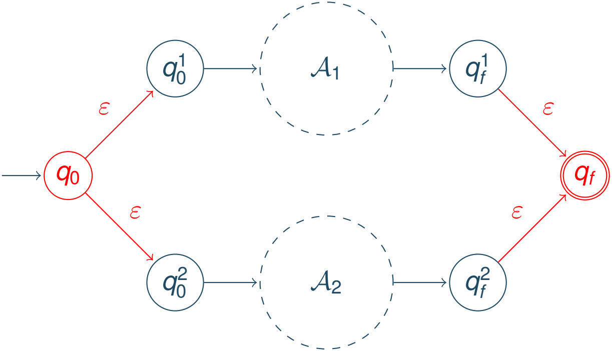
Pour le langage $\{\varepsilon\}$

Pour le langage $\{a\}$ avec $a \in \Sigma$

Pour l'union

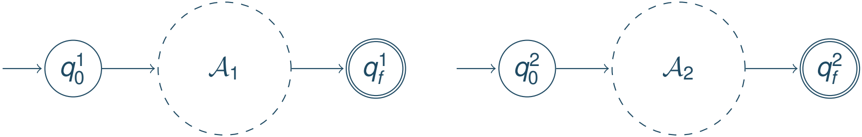


Pour l'union

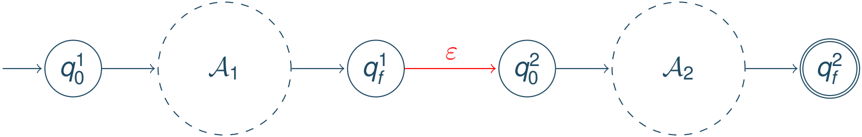


q_0^1
$\mathcal{A}_1$
q_f^1
ε
ε
q_0
q_f
ε
ε
q_0^2
$\mathcal{A}_2$
q_f^2

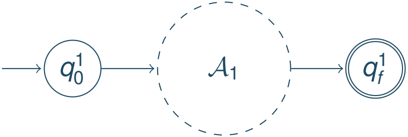
Pour la concaténation



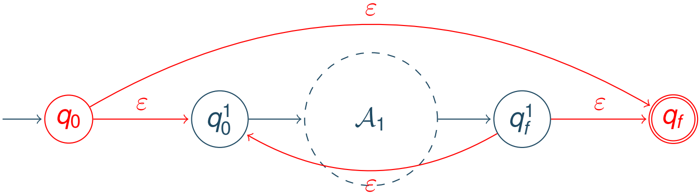
Pour la concaténation



Pour l'étoile



Pour l'étoile



Théorème de Kleene

Théorème de Kleene

Système d’équations

Où en sommes-nous ?

Nous avons vu :

- les expressions rationnelles et la classe de langages $\mathit{Rat}(\Sigma^*)$
- les DFA et et la classe de langages $\mathit{Rec}(\Sigma^*)$
- les ε-NFA, les NFA, l'équivalence entre ε-NFA et NFA et entre NFA et DFA, ils reconnaissent donc tous la même classe de langages $\mathit{Rec}(\Sigma^*)$
- et enfin l'inclusion $\mathit{Rat}(\Sigma^*) \subseteq \mathit{Rec}(\Sigma^*)$

<u>Théorème de Kleene</u>

Un langage est rationnel si et seulement si il est reconnaissable

<u>Preuve</u>

Nous avons vu que les langages reconnaissables sont clos par union, concaténation, étoile soit par les opérations rationnelles, et donc que $Rat(\Sigma^*) \subseteq Rec(\Sigma^*)$.

Reste à montrer la réciproque. Nous allons pour cela montrer que l'on peut construire l'expression rationnelle qui décrit le langage reconnu par un automate

Équation et langage

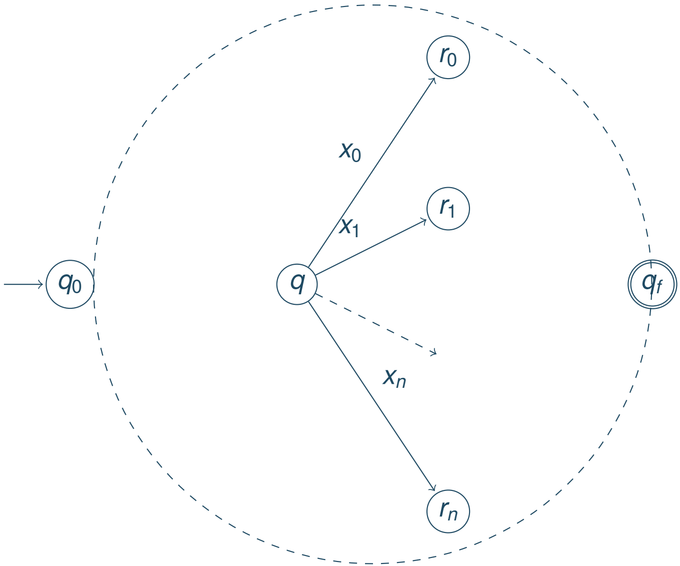
Soit $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ un automate fini déterministe.

Nous avons défini pour chaque état $q \in Q$ de $\mathcal{A}$, le langage $L_q = \{v \in \Sigma^* \mid \delta^*(q, v) \in F\}$ (lors de la minimisation des automates)

➥ en particulier $L_{q_0} = \mathcal{L}(\mathcal{A})$

Pour tout $q \in Q$, on note x_i, pour $0 \leq i \leq n$ les lettres de Σ telles qu'il existe une transition $\delta(q, x_i)$ et on note r_i l'état d'arrivée de celle-ci

Exemple



Équation et langage

Soit $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ un automate fini déterministe.

Nous avons défini pour chaque état $q \in Q$ de $\mathcal{A}$, le langage $L_q = \{v \in \Sigma^* \mid \delta^*(q, v) \in F\}$ (lors de la minimisation des automates)

➥ en particulier $L_{q_0} = \mathcal{L}(\mathcal{A})$

Pour tout $q \in Q$, on note x_i, pour $0 \leq i \leq n$ les lettres de Σ telles qu'il existe une transition $\delta(q, x_i)$ et on note r_i l'état d'arrivée de celle-ci

Pour $u \in L_q$

- si $q \notin F$, $\exists x_i$ tel que $u = x_i v$ avec v un mot de L_{r_i}
- si $q \in F$, $u = \varepsilon$ ou $\exists x_i$ tel que $u = x_i v$ avec v un mot de L_{r_i}

Donc

- si $q \notin F$, $L_q = x_0 L_{r_0} \cup \cdots \cup x_n L_{r_n}$
- si $q \in F$, $L_q = x_0 L_{r_0} \cup \cdots \cup x_n L_{r_n} \cup \{\varepsilon\}$

Système d'équations d'un automate

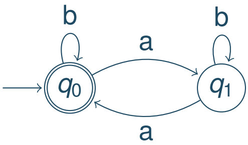
Si on note X_q **l'expression rationnelle d'un état** q, nous obtenons pour L_q une expression rationnelle définie par l'équation :

$$\begin{aligned} X_q = x_0 X_{r_0} + \ldots x_n X_{r_n} &\quad \text{si } q \notin F \\ X_q = x_0 X_{r_0} + \ldots x_n X_{r_n} + \varepsilon &\quad \text{si } q \in F \end{aligned}$$

Système d'équations d'un automate

Le système d'équations d'un automate est le système composé des équations de chacun des états de l'automate

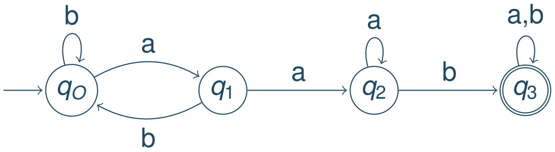
Exemple - Nombre pair de a - Équation



Système d'équations

$$
\begin{aligned}
X_0 &= aX_1 + bX_0 + \varepsilon \\
X_1 &= aX_0 + bX_1
\end{aligned}
$$

Exemple - *aab* facteur - Équation



Système d'équations

$$
\begin{aligned}
X_0 &= aX_1 + bX_0 \\
X_1 &= aX_2 + bX_0 \\
X_2 &= aX_2 + bX_3 \\
X_3 &= aX_3 + bX_3 + \varepsilon
\end{aligned}
$$

Lemme d'Arden

Lemme d'Arden Soient $A, B \subseteq \Sigma^*$ tels que $\varepsilon \notin A$, l'équation $X = AX \cup B$ admet A^*B comme unique solution

Preuve Il faut montrer que A^*B est une solution est qu'elle est unique.

(1) Montrons que A^*B **est une solution** :

$$\begin{array}{rcl} X & = & AX \cup B \\ & = & A(A^*B) \cup B \\ & = & A^+B \cup B \\ & = & (A^+ \cup \{\varepsilon\})B \\ & = & A^*B \end{array}$$

Lemme d'Arden - Preuve (suite)

(2) Supposons qu'il existe une autre solution $\mathcal{L}$, alors par exemple

$$\begin{aligned} \mathcal{L} &= A\mathcal{L} \cup B = A(A\mathcal{L} \cup B) \cup B = A(A(A\mathcal{L} \cup B) \cup B) \cup B \\ &= A(A^2\mathcal{L} \cup AB \cup B) \cup B \\ &= A^3\mathcal{L} \cup A^2B \cup AB \cup B \end{aligned}$$

ceci peut s'étendre pour tout $n > 0$

$$\mathcal{L} = A^{n+1}\mathcal{L} \cup A^nB \cup A^{n-1}B \cup \cdots \cup AB \cup B$$

comme $\forall n \geq 0, A^nB$, $A^*B \subseteq \mathcal{L}$ donc A^*B est **la plus petite solution**

(3) Si $\varepsilon \notin A$, alors A^*B est **l'unique solution**. Supposons qu'il existe une solution $\mathcal{L} \neq \{\varepsilon\}$ et telle que $\mathcal{L} \neq A^*B$. Alors $\exists u \in \mathcal{L}$ avec $|u| = n \neq 0$, en remplaçant n fois $\mathcal{L}$ dans l'équation, on obtient $\mathcal{L} = A^{n+1}\mathcal{L} \cup A^nB \cup A^{n-1}B \cup \cdots \cup AB \cup B$. Comme $\varepsilon \notin A$, tout mot de A^{n+1} a une longueur strictement supérieure à n, donc $u \in A^nB \cup A^{n-1}B \cup \cdots \cup AB \cup B$ donc $u \in A^*B$ ce qui contredit l'hypothèse

Résolution du système d'équations

Le système a n équations avec $n = |Q|$ et n inconnues et au moins un état final, et les équations ont la forme :

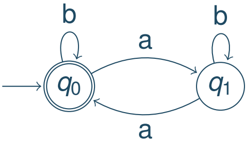
$$X_q = \begin{cases} \bigcup_{(q,a,q')\in\delta} aX_{q'} & \text{si } q \notin F \\ \bigcup_{(q,a,q')\in\delta} aX_{q'} + \varepsilon & \text{si } q \in F \end{cases}$$

Le système se résout par subsitutions comme dans la méthode de Gauss, on peut montrer par récurrence sur n que le système a une solution si l'automate a un état final. Idées : on choisit un X_q

- s'il existe une équation de forme $X_q = \mathcal{L}_1 X_q + \mathcal{L}_2$ avec $\mathcal{L}_1, \mathcal{L}_2 \in \Sigma^*$, on applique le lemme d'Arden
- sinon on choisit une des équations qui définit un X_q et si X_q dépend de X_q, on applique le lemme d'Arden

on remplace X_q dans les autres équations et on obtient un système de même forme de $n-1$ équations à $n-1$ inconnues

Exemple - Nombre pair de a - Résolution



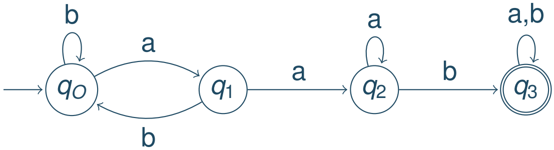
Système d'équations

$$\begin{aligned} X_0 &= aX_1 + bX_0 + \varepsilon \\ X_1 &= aX_0 + bX_1 \end{aligned}$$

Résolution

$$\begin{aligned} X_1 &= b^*(aX_0) \\ X_0 &= a(b^*(aX_0)) + bX_0 + \varepsilon \\ &= (ab^*a + b)X_0 + \varepsilon \\ &= (ab^*a + b)^* \end{aligned}$$

Exemple - *aab* facteur - Résolution



Système d'équations

$$\begin{aligned}
X_0 &= aX_1 + bX_0 \\
X_1 &= aX_2 + bX_0 \\
X_2 &= aX_2 + bX_3 \\
X_3 &= aX_3 + bX_3 + \varepsilon \\
&= (a+b)X_3 + \varepsilon
\end{aligned}$$

Résolution

$$\begin{aligned}
X_3 &= (a+b)^* \\
X_2 &= aX_2 + b(a+b)^* \\
&= a^*b(a+b)^* \\
X_0 &= aX_1 + bX_0 \\
&= b^*aX_1 \\
X_1 &= aa^*b(a+b)^* + bb^*aX_1 \\
&= (bb^*a)^*aa^*b(a+b)^* \\
&= (b^+a)^*a^+b(a+b)^* \\
X_0 &= b^*a(b^+a)^*a^+b(a+b)^*
\end{aligned}$$

Théorème de Kleene

Algorithme de Brzozowski et McCluskey

Automates généralisés

L'algorithme utilise une forme généralisée des automates finis non déterministes avec une différence majeure :

- les transitions sont étiquetées par des expressions rationnelles

par ailleurs cet automate a

- un seul état initial sans transition entrante et un seul état final sans transition sortante

Un mot u est reconnu par l'automate, s'il existe un chemin de l'état initial à l'état final étiqueté par les expressions rationnelles $e_0, \ldots, e_n$ telles que $u \in e_0.e_1 \ldots e_n$

Le langage reconnu par l'automate est l'ensemble des mots reconnus

Transformations préalables

Soit $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ l'automate dont on veut calculer l'expression. Il peut être facilement transformé en automate généralisé $\mathcal{A}' = (E, Q \cup \{q_i, q_f\}, \{q_i\}, \{q_f\}, \delta')$ avec :

- E l'ensemble des expressions rationnelles sur Σ
- $\delta' = \delta \cup \{(q_i, \varepsilon, q') \mid q' \in I\} \cup \{(q', \varepsilon, q_f) \mid q' \in F\}$

L'automate $\mathcal{A}'$ est ensuite tranformé en un automate $\mathcal{A}''$ pour lequel il existe au plus une transition entre deux états

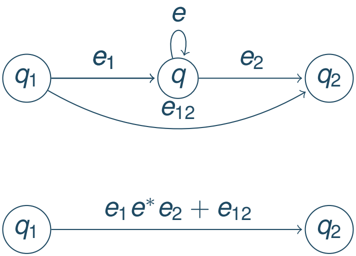
- $\delta'' = \{(q, \Sigma_{(q,x,q') \in \delta'} x, q') \mid q, q' \in Q\}$

$$q \xrightarrow{\quad x_0, x_1, \ldots x_n \quad} q'$$

$$q \xrightarrow{x_0 + x_1 + \cdots + x_n} q'$$

Algorithme - Illustration

Ensuite l'algorithme va ensuite supprimer un à un les états q qui ne sont ni initiaux ni finaux de la manière suivante



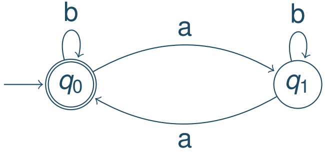
Algorithme

Pour tout état q tel que $q \neq q_i$ et $q \neq q_f$:

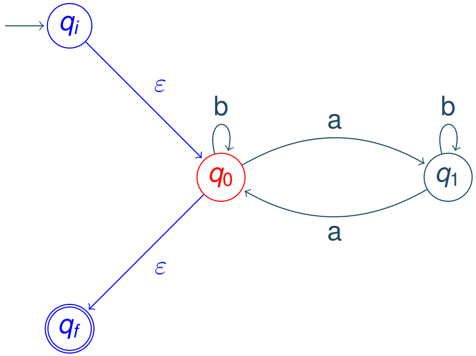
- pour toute paire d'états q_1 et q_2 tels que $(q_1, e_1, q) \in \delta''$ et $(q, e_2, q_2) \in \delta''$ (attention q_1 et q_2 peuvent être égaux) :
  - soit e telle que $(q, e, q) \in \delta''$ ou $e = \varepsilon$
  - soit e_{12} telle que $(q_1, e_{12}, q_2) \in \delta''$ ou $e_{12} = \varepsilon$
  - ajouter une transition $(q_1, e_1 e^* e_2 + e_{12}, q_2) \in \delta''$
- supprimer q et les transitions impliquant q

L'algorithme s'arrête lorsqu'il ne reste plus que l'état initial et l'état final et une seule transition entre les deux étiquetée par l'expression rationnelle du langage reconnu par $\mathcal{A}$

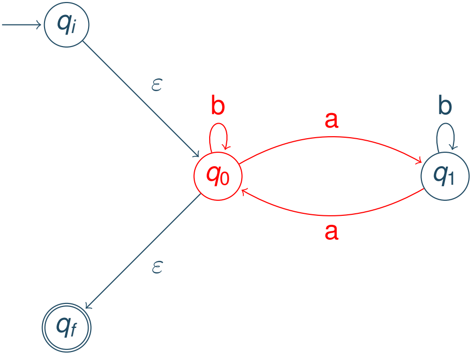
Exemple - Nombre pair de a - Algorithme

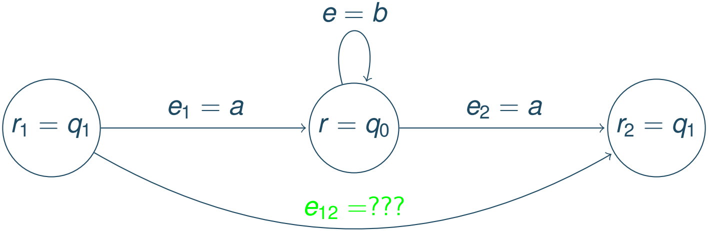


Exemple - Nombre pair de a - Algorithme

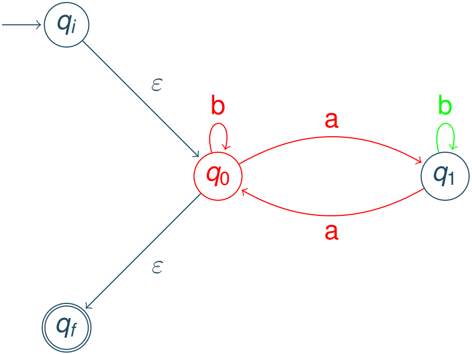


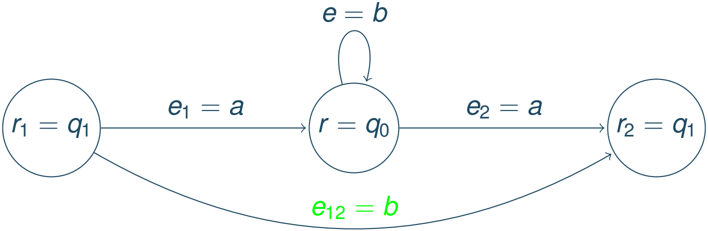
Exemple - Nombre pair de a - Algorithme



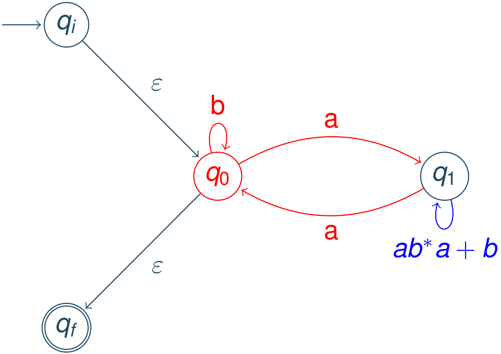


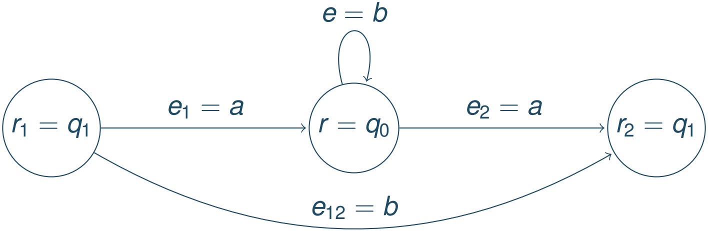
Exemple - Nombre pair de a - Algorithme



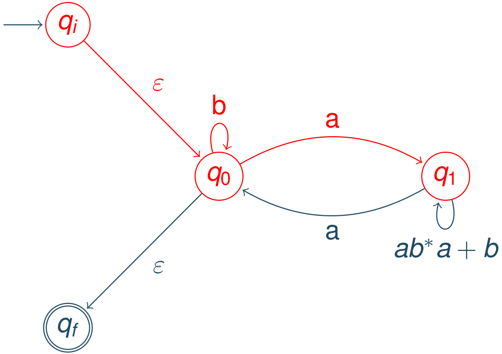


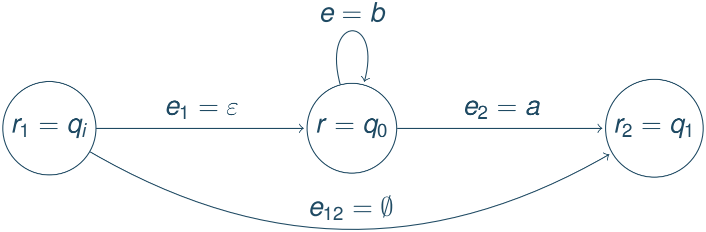
Exemple - Nombre pair de a - Algorithme



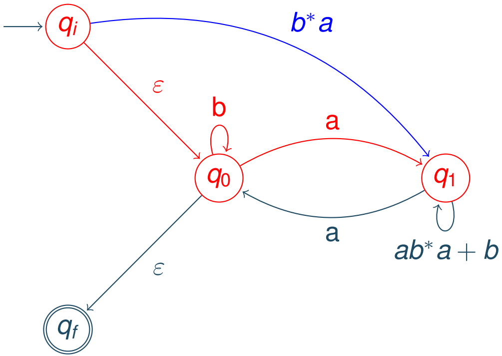


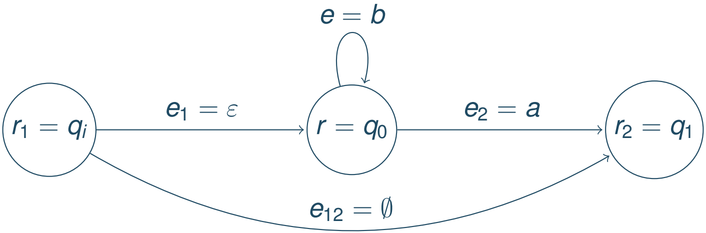
Exemple - Nombre pair de a - Algorithme



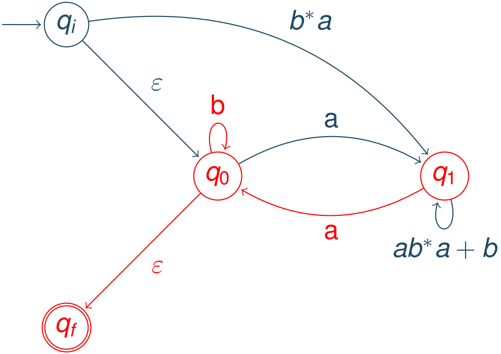


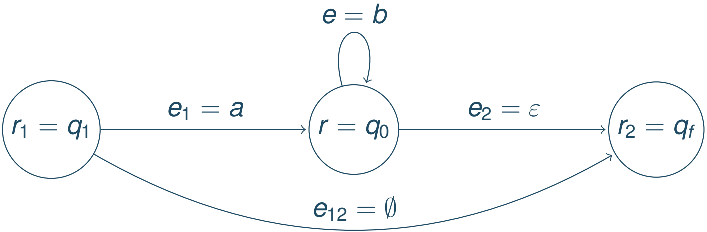
Exemple - Nombre pair de a - Algorithme



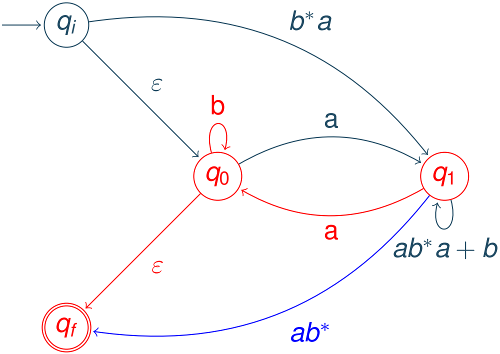


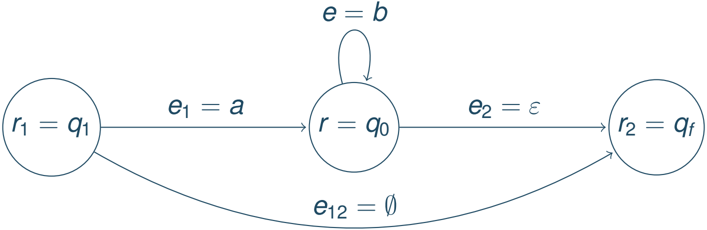
Exemple - Nombre pair de a - Algorithme



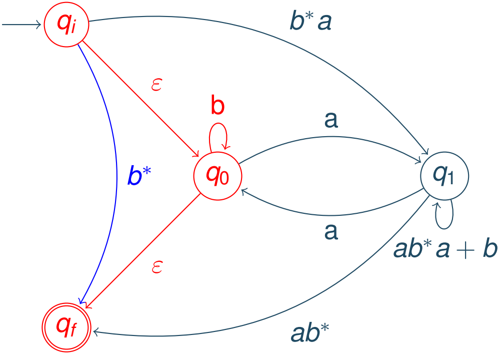


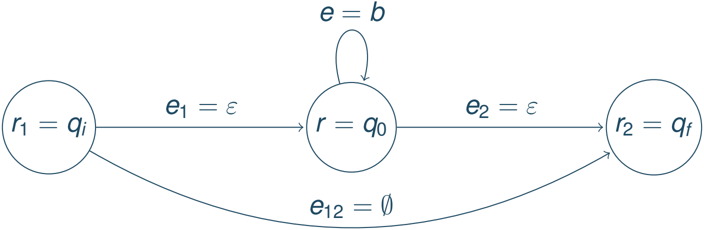
Exemple - Nombre pair de a - Algorithme



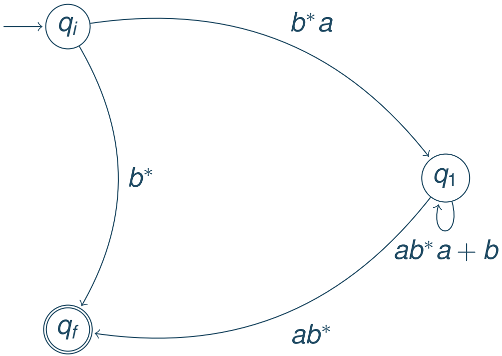


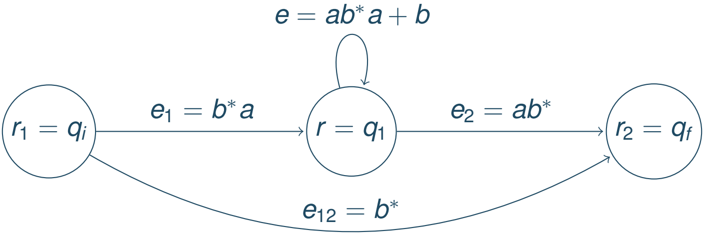
Exemple - Nombre pair de a - Algorithme



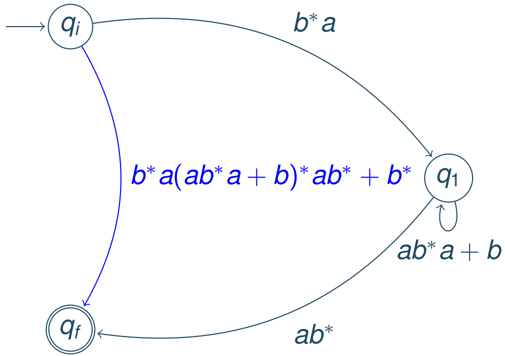


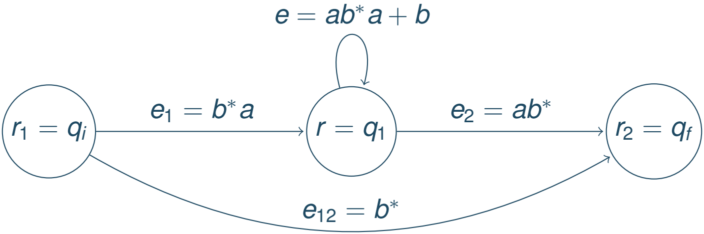
Exemple - Nombre pair de a - Algorithme



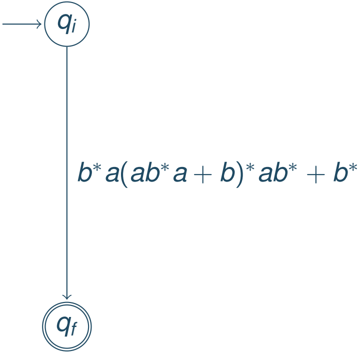


Exemple - Nombre pair de a - Algorithme





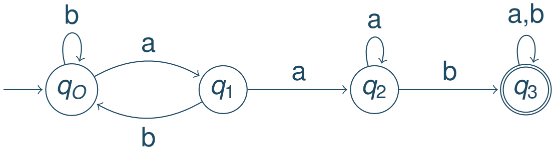
Exemple - Nombre pair de a - Algorithme



➥L'expression rationnelle calculée pour l'automate de départ est $b^*a(ab^*a+b)^*ab^*+b^*$

Exemple - *aab* facteur - Équation

Autre exemple à faire en exercice



Résumé

Nous savons à présent que les classes des langages reconnaissables et des langages rationnels sont les mêmes, et nous savons passer d'une représentation à l'autre

- Pour montrer d'un langage est reconnaissable / rationnel, nous pouvons produire une expression rationnelle ou un automate
- Comment prouver qu'un langage ne l'est pas ?

  ➥Nous allons voir une condition nécessaire (mais pas suffisante) pour qu'un langage soit reconnaissable qui permettra de montrer par l'absurde qu'un langage ne l'est pas

Idée du Lemme de l'étoile

Aussi appelé lemme de pompage.

Idée. Lorsqu'un mot d'un langage reconnaissable est assez long, il contient un motif qui se répète, autrement dit :

- le pouvoir d'expression des automates (ou expressions rationnelles) est limité
- un « morceau » du mot correspond à une itération (une « étoile ») ou peut être « pompé »

Lemme de l'étoile

Lemme de l'étoile Soit $\mathcal{L}$ un langage reconnaissable. Il existe un entier k tel que pour tout $u \in \mathcal{L}$, tel que $|u| \geq k$ alors :

- $u = vwv'$ avec $0 < |w| \leq k$
- $\forall i \in \mathbb{N}, vw^iv' \in \mathcal{L}$

Preuve. Soit $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ un DFA tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}$ avec $|Q| = k$

Soit $u = x_0x_1 \dots x_{n-1} \in \mathcal{L}$ avec $\forall 0 \leq i < n$, $x_i \in \Sigma$, il existe un calcul dans $\mathcal{A}$:

$$q_0 \xrightarrow{x_0} q_1 \xrightarrow{x_1} q_2 \xrightarrow{x_2} \dots \xrightarrow{x_{n-1}} q_n \in F$$

Lemme de l'étoile - Preuve

Si $n \geq k$, il existe $0 \leq i < j \leq n$ tels que $q_i = q_j$, le calcul suivant est un calcul de $\mathcal{A}$

$$q_0 \xrightarrow{x_0} q_1 \xrightarrow{x_1} \ldots \xrightarrow{x_{i-1}} q_i = q_j \xrightarrow{x_j} \ldots \xrightarrow{x_{n-1}} q_n$$

ainsi que tout calcul répétant le chemin de q_i à q_j un nombre arbitraire de fois :

$$q_0 \xrightarrow{x_0} q_1 \xrightarrow{x_1} \ldots \xrightarrow{x_{i-1}} q_i \xrightarrow{x_i} \ldots \xrightarrow{x_{j-1}} q_j = q_i \xrightarrow{x_i} \ldots \xrightarrow{x_{j-1}} q_j \xrightarrow{x_j} \ldots \xrightarrow{x_{n-1}} q_n$$

Posons $v = x_0 x_1 \ldots x_{i-1}$, $w = x_i \ldots x_{j-1}$, $v' = x_j x_{j+1} \ldots x_{n-1}$, nous obtenons $\forall i \in \mathbb{N}, vw^i v' \in \mathcal{L}$

Lemme de l'étoile - Utilisation

Montrons que $a^n b^n$ n'est pas rationnel

Par l'absurde, si $a^n b^n$ est rationnel, il existe un entier k donné par le lemme de l'étoile, considérons $a^k b^k$. D'après le lemme de l'étoile il existe $vwv' = a^k b^k$ tel que $0 < |w| \leq k$ et $vw^* v' \subseteq a^n b^n$ alors soit :

- $w \in a^+$ et dans ce cas $vv' = a^i b^k$ avec $i < k$ et donc $vv' \notin a^n b^n$. Contradiction
- $w \in b^+$, idem
- $w = a^i b^j$ avec $i, j > 0$ et dans ce cas $vwwv' = a^{i'} a^i b^j a^i b^j b^{j'} \notin a^n b^n$. Contradiction

Application à la décision

Proposition

Soit $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ un automate fini tel que $|Q| = k$ alors :

1. $(\mathcal{L}(\mathcal{A}) \neq \emptyset) \Leftrightarrow (\exists u \in \mathcal{L}(\mathcal{A})$ tel que $|u| < k)$
2. $(\mathcal{L}(\mathcal{A})$ est infini$) \Leftrightarrow (\exists u \in \mathcal{L}(\mathcal{A})$ tel que $k \leq |u| < 2k)$

Algorithme

- Pour tester si le langage reconnu par un automate est vide, il suffit de tester tous les mots $u \in \Sigma^*$ tels que $|u| < |Q|$, ce qui revient en pratique à tester tous les mots de longueur $|Q| - 1$ soit $|\Sigma|^{|Q|-1}$
- Pour tester si le langage reconnu par un automate est fini (ou infini), il suffit de tester tous les mots $u \in \Sigma^*$ tels que $|Q| \leq |u| < 2|Q|$ soit en pratique tous les mots de longueur $2|Q| - 1$, soit $|\Sigma|^{2|Q|-1}$

Application à la décision - Preuves

Preuve du 1. $(\mathcal{L}(\mathcal{A}) \neq \emptyset) \Leftrightarrow (\exists u \in \mathcal{L}(\mathcal{A})$ tel que $|u| < k)$

L'implication $\Leftarrow$ est évidente. Si $\mathcal{L}(\mathcal{A}) \neq \emptyset$, notons u le plus petit mot de $\mathcal{L}(\mathcal{A})$ et supposons que $|u| \geq k$. Le lemme de l'étoile nous permet de déduire qu'il existe $u = vwv'$ avec $0 < |w| \leq k$ et $\forall i \in \mathbb{N}, vw^i v' \in \mathcal{L}(\mathcal{A})$, donc $vv' \in \mathcal{L}(\mathcal{A})$ et $|vv'| < |u|$. Contradiction

Preuve du 2. $(\mathcal{L}(\mathcal{A})$ est infini$) \Leftrightarrow (\exists u \in \mathcal{L}(\mathcal{A})$ tel que $k \leq |u| < 2k)$

L'implication $\Leftarrow$ s'obtient en appliquant le lemme de pompage à u et en montrant que $\mathcal{L}(\mathcal{A})$ contient donc un langage de la forme vw^*v'. Si $\mathcal{L}(\mathcal{A})$ est infini, il contient des mot de longueur arbitrairement grande. Considérons u le plus petit mot de $\mathcal{L}(\mathcal{A})$ tel que $|u| \geq k$. Si $|u| < 2k$, u vérifie $k \leq |u| < 2k$, sinon par le lemme de l'étoile, $u = vwv'$ avec $1 \leq |w| \leq k$ et $vv' \in \mathcal{L}(\mathcal{A})$ avec $|vv'| \leq |u|$ et $|vv'| \geq k$ et donc u n'est pas le plus petit mot tel que $k \leq |u|$. Contradiction