Langages formels

Paul Gastin / Stefan Schwoon

{Paul.Gastin,Stefan.Schwoon}@ens-paris-saclay.fr

http://www.lsv.fr/~gastin/Langages/

L3 Informatique Cachan
2020-2021

Organisation

Définition : Emploi de temps

- Cours: jeudi 13:30 – 15:30 (Stefan Schwoon / Paul Gastin)
- TD: mercredi 14:00 – 17:00 (Émilie Grienenberger / Mathieu Hilaire)

Remarque : Modalités de validation

- Contrôle continu 1 + 2
- TP sur l'analyse syntaxique
- Examens 1 + 2
- tous à poids égal

Plan

Introduction

Langages reconnaissables

Grammaires

Langages algébriques

Automates à pile

Analyse syntaxique

Fonctions séquentielles

Automates d'arbres

Motivations

Définition :

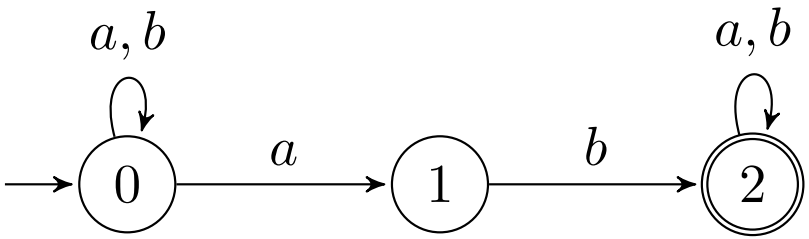
1. Description et analyse (lexicale et syntaxique) des langages (programmation, naturels, ...)
2. Modèles de calcul
3. Abstractions mathématiques simples de phénomènes complexes dans le but de
   - Prouver des propriétés.
   - Concevoir des algorithmes permettant de tester des propriétés ou de résoudre des problèmes.
4. Types de données

Modèles de calcul

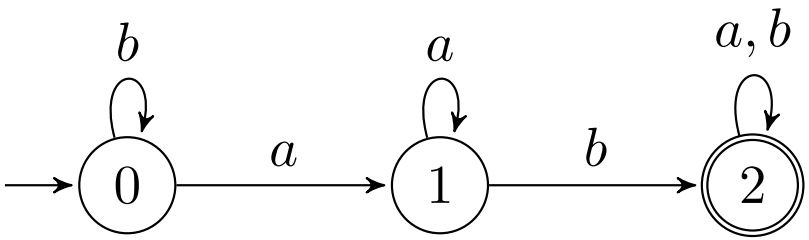
Exemple: Les mots sur $\Sigma := \{a, b\}$ qui contiennent un facteur ab.

Expression régulière : $\Sigma^* ab \Sigma^*$

Automate non-déterministe :



Automate déterministe :



Modèles de calcul

Exemple: Les mots sur $\Sigma := \{a, b\}$ qui contiennent un facteur ab.

Formule logique :

$$\exists p : a(p) \wedge b(p+1)$$

Homomorphisme vers un monoïde :

- $\phi(a) := A$, $\phi(b) := B$, $\phi(uv) = \phi(u) \circ \phi(v)$

$u \setminus v$	A	B	C	D	E
A	A	C	C	C	A
B	D	B	C	D	B
C	C	C	C	C	C
D	D	C	C	C	D
E	A	B	C	D	E

$A \mathrel{\widehat{=}} a^+, \quad B \mathrel{\widehat{=}} b^+, \quad C \mathrel{\widehat{=}} \Sigma^* ab \Sigma^*, \quad D \mathrel{\widehat{=}} b^+ a^+, \quad E \mathrel{\widehat{=}} \varepsilon$

Principales Références

[7] Olivier Carton.
Langages formels, calculabilité et complexité.
Vuibert, 2008.

[9] John E. Hopcroft et Jeffrey D. Ullman.
Introduction to automata theory, languages and computation.
Addison-Wesley, 1979.

[10] Dexter C. Kozen.
Automata and Computability.
Springer, 1997.

[13] Jacques Sakarovitch.
Éléments de théorie des automates.
Vuibert informatique, 2003.

[8] Hubert Comon, Max Dauchet, Remi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, Marc Tommasi.
Tree Automata Techniques and Applications.
http://www.grappa.univ-lille3.fr/tata/

Autres Références

[1] Alfred V. Aho, Ravi Sethi et Jeffrey D. Ullman.
Compilers: principles, techniques and tools.
Addison-Wesley, 1986.

[2] Alfred V. Aho et Jeffrey D. Ullman.
The theory of parsing, translation, and compiling. Volume I: Parsing.
Prentice-Hall, 1972.

[3] Luc Albert, Paul Gastin, Bruno Petazzoni, Antoine Petit, Nicolas Puech et Pascal Weil.
Cours et exercices d'informatique.
Vuibert, 1998.

[4] Jean-Michel Autebert.
Théorie des langages et des automates.
Masson, 1994.

Autres Références

[5] Jean-Michel Autebert, Jean Berstel et Luc Boasson.
Context-Free Languages and Pushdown Automata.
Handbook of Formal Languages, Vol. 1, Springer, 1997.

[6] Jean Berstel.
Transduction and context free languages.
Teubner, 1979.

[11] Jean-Éric Pin.
Automates finis et applications.
Polycopié du cours à l'École Polytechnique, 2004.

[12] Grzegorz Rozenberg et Arto Salomaa, éditeurs.
Handbook of Formal Languages,
Vol. 1, Word, Language, Grammar,
Springer, 1997.

[14] Jacques Stern.
Fondements mathématiques de l'informatique.
Mc Graw Hill, 1990.

Plan

Introduction

Langages reconnaissables

- Mots
- Langages
- Automates déterministes
- Automates non déterministes
- Automates avec ε-transitions
- Propriétés de fermeture
- Langages rationnels
- Critères de reconnaissabilité
- Minimisation
- Logique MSO
- Morphismes et congruences

Grammaires

Langages algébriques

Bibliographie

[4] Jean-Michel Autebert.
Théorie des langages et des automates.
Masson, 1994.

[7] Olivier Carton.
Langages formels, calculabilité et complexité.
Vuibert, 2008.

[9] John E. Hopcroft et Jeffrey D. Ullman.
Introduction to automata theory, languages and computation.
Addison-Wesley, 1979.

[10] Dexter C. Kozen.
Automata and Computability.
Springer, 1997.

[13] Jacques Sakarovitch.
Éléments de théorie des automates.
Vuibert informatique, 2003.

Mots

A ou Σ : alphabet (ensemble fini).
$u \in \Sigma^*$: mot $=$ suite finie de lettres.

$\cdot$: concaténation associative.
ε ou 1 : mot vide, neutre pour la concaténation.
$(\Sigma^*, \cdot)$: monoïde libre engendré par Σ.

$|u|$: longueur du mot u.
$|\cdot| : \Sigma^* \to \mathbb{N}$ est le morphisme défini par $|a| = 1$ pour $a \in \Sigma$.
$|u|_a$: nombre de a dans le mot u.

$\tilde{u}$: miroir du mot u.

Mots

Ordres partiels :

- u préfixe de v si $\exists u'$, $v = uu'$
- u suffixe de v si $\exists u'$, $v = u'u$
- u facteur de v si $\exists u', u''$, $v = u'uu''$
- u sous-mot de v si $v = v_0u_1v_1u_2 \cdots u_nv_n$ avec $u_i, v_i \in \Sigma^*$ et $u = u_1u_2 \cdots u_n$

Théorème : Higman

L'ordre sous-mot est un *bon* ordre, i.e.
(de toute suite infinie on peut extraire une sous-suite infinie croissante)
(ou tout ensemble de mots a un nombre fini d'éléments minimaux)

Langages

Langage $=$ sous-ensemble de Σ^*.
Exemples.

Opérations sur les langages : soient $K, L \subseteq \Sigma^*$

Ensemblistes : union, intersection, complément, différence, ...

Concaténation : $K \cdot L = \{u \cdot v \mid u \in K \text{ et } v \in L\}$
La concaténation est associative et distributive par rapport à l'union.
$|K \cdot L| \leq |K| \cdot |L|$
notion de multiplicité, d'ambigüité

Langages

Itération : $L^0 = \{\varepsilon\}$, $L^{n+1} = L^n \cdot L = L \cdot L^n$,
$L^* = \bigcup_{n \geq 0} L^n$, $L^+ = \bigcup_{n > 0} L^n$.
Exemples : Σ^n, Σ^*, $(\Sigma^2)^*$.

Quotients : $K^{-1} \cdot L = \{v \in \Sigma^* \mid \exists u \in K, u \cdot v \in L\}$
$L \cdot K^{-1} = \{u \in \Sigma^* \mid \exists v \in K, u \cdot v \in L\}$

Automates déterministes

Définition : Automate déterministe

$\mathcal{A} = (Q, \delta, i, F)$
Q ensemble *fini* d'états, $i \in Q$ état initial, $F \subseteq Q$ états finaux,
$\delta : Q \times \Sigma \to Q$ fonction de transition (totale ou partielle).

Exemples.

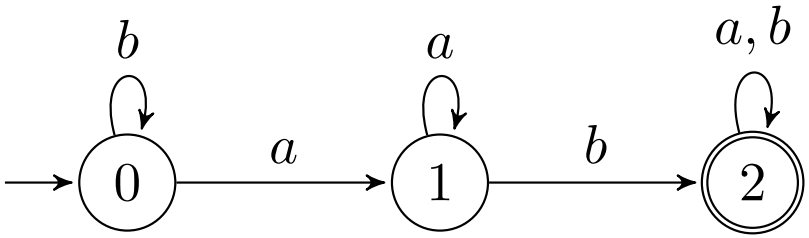
Calcul de $\mathcal{A}$ sur un mot $u = a_1 \cdots a_n$: $q_0 \xrightarrow{u} q_n$

$$q_0 \xrightarrow{a_1} q_1 \cdots q_{n-1} \xrightarrow{a_n} q_n$$

avec $q_i = \delta(q_{i-1}, a_i)$ pour tout $0 < i \leq n$.

Généralisation de δ à $Q \times \Sigma^*$:
$\delta(q, \varepsilon) = q$,
$\delta(q, u \cdot a) = \delta(\delta(q, u), a)$ si $u \in \Sigma^*$ et $a \in \Sigma$.

Automate déterministe: Exemple



- $Q = \{0, 1, 2\}$
- $\delta(0, a) = 1,\ \delta(0, b) = 0,\ \delta(1, a) = 1,$
  $\delta(1, b) = 2,\ \delta(2, a) = 2,\ \delta(2, b) = 2$
- $\iota = 0$
- $F = \{2\}$

Calcul: $0 \xrightarrow{b} 0 \xrightarrow{a} 1 \xrightarrow{a} 1 \xrightarrow{b} 2 \xrightarrow{a} 2$

Généralisation: $\delta(1, abba) = 2$

Automates déterministes

Langage accepté (reconnu) par $\mathcal{A}$: $\mathcal{L}(\mathcal{A}) = \{u \in \Sigma^* \mid \delta(i,u) \in F\}$.

Exemples.

Définition : Reconnaissables

Un langage $L \subseteq \Sigma^*$ est *reconnaissable*, s'il existe un automate fini $\mathcal{A}$ tel que $L = \mathcal{L}(\mathcal{A})$.

On note $\mathrm{Rec}(\Sigma^*)$ la famille des langages reconnaissables sur Σ^*.

Exemple de langages reconnaissables

Les entiers divisibles par $d \geq 2$, exprimés à la base $b \geq 1$.

- $Q = \{\iota\} \uplus \{0, \ldots, d-1\}$
- $F = \{0\}$
- $\delta(\iota, j) = j \bmod d$ pour tout $j = 0, \ldots, b-1$
- $\delta(i, j) = (i \cdot b + j) \bmod d$ pour tout $i = 0, \ldots, d-1$, $j = 0, \ldots, b-1$

Automates non déterministes

Exemple : automate non déterministe pour $\Sigma^* \cdot \{aba\}$

Définition : Automate non déterministe

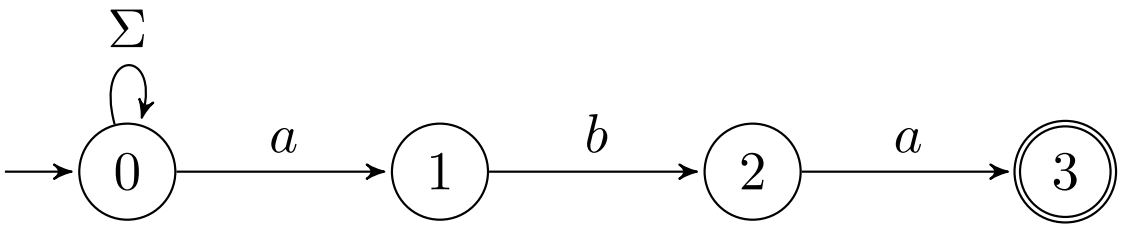
$\mathcal{A} = (Q, T, I, F)$
Q ensemble *fini* d'états, $I \subseteq Q$ états initiaux, $F \subseteq Q$ états finaux,
$T \subseteq Q \times \Sigma \times Q$ ensemble des transitions.
On utilise aussi $\delta : Q \times \Sigma \to 2^Q$.

Calcul de $\mathcal{A}$ sur un mot $u = a_1 \cdots a_n$: $q_0 \xrightarrow{a_1} q_1 \cdots q_{n-1} \xrightarrow{a_n} q_n$ avec $(q_{i-1}, a_i, q_i) \in T$ pour tout $0 < i \leq n$.
Langage accepté (reconnu) par $\mathcal{A}$:

$$\mathcal{L}(\mathcal{A}) = \{u \in \Sigma^* \mid \exists\, i \xrightarrow{u} f \text{ calcul de } \mathcal{A} \text{ avec } i \in I \text{ et } f \in F\}.$$

Automate nondéterministe: Exemple

Automate ND acceptant $\Sigma^* \cdot aba$, pour $\Sigma = \{a, b\}$.



- $Q = \{0, 1, 2, 3\}$
- $T = \{\langle 0, a, 0\rangle, \langle 0, b, 0\rangle, \langle 0, a, 1\rangle, \langle 1, b, 2\rangle, \langle 2, a, 3\rangle\}$
- $I = \{0\}$
- $F = \{3\}$

Avec δ généralisé : $\delta(\{0\}, aba) = \{0, 1, 3\}$, $\delta(\{1\}, a) = \emptyset$

Automates non déterministes

Théorème : Déterminisation

Soit $\mathcal{A}$ un automate non déterministe. On peut construire un automate déterministe $\mathcal{B}$ qui reconnaît le même langage ($\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$).

Preuve

Automate des parties

Exemple : automate déterministe pour $\Sigma^* \cdot \{aba\}$

On appelle déterminisé de $\mathcal{A}$ l'automate des parties *émondé*.

Exercices :

1. Donner un automate non déterministe avec n états pour $L = \Sigma^* a \Sigma^{n-2}$.
2. Montrer que tout automate déterministe reconnaissant ce langage L a au moins 2^{n-1} états.
3. Donner un automate non déterministe à n états tel que tout automate déterministe reconnaissant le même langage a au moins $2^n - 1$ états.

Automate des parties

Soit $\mathcal{A} = \langle Q, T, I, F \rangle$ un automate ND sur Σ.
Construisons $\mathcal{A}' = \langle Q', \delta', \iota', F' \rangle$ déterministe avec $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$.
Mettons $Q' := 2^Q$. On garde l'invariante suivante pour tout $w \in \Sigma^*$:

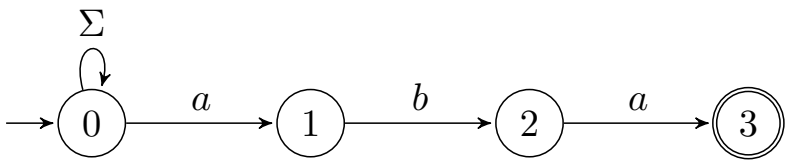
$$\delta'(\iota', w) = \{\, q \in Q \mid \exists \iota \in I : \iota \xrightarrow{w}{}^{*} q \,\}$$

- $\iota' := I$
- $F' := \{\, S \subseteq Q \mid S \cap F \neq \emptyset \,\}$
- $\delta'(S, a) = \{\, q \in Q \mid \exists p \in S : \langle p, a, q \rangle \in T \,\}$

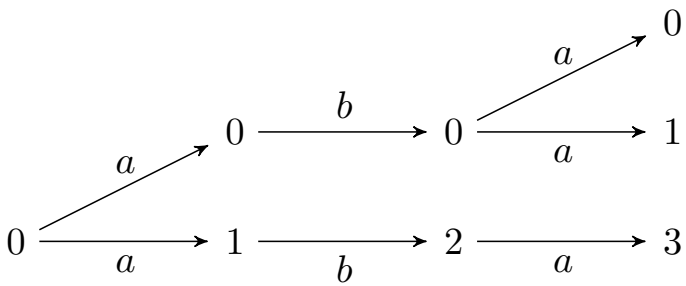
Preuve de correction: montrer l'invariante par récurrence sur $|w|$
Remarque : Il suffit de construire la partie accessible depuis ι'.

Automate des parties: Exemple

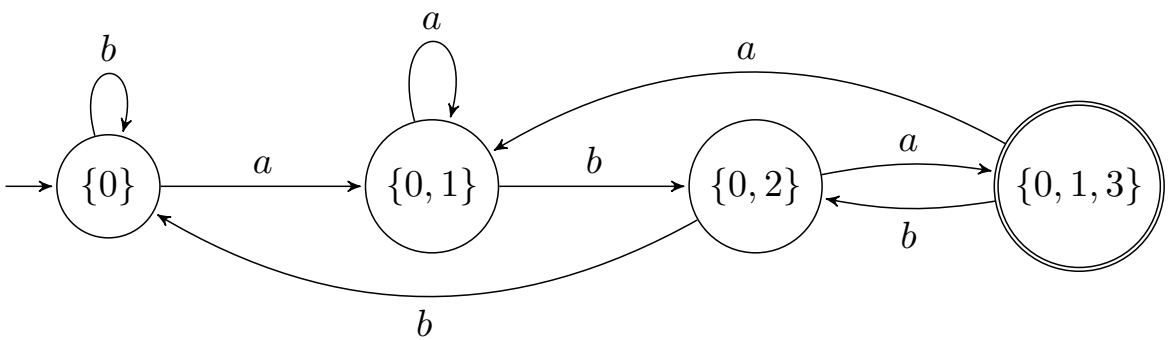
Automate ND pour $\{a, b\}^* aba$:



Calcul sur aba:



Automate des parties:



Automate des parties à la volée

Preuve Problème du mot pour automate ND

Étant donné un automate ND $\mathcal{A}$ et un mot w, on peut vérifier en $\emptyset(|w| \cdot |Q|^2)$ temps et $\emptyset(|Q|)$ espace si $w \in \mathcal{L}(\mathcal{A})$.

Preuve: On simule le calcul du déterminé de $\mathcal{A}$ à la volée.

Automates non déterministes

Un automate (D ou ND) est *complet* si $\forall p \in Q$, $\forall a \in \Sigma$, $\delta(p, a) \neq \emptyset$.

On peut toujours compléter un automate.

Un automate (D ou ND) est émondé si tout état $q \in Q$ est

- accessible d'un état initial : $\exists i \in I$, $\exists u \in \Sigma^*$ tels que $i \xrightarrow{u} q$,
- co-accessible d'un état final : $\exists f \in F$, $\exists u \in \Sigma^*$ tels que $q \xrightarrow{u} f$

On peut calculer l'ensemble $\mathrm{Acc}(I)$ des états accessibles à partir de I et l'ensemble $\mathrm{coAcc}(F)$ des états co-accessibles des états finaux.

Corollaire :

Soit $\mathcal{A}$ un automate.

1. On peut construire $\mathcal{B}$ émondé qui reconnaît le même langage.
2. On peut décider si $\mathcal{L}(\mathcal{A}) = \emptyset$.

Automates avec ε-transitions

Exemple.

Définition : Automate avec ε-transitions

$\mathcal{A} = (Q, T, I, F)$
Q ensemble *fini* d'états, $I \subseteq Q$ états initiaux, $F \subseteq Q$ états finaux,
$T \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ ensemble des transitions.

Un calcul de $\mathcal{A}$ est une suite $q_0 \xrightarrow{a_1} q_1 \cdots q_{n-1} \xrightarrow{a_n} q_n$ avec $(q_{i-1}, a_i, q_i) \in T$ pour tout $0 < i \leq n$.

Ce calcul reconnaît le mot $u = a_1 \cdots a_n$ (les ε disparaissent).

Remarque : Soit $\mathcal{A}$ un automate. On peut construire un automate sans ε-transition $\mathcal{B}$ qui reconnaît le même langage.

Décision

Presque tout est décidable sur les langages reconnaissables donnés par des automates.

Définition :

Problème du vide : étant donné un automate fini $\mathcal{A}$, décider si $\mathcal{L}(\mathcal{A}) = \emptyset$.

Problème du mot : étant donnés un mot $w \in \Sigma^*$ et un automate $\mathcal{A}$, décider si $w \in \mathcal{L}(\mathcal{A})$.

Théorème : vide et mot

Le problème du vide et le problème du mot sont décidables en **NLOGSPACE** pour les langages reconnaissables donnés par automates (déterministe ou non, avec ou sans ε-transitions).

Preuve

C'est de l'accessibilité.

Propriétés de fermeture

Opérations ensemblistes

Proposition :

La famille $\mathrm{Rec}(\Sigma^*)$ est fermée par les opérations ensemblistes (union, complément, ...).

Preuve

Union : construction non déterministe.
Intersection : produit d'automates (préserve le déterminisme).
Complément : utilise la déterminisation.

Corollaire :

On peut décider de l'égalité ou de l'inclusion de langages reconnaissables.
Plus précisément, soient $L_1, L_2 \in \mathrm{Rec}(\Sigma^*)$ donnés par deux automates $\mathcal{A}_1$ et $\mathcal{A}_2$.
On peut décider si $L_1 \subseteq L_2$.

Union et Intersection

Soient $\mathcal{A}_i = \langle Q_i, T_i, I_i, F_i \rangle$, $i = 1, 2$ deux automates ND.
L'automate suivant reconnaît $\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$:

- $\langle Q_1 \uplus Q_2, T_1 \cup T_2, I_1 \cup I_2, F_1 \cup F_2 \rangle$

L'automate suivant (produit) reconnaît $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$:

- $Q := Q_1 \times Q_2$
- $I := I_1 \times I_2$
- $F := F_1 \times F_2$
- $T := \{ \langle \langle p, q \rangle, a, \langle p', q' \rangle \rangle \mid \langle p, a, p' \rangle \in T_1 \wedge \langle q, a, q' \rangle \in T_2 \}$

Propriétés de fermeture

Opérations liées à la concaténation

Proposition :

$\mathrm{Rec}(\Sigma^*)$ est fermée par concaténation et itération.

Concaténation :

Méthode 1 : union disjointe des automates et ajout de transitions.

Méthode 2 : fusion d'états.
On suppose que les automates ont un seul état initial sans transition entrante et un seul état final sans transition sortante.

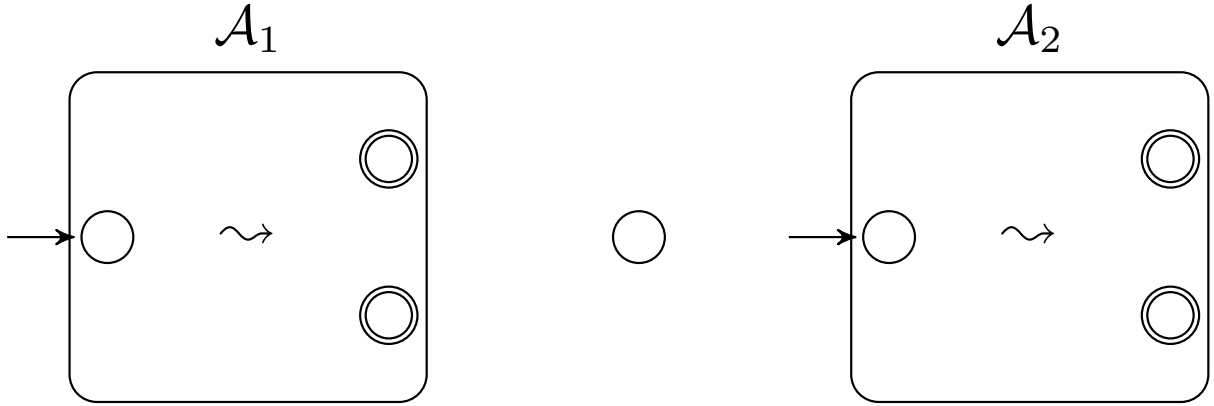
Itération :

Méthode 1 : ajout de transitions. Ajouter un état pour reconnaître le mot vide.

Méthode 2 : ajout d'ε-transitions.

Illustration: Concaténation

- Soient $\mathcal{A}_1, \mathcal{A}_2$ deux automates déterministes.



- L'automate suivant (avec ε) accepte $\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$:

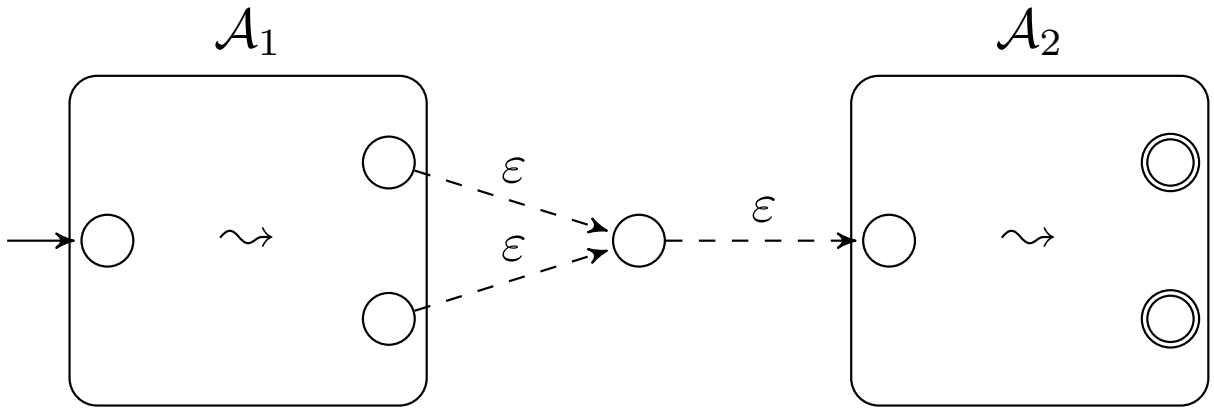
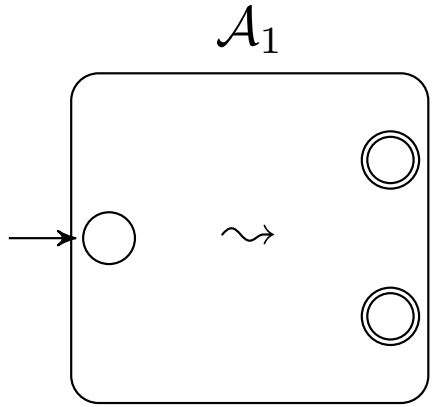
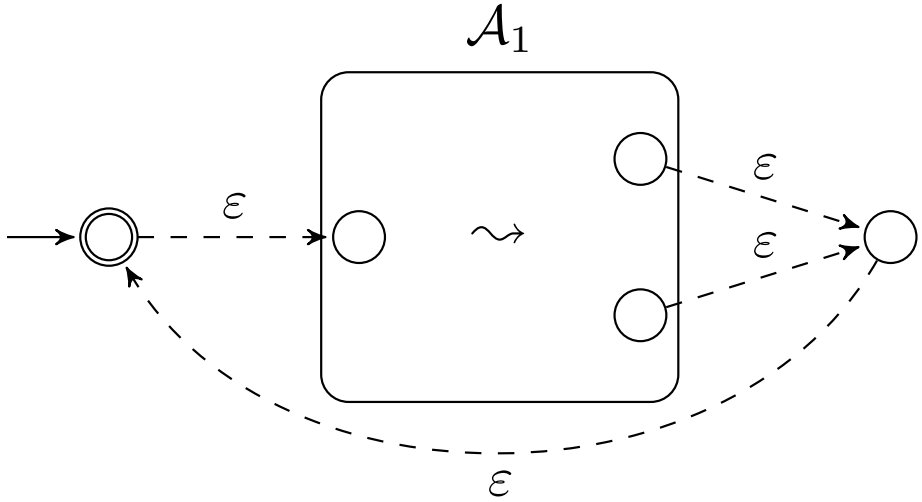


Illustration: Itération

- Soit $\mathcal{A}_1$ un automate déterministe.



- L'automate suivant (avec ε) accepte $\mathcal{L}(\mathcal{A}_1)^*$:



Propriétés de fermeture

Si $L \subseteq \Sigma^*$, on note

- $\mathrm{Pref}(L) = \{u \in \Sigma^* \mid \exists v \in \Sigma^*,\ uv \in L\}$,
- $\mathrm{Suff}(L) = \{v \in \Sigma^* \mid \exists u \in \Sigma^*,\ uv \in L\}$,
- $\mathrm{Fact}(L) = \{v \in \Sigma^* \mid \exists u, w \in \Sigma^*,\ uvw \in L\}$.

Proposition :

$\mathrm{Rec}(\Sigma^*)$ est fermée par préfixe, suffixe, facteur.

Preuve

Modification des états initiaux et/ou finaux.

Propriétés de fermeture

Proposition :

La famille $\mathrm{Rec}(\Sigma^*)$ est fermée par quotients gauches et droits :
Soit $L \in \mathrm{Rec}(\Sigma^*)$ et $K \subseteq \Sigma^*$ arbitraire.
Les langages $K^{-1} \cdot L$ et $L \cdot K^{-1}$ sont reconnaissables.

Preuve

Modification des états initiaux et/ou finaux.

Exercice :

Montrer que si de plus K est reconnaissable, alors on peut effectivement calculer les nouveaux états initiaux/finaux.

Propriétés de fermeture

Morphismes

Soient A et B deux alphabets et $f : A^* \to B^*$ un *morphisme*.

Pour $L \subseteq A^*$, on note $f(L) = \{f(u) \in B^* \mid u \in L\}$.

Pour $L \subseteq B^*$, on note $f^{-1}(L) = \{u \in A^* \mid f(u) \in L\}$.

Proposition :

La famille des langages reconnaissables est fermée par morphisme et morphisme inverse.

1. Si $L \in \mathrm{Rec}(A^*)$ et $f : A^* \to B^*$ est un morphisme alors $f(L) \in \mathrm{Rec}(B^*)$.
2. Si $L \in \mathrm{Rec}(B^*)$ et $f : A^* \to B^*$ est un morphisme alors $f^{-1}(L) \in \mathrm{Rec}(A^*)$.

Preuve

Modification des transitions de l'automate.

Morphisme inverse

Soit $f : A^* \to B^*$ un morphisme, on suppose s.p.d.g. que $A \cap B = \emptyset$.
f est entièrement défini par les $f(a)$, $a \in A$.

$w \in f(L)$ ssi il existe $a_1, \ldots, a_n$ t.q.
(i) $w = f(a_1) \cdots f(a_n)$; (ii) $a_1 \cdots a_n \in L$.

Soit $\mathcal{B}_1$ un automate reconnaissant $L \in Rec(B^*)$.

1. On sature l'automate avec des transitions

$$q \xrightarrow{a} q' \qquad \text{if} \qquad q \xrightarrow{f(a)^*} q'$$

2. On supprime les transitions étiquetées par éléments de B.

L'automate résultant accepte $f^{-1}(w)$ ssi $w \in L$.

Propriétés de fermeture

Définition : Substitutions

Une *substitution* est définie par une application $\sigma : A \to \mathcal{P}(B^*)$.
Elle s'étend en un morphisme $\sigma : A^* \to \mathcal{P}(B^*)$ défini par
$\sigma(\varepsilon) = \{\varepsilon\}$ et
$\sigma(a_1 \cdots a_n) = \sigma(a_1) \cdots \sigma(a_n)$.

Pour $L \subseteq A^*$, on note $\sigma(L) = \bigcup_{u \in L} \sigma(u)$.
Pour $L \subseteq B^*$, on note $\sigma^{-1}(L) = \{u \in A^* \mid \sigma(u) \cap L \neq \emptyset\}$.

Une substitution est *rationnelle* (ou *reconnaissable*) si elle est définie par une application $\sigma : A \to \mathrm{Rec}(B^*)$.

Propriétés de fermeture

Proposition :

La famille des langages reconnaissables est fermée par substitution rationnelle et substitution rationnelle inverse.

1. Si $L \in \mathrm{Rec}(A^*)$ et $\sigma : A \to \mathrm{Rec}(B^*)$ est une substitution rationnelle alors $\sigma(L) \in \mathrm{Rec}(B^*)$.
2. Si $L \in \mathrm{Rec}(B^*)$ et $\sigma : A \to \mathrm{Rec}(B^*)$ est une substitution rationnelle alors $\sigma^{-1}(L) \in \mathrm{Rec}(A^*)$.

Preuve

1. On remplace des transitions par des automates.
2. Plus difficile.

Substitution inverse

Soit $\sigma : A \to Rec(B^*)$ une substituion rationnelle, on suppose s.p.d.g. que $A \cap B = \emptyset$.

$w \in \sigma(L)$ ssi il existe $a_1, \ldots, a_n$ et $v_1, \ldots, v_n \in B^*$ t.q.
(i) $w = v_1 \cdots v_n$; (ii) $\forall i : v_i \in \sigma(a_i)$; (iii) $a_1 \cdots a_n \in L$.

Soit $\mathcal{B}_1$ un automate reconnaissant $L \in Rec(B^*)$.

1. Soit $L_{q,q'}$ le langage accepté par $\mathcal{B}_1$ si q état initial et q' seul état final.
2. Pour toute paire q, q' et $a \in A$, ajouter $q \xrightarrow{a} q'$ ssi $L_{q,q'} \cap \sigma(a) \neq \emptyset$.
3. On supprime les transitions étiquetées par éléments de B.

L'automate résultant accepte $\sigma^{-1}(w)$ ssi $w \in L$.

Langages rationnels

Syntaxe pour représenter des langages.

Soit Σ un alphabet et $\underline{\Sigma}$ une copie de Σ.
Une expression rationnelle (ER) est un mot sur l'alphabet $\underline{\Sigma} \cup \{(,), +, \cdot, *, \underline{\emptyset}\}$

Définition : Syntaxe

L'ensemble des ER est défini par

B : $\underline{\emptyset}$ et $\underline{a}$ pour $a \in \Sigma$ sont des ER,

I : Si E et F sont des ER alors $(E + F)$, $(E \cdot F)$ et (E^*) aussi.

On note $\mathcal{E}$ l'ensemble des expressions rationnelles.

Langages rationnels

Définition : Sémantique

On définit $\mathcal{L} : \mathcal{E} \to \mathcal{P}(\Sigma^*)$ par

B : $\mathcal{L}(\underline{\emptyset}) = \emptyset$ et $\mathcal{L}(\underline{a}) = \{a\}$ pour $a \in \Sigma$,

I : $\mathcal{L}((E + F)) = \mathcal{L}(E) \cup \mathcal{L}(F)$, $\mathcal{L}((E \cdot F)) = \mathcal{L}(E) \cdot \mathcal{L}(F)$ et $\mathcal{L}((E^*)) = \mathcal{L}(E)^*$.

Un langage $L \subseteq \Sigma^*$ est rationnel s'il existe une ER E telle que $L = \mathcal{L}(E)$.
On note $\mathrm{Rat}(\Sigma^*)$ l'ensemble des langages rationnels sur l'alphabet Σ.

Remarque : $\mathrm{Rat}(\Sigma^*)$ est la plus petite famille de langages de Σ^* contenant $\emptyset$ et $\{a\}$ pour $a \in \Sigma$ et fermée par union, concaténation, itération.

Langages rationnels

Définition :

Deux ER E et F sont équivalentes (noté $E \equiv F$) si $\mathcal{L}(E) = \mathcal{L}(F)$.

Exemples : commutativité, associativité, distributivité, ...

Peut-on trouver un système de règles de réécriture caractérisant l'équivalence des ER ?

Oui, mais il n'existe pas de système fini.

Comment décider de l'équivalence de deux ER ?

On va utiliser le théorème de Kleene.

Abus de notation :

- On ne souligne pas les lettres de Σ : $((a+b)^*)$.
- On enlève les parenthèses inutiles : $(aa+bb)^* + (aab)^*$.
- On confond langage rationnel et expression rationnelle.

Langages rationnels

Théorème : Kleene, 1936

$$\mathrm{Rec}(\Sigma^*) = \mathrm{Rat}(\Sigma^*)$$

Preuve

$\boxed{\supseteq}$: les langages $\emptyset$ et $\{a\}$ pour $a \in \Sigma$ sont reconnaissables et la famille $\mathrm{Rec}(\Sigma^*)$ est fermée par union, concaténation, itération.

$\boxed{\subseteq}$: Algorithme de McNaughton-Yamada.

Corollaire :

L'équivalence des expressions rationnelles est décidable.

Preuve

Il suffit de l'inclusion $\mathrm{Rat}(\Sigma^*) \subseteq \mathrm{Rec}(\Sigma^*)$.

Algorithme de McNaughton-Yamada

Soit $\mathcal{A} = \langle Q, T, I, F \rangle$ un automate ND. On construit $e \in \mathcal{E}$ t.q. $\mathcal{L}(e) = \mathcal{L}(\mathcal{A})$.
S.p.d.q. les états de $\mathcal{A}$ sont $\{1, \ldots, n\}$.

On note $L_{p,q}$ le langage accepté par $\mathcal{A}$ avec p initial et q seul état final.

On note $L_{p,q}^{(k)}$ le langage de mots non-vides acceptés par $\mathcal{A}$ avec p initial et q final et en n'utilisant que $1, \ldots, k$ entre p et q.

On exprimera $L_{p,q}^{(k)}$ avec des expressions rationnelles $e_{p,q}^{(k)}$.

- $e_{p,q}^{(0)} = \sum\{\, a \mid p \xrightarrow{a} q \,\}$
- pour $1 \leq k \leq n$, $e_{p,q}^{(k)} = e_{p,q}^{(k-1)} + e_{p,k}^{(k-1)} \left(e_{k,k}^{(k-1)}\right)^* e_{k,q}^{(k-1)}$

Finalement, $L_{p,q} = \begin{cases} \mathcal{L}(e_{p,q}^{(n)}) & \text{si } p \neq q \\ \mathcal{L}(e_{p,q}^{(n)} + \emptyset^*) & \text{si } p = q \end{cases}$ et $\mathcal{L}(\mathcal{A}) = \bigcup_{i \in I} \bigcup_{f \in F} L_{i,f}$.

Critères de reconnaissabilité

Y a-t-il des langages non reconnaissables ?

Oui, par un argument de cardinalité.

Comment montrer qu'un langage n'est pas reconnaissable ?

Exemples.

1. $L_1 = \{a^n b^n \mid n \geq 0\}$,
2. $L_2 = \{u \in \Sigma^* \mid |u|_a = |u|_b\}$,
3. $L_3 = L_2 \setminus (\Sigma^*(a^3 + b^3)\Sigma^*)$

Preuves : *à la main* (par l'absurde).

Critères de reconnaissabilité

Lemme : itération

Soit $L \in \mathrm{Rec}(\Sigma^*)$. Il existe $N \geq 0$ tel que pour tout $x \in L$,

1. si $|x| \geq N$ alors $\exists u_1, u_2, u_3 \in \Sigma^*$ tels que $x = u_1u_2u_3$, $u_2 \neq \varepsilon$ et $u_1u_2^*u_3 \subseteq L$.
2. si $x = w_1w_2w_3$ avec $|w_2| \geq N$ alors $\exists u_1, u_2, u_3 \in \Sigma^*$ tels que $w_2 = u_1u_2u_3$, $u_2 \neq \varepsilon$ et $w_1u_1u_2^*u_3w_3 \subseteq L$.
3. si $x = uv_1v_2 \dots v_Nw$ avec $|v_i| \geq 1$ alors il existe $0 \leq j < k \leq N$ tels que $uv_1 \dots v_j(v_{j+1} \dots v_k)^*v_{k+1} \dots v_Nw \subseteq L$.

Preuve

Sur l'automate qui reconnaît L.

Application à L_1, L_2, L_3 et aux palindromes $L_4 = \{u \in \Sigma^* \mid u = \tilde{u}\}$.

Critères de reconnaissabilité

Exercice : Puissance des lemmes d'itérations

1. Montrer que les langages suivants satisfont (1) mais pas (2) :

$$K_1 = \{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$$

$$K_1' = \{b^p a^n \mid p > 0 \text{ et } n \text{ est premier}\} \cup \{a\}^*$$

2. Montrer que le langage suivant satisfait (2) mais pas (3) :

$$K_2 = \{(ab)^n (cd)^n \mid n \geq 0\} \cup \Sigma^* \{aa, bb, cc, dd, ac\} \Sigma^*$$

3. Montrer que le langage suivant satisfait (3) mais n'est pas reconnaissable :

$$K_3 = \{udv \mid u, v \in \{a,b,c\}^* \text{ et soit } u \neq v \text{ soit } u \text{ ou } v \text{ contient un carré}\}$$

Critères de reconnaissabilité

Théorème : Ehrenfeucht, Parikh, Rozenberg ([13, p. 128])

Soit $L \subseteq \Sigma^*$. Les conditions suivantes sont équivalentes :

1. L est reconnaissable
2. Il existe $N > 0$ tel que pour tout mot $x = uv_1 \dots v_N w \in \Sigma^*$ avec $|v_i| \geq 1$, il existe $0 \leq j < k \leq N$ tels que pour tout $n \geq 0$,

$$x \in L \quad \text{ssi} \quad uv_1 \dots v_j (v_{j+1} \dots v_k)^n v_{k+1} \dots v_N w \in L$$

3. Il existe $N > 0$ tel que pour tout mot $x = uv_1 \dots v_N w \in \Sigma^*$ avec $|v_i| \geq 1$, il existe $0 \leq j < k \leq N$ tels que

$$x \in L \quad \text{ssi} \quad uv_1 \dots v_j v_{k+1} \dots v_N w \in L$$

Remarque : la preuve utilise le théorème de Ramsey.

Critères de reconnaissabilité

Pour montrer qu'un langage n'est pas reconnaissable, on peut aussi utiliser les propriétés de clôture.

Exemples : Sachant que L_1 n'est pas reconnaissable.

- $L_2 \cap a^*b^* = L_1$.
  Donc L_2 n'est pas reconnaissable.
- Soit $f : \Sigma^* \to \Sigma^*$ défini par $f(a) = aab$ et $f(b) = abb$.
  On a $f^{-1}(L_3) = L_2$.
  Donc L_3 n'est pas reconnaissable.
- $L_5 = \{u \in \Sigma^* \mid |u|_a \neq |u|_b\} = \overline{L_2}$.
  Donc L_5 n'est pas reconnaissable.

Minimisation

Il y a une infinité d'automates pour un langage donné.

Exemple : automates D ou ND pour a^*.

Questions :

- Y a-t-il un automate canonique ?
- Y a-t-il unicité d'un automate minimal en nombre d'états ?
- Y a-t-il un lien structurel entre deux automates qui reconnaissent le même langage ?

Automate des résiduels

Définition : Résiduels

Soient $u \in \Sigma^*$ et $L \subseteq \Sigma^*$.
Le résiduel de L par u est le quotient $u^{-1}L = \{v \in \Sigma^* \mid uv \in L\}$.

Exemple : Calculer les résiduels des langages

$L_j = \{u = u_0 u_1 \cdots u_n \in \{0,1\}^* \mid \overline{u}^2 = \sum_{i=0}^{n} u_i 2^i \equiv j \ [3]\}$.

Définition : Automate des résiduels

Soit $L \subseteq \Sigma^*$. L'automate des résiduels de L est $\mathcal{R}(L) = (Q_L, \delta_L, i_L, F_L)$ avec

- $Q_L = \{u^{-1}L \mid u \in \Sigma^*\}$,
- $\delta_L(u^{-1}L, a) = a^{-1}(u^{-1}L) = (ua)^{-1}L$,
- $i_L = L = \varepsilon^{-1}L$,
- $F_L = \{u^{-1}L \mid \varepsilon \in u^{-1}L\} = \{u^{-1}L \mid u \in L\}$.

Théorème :

Un langage $L \subseteq \Sigma^*$ est reconnaissable ssi L a un nombre fini de résiduels.

Exemples: Résiduels

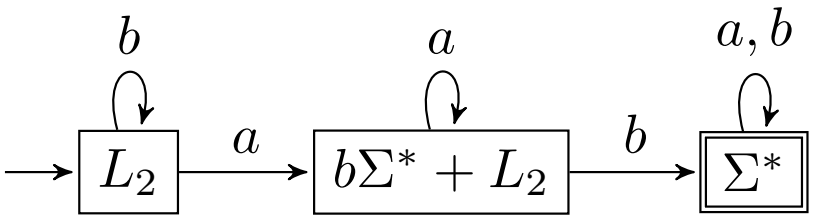
Résiduels de $L_1 := \{ a^n b^n \mid n \geq 0 \}$:

- pour $i \geq 0$, $(a^i)^{-1} L_1 = \{ a^j b^{i+j} \mid j \geq 0 \}$;
- pour $1 \leq j \leq i$, $(a^i b^j)^{-1} L_1 = \{ b^{i-j} \}$;
- pour tout autre mot u, $u^{-1} L = \emptyset$.

Résiduels de $L_2 := \Sigma^* ab \Sigma^*$:

- L_2
- $b\Sigma^* + L_2$
- Σ^*

Automate de résiduels ($\Sigma = \{a, b\}$):



Preuve du théorème

Théorème :

Un langage $L \subseteq \Sigma^*$ est reconnaissable ssi L a un nombre fini de résiduels.

- Si L a un nombre fini de résiduels, alors l'aut. de rés. est fini et DC.
  On montre (par récurrence sur la longueur) que pour tout $u \in \Sigma^*$, l'unique calcul de cet automate mène dans l'état $u^{-1}L$.
  L'automate accepte alors L, grâce à la définition de F.
- Soit L avec un nombre infini de résiduels. Supposons qu'il existe un automate fini DC $\mathcal{A}$ qui accepte L.
  Alors il existe $u, v \in \Sigma^*$ avec $u^{-1}L \neq v^{-1}L$ et que les calculs uniques de $\mathcal{A}$ pour u et v mènent au même état q.
  S.p.d.g., on trouve w tq $w \in u^{-1}L \setminus v^{-1}L$.
  Du coup, $\mathcal{A}$ accepte w depuis q. Mais alors $\mathcal{A}$ accepte vw depuis son état initial, une contradiction.

Congruences et quotients

Définition : Congruence sur les automates

Soit $\mathcal{A}$ un automate DC. Une relation d'équivalence $\sim$ sur Q est une congruence si

- $\forall p, q \in Q$, $\forall a \in \Sigma$, $p \sim q$ implique $\delta(p, a) \sim \delta(q, a)$,
- F est saturé par $\sim$, i.e., $\forall p \in F$, $[p] = \{q \in Q \mid p \sim q\} \subseteq F$.

Le quotient de $\mathcal{A}$ par $\sim$ est $\mathcal{A}/\sim = (Q/\sim, \delta_\sim, [i], F/\sim)$
où $\delta_\sim$ est définie par $\delta_\sim([p], a) = [\delta(p, a)]$.

Exemple :

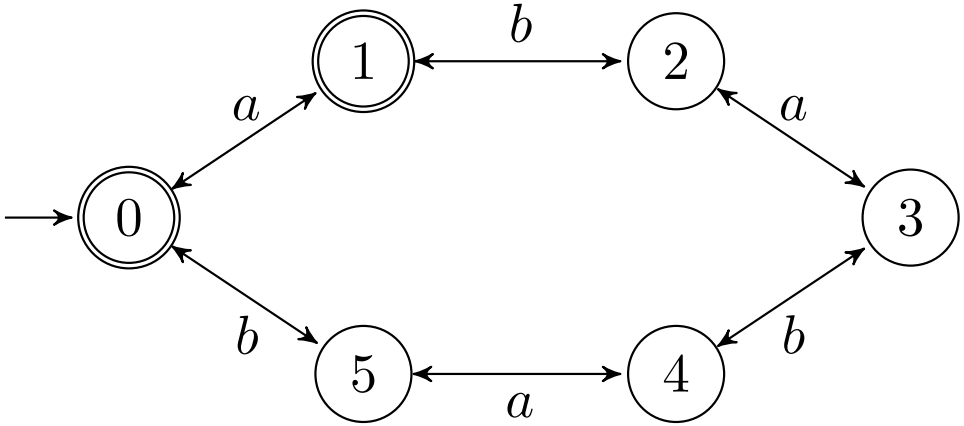
Donner un automate DCA $\mathcal{A}$ à 6 états qui 'calcule' $\overline{u}^2 \bmod 3$ et accepte L_1.
Exhiber une congruence non triviale sur $\mathcal{A}$.
Calculer le quotient $\mathcal{A}/\sim$.

Proposition :

$\mathcal{A}/\sim$ est bien défini et $\mathcal{L}(\mathcal{A}/\sim) = \mathcal{L}(\mathcal{A})$.

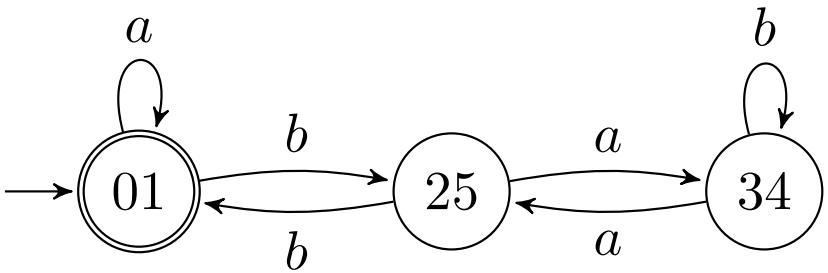
Exemple : Congruence

Automate $\mathcal{A}$:



Congruence possible: $0 \sim 1$, $2 \sim 5$, $3 \sim 4$

Automate $\mathcal{A}/\sim$:



Équivalence de Nerode

Définition : Équivalence de Nerode

Soit $\mathcal{A} = (Q, \delta, i, F)$ un automate DCA (DC et accessible) reconnaissant L.

Pour $q \in Q$, on note $\mathcal{L}(\mathcal{A}, q) = \{u \in \Sigma^* \mid \delta(q, u) \in F\}$.

L'équivalence de Nerode de $\mathcal{A}$ est définie par $p \sim q$ si $\mathcal{L}(\mathcal{A}, p) = \mathcal{L}(\mathcal{A}, q)$.

Proposition :

L'équivalence de Nerode est une congruence.

L'automate $\mathcal{A}/\sim$ est appelé quotient de Nerode de $\mathcal{A}$.

Théorème : $\mathcal{A}/\sim = \mathcal{R}(L)$

Le quotient de Nerode est isomorphe à l'automate des résiduels.

$\varphi : Q/\sim \to Q_L$ définie par $\varphi([q]) = \mathcal{L}(\mathcal{A}, q)$ est un isomorphisme de $\mathcal{A}/\sim$ sur $\mathcal{R}(L)$.

Preuve: L'équivalence de Nerode est une congruence

(i) $p \sim q$ implique $\delta(p,a) \sim \delta(q,a)$, pour tout $a \in \Sigma$:

$$\begin{aligned} p \sim q &\Leftrightarrow \mathcal{L}(\mathcal{A},p) = \mathcal{L}(\mathcal{A},q) \\ &\Rightarrow \forall w = aw' : w \in \mathcal{L}(\mathcal{A},p) \leftrightarrow w \in \mathcal{L}(\mathcal{A},q) \\ &\Rightarrow \forall w' : w' \in \mathcal{L}(\mathcal{A},\delta(p,a)) \leftrightarrow w' \in \mathcal{L}(\mathcal{A},\delta(q,a)) \\ &\Leftrightarrow \mathcal{L}(\mathcal{A},\delta(p,a)) = \mathcal{L}(\mathcal{A},\delta(q,a)) \end{aligned}$$

(ii) F est saturé par $\sim$:

$$\begin{aligned} p \sim q &\Rightarrow \varepsilon \in \mathcal{L}(\mathcal{A},p) \leftrightarrow \varepsilon \in \mathcal{L}(\mathcal{A},q) \\ &\Leftrightarrow p \in F \leftrightarrow q \in F \end{aligned}$$

Automate minimal

Théorème :

Soit $L \in \mathrm{Rec}(\Sigma^*)$.

1. Si $\mathcal{A}$ est un automate DCA qui reconnaît L, alors $\mathcal{R}(L)$ est un quotient de $\mathcal{A}$.
2. $\mathcal{R}(L)$ est minimal parmi les automates DCA reconnaissant L.
   (minimal en nombre d'états et minimal pour l'ordre quotient)
3. Soit $\mathcal{A}$ un automate DC reconnaissant L avec un nombre minimal d'états.
   $\mathcal{A}$ est isomorphe à $\mathcal{R}(L)$ (unicité de l'automate minimal)

Corollaire :

1. On obtient l'automate minimal de L en calculant le quotient de Nerode de n'importe quel automate DCA qui reconnaît L.
2. Soient $\mathcal{A}$ et $\mathcal{B}$ deux automates DCA. Pour tester si $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$:
   Calculer les quotients de Nerode puis tester leur égalité : $\mathcal{A}/\sim = \mathcal{B}/\sim$.

Problème : comment calculer le quotient de Nerode *efficacement* ?

Preuve: Automate minimal

1. C'est le quotient de Nerode.

2. et 3.: Soit $\mathcal{A}$ un automate déterministe acceptant L, et pour un état q de $\mathcal{A}$ soit

$$L_q := \{\, w \mid \delta_{\mathcal{A}}(q_0, w) = q \,\}$$

On prouve d'abord que $u, v \in L_q$ implique $u^{-1}L = v^{-1}L$.

2. Si $\mathcal{A}$ est DC et possède moins d'états que $\mathcal{R}(L)$, le principe de tiroirs montre une contradiction avec le lemme.

3. Si $\mathcal{A}$ est DC avec le même nombre d'états que $\mathcal{R}(L)$, le lemme implique $\{\, L_q \mid q \in Q \,\} = Q_L$. L'isomorphisme sur les états est $\phi(q) = L_q$.

Algorithme de Moore

Pour $n \geq 0$, on note $\Sigma^{\leq n} = \Sigma^0 \cup \Sigma^1 \cup \cdots \cup \Sigma^n$ et on définit l'équivalence $\sim_n$ sur Q par

$$\begin{array}{lll} p \sim_n q & \text{ssi} & \mathcal{L}(\mathcal{A}, p) \cap \Sigma^{\leq n} = \mathcal{L}(\mathcal{A}, q) \cap \Sigma^{\leq n} \\ & \text{ssi} & \forall w \in \Sigma^{\leq n}, \quad \delta(p, w) \in F \iff \delta(q, w) \in F \end{array}$$

Remarque 1 : $\sim_0$ a pour classes d'équivalence F et $Q \setminus F$.

Remarque 2 : $\sim_{n+1}$ est plus fine que $\sim_n$, i.e., $p \sim_{n+1} q \Longrightarrow p \sim_n q$.

Remarque 3 : $\sim = \bigcap_{n \geq 0} \sim_n$, i.e., $p \sim q$ ssi $\forall n \geq 0$, $p \sim_n q$.

Proposition : Soit $\mathcal{A}$ automate DC

- $p \sim_{n+1} q$ ssi $p \sim_n q$ et $\forall a \in \Sigma$, $\delta(p, a) \sim_n \delta(q, a)$.
- Si $\sim_n = \sim_{n+1}$ alors $\sim = \sim_n$.
- $\sim = \sim_{|Q|-2}$ si $\emptyset \neq F \neq Q$ et $\sim = \sim_0$ sinon.

Permet de calculer l'équivalence de Nerode par raffinements successifs.

Exercice :

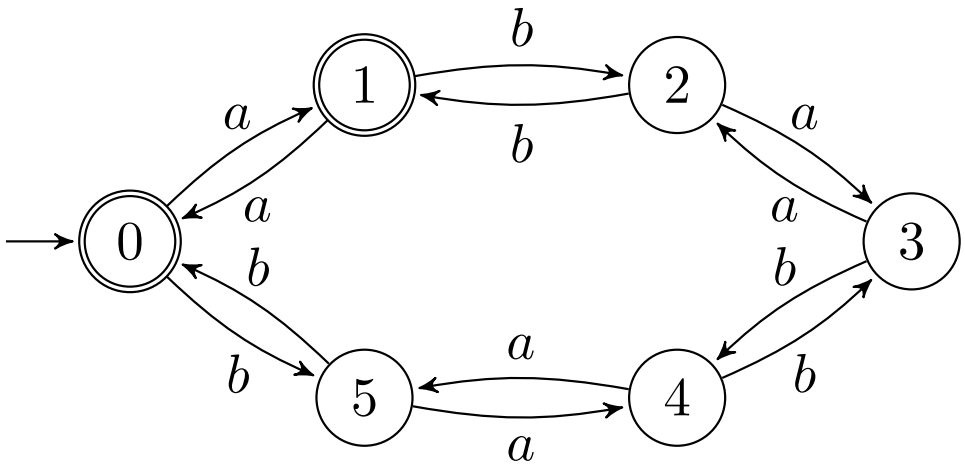
Calculer l'automate minimal par l'algorithme d'Hopcroft de raffinement de partitions en $\mathcal{O}(n \log(n))$ (l'algo naïf est en $\mathcal{O}(n^2)$ avec $n = |Q|$).

Explications: Algorithme de Moore

- Disons qu'un mot w distingue une paire d'états p, q si w est accepté depuis p mais pas depuis q (ou l'inverse).
- L'algorithme de Moore cherche à trouver, pour toute paire d'états p, q, un mot de longueur minimal qui distingue p et q.
- Si $p \in F$ et $q \in Q \setminus F$, c'est ε ($\sim_0$).
- Sinon, pour toute paire $p \sim_n q$, on teste $\delta(p, a) \sim_n \delta(q, a)$ pour tout $a \in \Sigma$.
  - Si un tel a existe et w distingue $\delta(p, a)$ et $\delta(q, a)$, alors aw distingue p et q.
  - Si aucun tel a existe, alors $p \sim_{n+1} q$.
- Si $\sim_n = \sim_{n+1}$, on peut terminer.

Exemple: Algorithme de Moore

Automate $\mathcal{A}$:

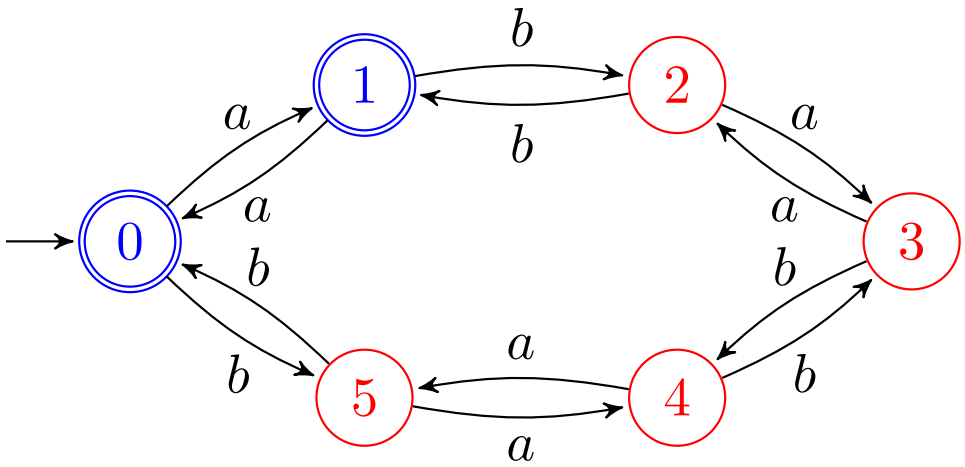


Notons les mots distinctifs dans un tableau triangulaire :

1	2	3	4	5	
					0
—					1
—	—				2
—	—	—			3
—	—	—	—		4

Exemple: Algorithme de Moore

Automate $\mathcal{A}$:

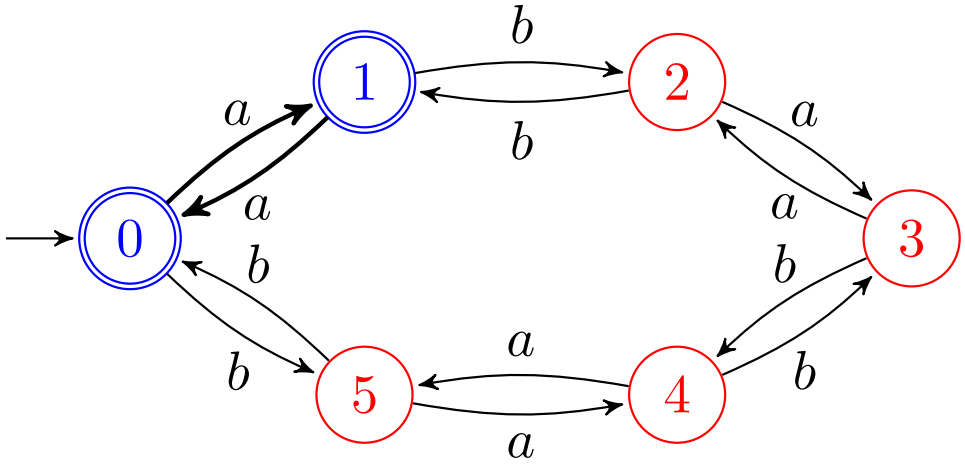


On démarre avec les paires état final / état non-final ($\sim_0$) :

1	2	3	4	5	
	ε	ε	ε	ε	0
—	ε	ε	ε	ε	1
—	—				2
—	—	—			3
—	—	—	—		4

Exemple: Algorithme de Moore

Automate $\mathcal{A}$:

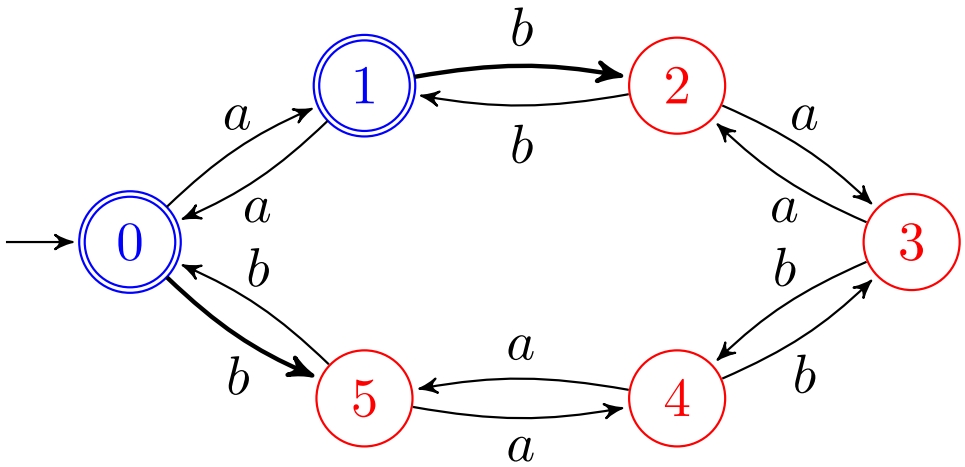


Tester si a distingue $0, 1$: non, $1 = \delta(0, a) \sim_0 \delta(1, a) = 0$

1	2	3	4	5	
	ε	ε	ε	ε	0
—	ε	ε	ε	ε	1
—	—				2
—	—	—			3
—	—	—	—		4

Exemple: Algorithme de Moore

Automate $\mathcal{A}$:

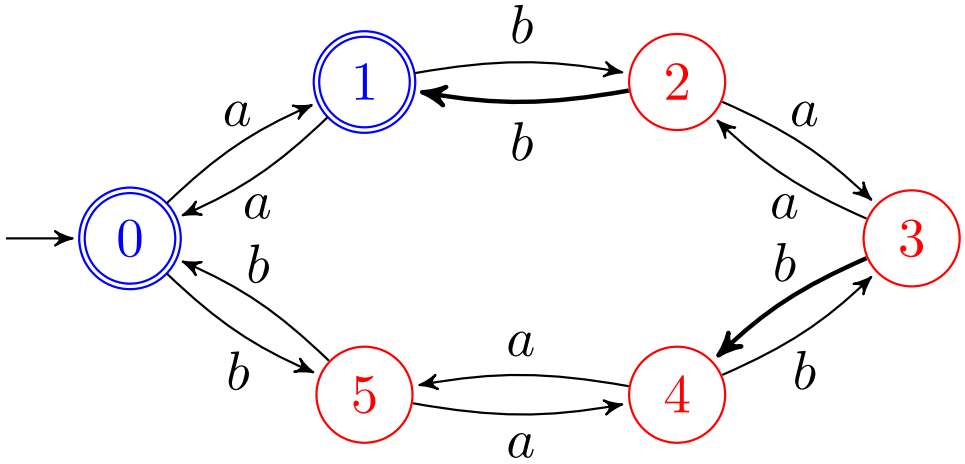


Tester si b distingue $0, 1$: non, $5 = \delta(0, b) \sim_0 \delta(1, b) = 2$

1	2	3	4	5	
	ε	ε	ε	ε	0
—	ε	ε	ε	ε	1
—	—				2
—	—	—			3
—	—	—	—		4

Exemple: Algorithme de Moore

Automate $\mathcal{A}$:

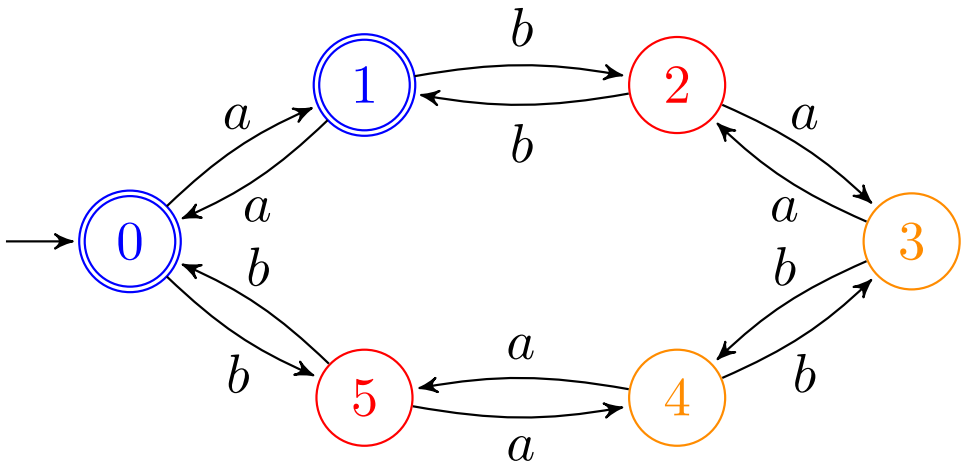


Tester si b distingue $2, 3$: oui, $1 = \delta(2, b) \not\sim_0 \delta(3, b) = 4$

1	2	3	4	5	
	ε	ε	ε	ε	0
—	ε	ε	ε	ε	1
—	—	b			2
—	—	—			3
—	—	—	—		4

Exemple: Algorithme de Moore

Automate $\mathcal{A}$:

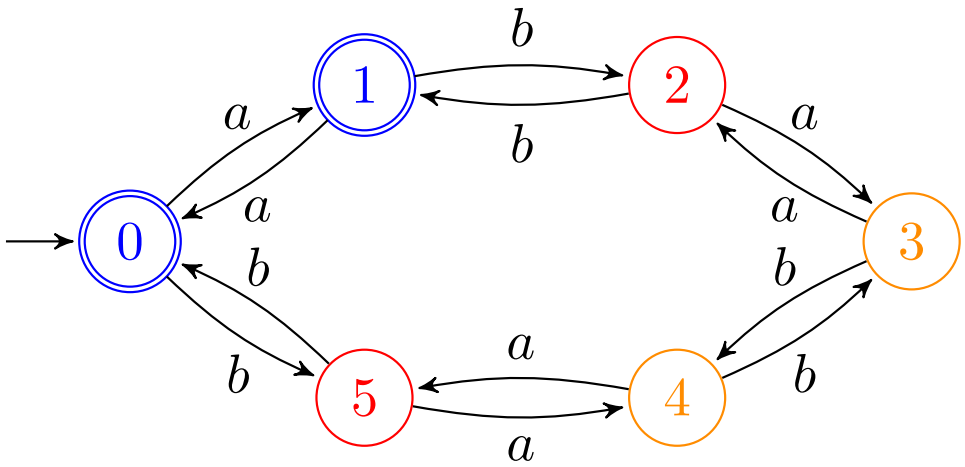


Et de suite, le résultat final pour $\sim_1$:

1	2	3	4	5	
	ε	ε	ε	ε	0
—	ε	ε	ε	ε	1
—	—	b	b		2
—	—	—		b	3
—	—	—	—	b	4

Exemple: Algorithme de Moore

Automate $\mathcal{A}$:



On constate $\sim_2 = \sim_1$, du coup on fusionne 0 avec 1, 2 avec 5, et 3 avec 4.

1	2	3	4	5	
	ε	ε	ε	ε	0
—	ε	ε	ε	ε	1
—	—	b	b		2
—	—	—		b	3
—	—	—	—	b	4

Logique sur les mots

Définition : Syntaxe de $\mathrm{MSO}(\Sigma, <)$

$$\varphi ::= \bot \mid P_a(x) \mid x < y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\, \varphi \mid \exists X\, \varphi$$

avec $a \in \Sigma$, $\{x, y, \ldots\}$ variables du premier ordre, $\{\mathrm{X}, \mathrm{Y}, \ldots\}$ variables monadiques du second ordre.

Définition : Sémantique de $\mathrm{MSO}(\Sigma, <)$

Soit $w = w_1 w_2 \cdots w_n \in \Sigma^+$ un mot et $\mathrm{pos}(w) = \{1, 2, \ldots, n\}$ les positions du mot.
Soit σ une valuation :
$\sigma(x) \in \mathrm{pos}(w)$ si x est une variable du premier ordre et
$\sigma(X) \subseteq \mathrm{pos}(w)$ si X est une variable monadique du second ordre.

$$\begin{array}{lll}
w, \sigma \models P_a(x) & \text{si} & w_{\sigma(x)} = a \\
w, \sigma \models x < y & \text{si} & \sigma(x) < \sigma(y) \\
w, \sigma \models x \in X & \text{si} & \sigma(x) \in \sigma(X) \\
w, \sigma \models \exists x\, \varphi & \text{si} & \exists\, i \in \mathrm{pos}(w) \text{ tel que } w, \sigma[x \mapsto i] \models \varphi \\
w, \sigma \models \exists X\, \varphi & \text{si} & \exists\, I \subseteq \mathrm{pos}(w) \text{ tel que } w, \sigma[X \mapsto I] \models \varphi
\end{array}$$

MSO: Raccourcis

- On peut exprimer $\forall x$, $\forall X$, $\vee$, etc comme d'habitude.
- $X \subseteq Y$:

$$\forall x.(x \in X \rightarrow x \in Y)$$

- $Z = X \cup Y$:

$$\forall x.(x \in Z \leftrightarrow x \in X \vee x \in Y)$$

- $Partition(X_1, \ldots, X_m)$:

$$\left(\forall x. \bigvee_{i=1}^{m} x \in X_i\right) \wedge \left(\bigwedge_{i=1}^{m} \bigwedge_{j \neq i} \forall x.(x \notin X_i \vee x \notin X_j)\right)$$

- $First(x)$:

$$\forall y : x \leq y$$

- $Succ(x, y)$:

$$x < y \wedge \forall z : \neg(x < z \wedge z < y)$$

- En plus, $X = \emptyset$, $X = \{x\}$, $X = Y$,...

Logique sur les mots

Définition : Codage d'une valuation dans l'alphabet

Soit V un ensemble de variables, on note $\Sigma_V = \Sigma \times \{0,1\}^V$.
Un couple (w, σ) où $w = w_1 w_2 \cdots w_n \in \Sigma^+$ est un mot sur l'alphabet Σ et σ est une valuation des variables de V est codé par un mot $W = (w_1, \tau_1) \cdots (w_n, \tau_n) \in \Sigma_V^+$ sur l'alphabet Σ_V avec:

- $\forall i \in \mathrm{pos}(w)$, $\tau_i(x) = 1$ ssi $\sigma(x) = i$
  si $x \in V$ est une variable du premier ordre,
- $\forall i \in \mathrm{pos}(w)$, $\tau_i(X) = 1$ ssi $i \in \sigma(X)$
  si $X \in V$ est une variable monadique du second ordre.

Un mot $W \in \Sigma_V^+$ est valide si il code un couple (w, σ). On identifie W et (w, σ).

Remarque: $\{W \in \Sigma_V^+ \mid W \text{ est valide}\}$ est reconnaissable.

Définition : Sémantique de $\mathrm{MSO}(\Sigma, <)$

Soit $\varphi \in \mathrm{MSO}(\Sigma, <)$ et soit V un ensemble de variables contenant les variables libres de φ,

$$\mathcal{L}_V(\varphi) = \{W \in \Sigma_V^+ \mid W = (w, \sigma) \text{ est valide et } (w, \sigma) \models \varphi\}$$

Exemples

- Soit $w = abc$ et $\sigma(x) = 2$, $\sigma(y) = 3$, and $\sigma(Z) = \{1, 3\}$.
  Alors $\langle w, \sigma \rangle$ est codé par $\langle a, 001 \rangle \langle b, 100 \rangle \langle c, 011 \rangle$.

- Si on n'a que $\sigma'(x) = 2$, alors $\langle w, \sigma' \rangle$ est codé par $\langle a, 0 \rangle \langle b, 1 \rangle \langle c, 0 \rangle$.

- $w, \sigma' \models P_b(x)$, du coup $\langle w, \sigma' \rangle \in \mathcal{L}(P_b(x))$

- $w \in \mathcal{L}(\exists x.P_b(x))$

Logique sur les mots

Théorème : Büchi 1960, Elgot 1961, Trakhtenbrot 1961

Un langage $L \subseteq \Sigma^+$ est reconnaissable si et seulement si il est définissable par une formule close $\varphi \in \mathrm{MSO}(\Sigma, <)$.

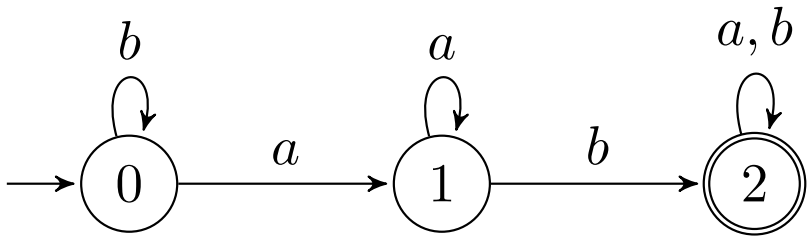
Preuve

$\Longrightarrow$: Si L est reconnu par un automate $\mathcal{A}$ ayant n états $Q = \{q_1, \ldots, q_n\}$, on écrit une formule de la forme $\varphi = \exists X_1 \cdots \exists X_n \, \psi(X_1, \ldots, X_n)$ qui caractérise l'existence d'un calcul acceptant de $\mathcal{A}$ sur un mot $w \in \Sigma^+$.
X_i est l'ensemble des positions de w pour lesquelles le calcul est dans l'état q_i.
La formule ψ assure que les transitions de l'automate sont respectées.

$\Longleftarrow$: On donne des expressions rationnelles pour les formules atomiques.
On note $\Sigma_V^{x=1} = \{(a, \tau) \in \Sigma_V \mid \tau(x) = 1\}$, de même $\Sigma_V^{x=0}$ ou $\Sigma_V^{X=1}$ou $\Sigma_V^{X=0}$.
$\mathcal{L}_V(P_a(x)) = (\Sigma_V^{x=0})^*(\Sigma_V^{x=1} \cap (\{a\} \times \{0,1\}^V))(\Sigma_V^{x=0})^*$.
$\mathcal{L}_V(x \in X) = (\Sigma_V^{x=0})^*(\Sigma_V^{x=1} \cap \Sigma_V^{X=1})(\Sigma_V^{x=0})^*$.
$\mathcal{L}_V(x < y) = (\Sigma_V^{x=y=0})^*(\Sigma_V^{x=1} \cap \Sigma_V^{y=0})(\Sigma_V^{x=y=0})^*(\Sigma_V^{x=0} \cap \Sigma_V^{y=1})(\Sigma_V^{x=y=0})^*$.
On utilise les propriétés de clôture des langages reconnaissables :
union ($\vee$), complémentaire ($\neg$) et projection ($\exists x$ et $\exists X$).

Exemple: Automate $\rightarrow$ MSO

- Automate :



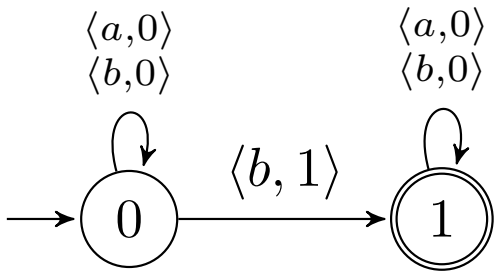
- Formule MSO :

$$\begin{aligned}\phi \quad = \quad & \exists Q_0, Q_1, Q_2.Partition(Q_0, Q_1, Q_2) \\ & \wedge \forall x.(First(x) \rightarrow x \in Q_0) \\ & \wedge \forall x, y.(Succ(x, y) \wedge x \in Q_0 \wedge P_a(x) \rightarrow y \in Q_1) \\ & \wedge \forall x, y.(Succ(x, y) \wedge x \in Q_0 \wedge P_b(x) \rightarrow y \in Q_0) \\ & \wedge \ \ldots \\ & \wedge \forall x.Last(x) \rightarrow (x \in Q_2 \vee (x \in Q_1 \wedge P_b(x)))\end{aligned}$$

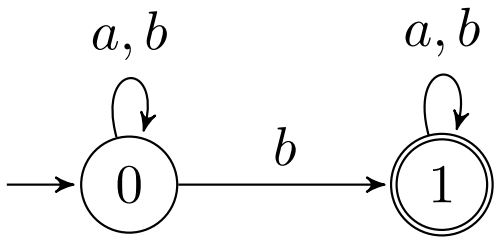
Exemple: MSO $\rightarrow$ Automate

Soit $\Sigma = \{a, b\}$.

- $\phi = P_b(x)$ donne l'automate $\mathcal{A}_\phi$ suivant :



- $\phi' = \exists x.P_b(x)$; on l'obtient $\mathcal{A}_{\phi'}$ en supprimant x dans $\mathcal{A}_\phi$ (autrement dit, on projète toute lettre $\langle a, x\rangle$ vers a).



Morphismes

Définition : Reconnaissance par morphisme

- $\varphi : \Sigma^* \to M$ morphisme dans un monoïde fini M.
  $L \subseteq \Sigma^*$ est *reconnu* ou *saturé* par φ si $L = \varphi^{-1}(\varphi(L))$.
- $L \subseteq \Sigma^*$ est reconnu par un monoïde fini M s'il existe un morphisme $\varphi : \Sigma^* \to M$ qui reconnaît L.
- $L \subseteq \Sigma^*$ est *reconnaissable par morphisme* s'il existe un monoïde fini qui reconnaît L.

Définition : Monoïde de transitions

Soit $\mathcal{A} = (Q, \Sigma, \delta, i, F)$ un automate déterministe complet.
Le monoïde de transitions de $\mathcal{A}$ est le sous monoïde de $(Q^Q, *)$ engendré par les applications $\delta_a : Q \to Q$ $(a \in \Sigma)$ définies par $\delta_a(q) = \delta(q, a)$ et avec la loi de composition interne $f * g = g \circ f$.

Proposition :

Le monoïde de transitions de $\mathcal{A}$ reconnaît $\mathcal{L}(\mathcal{A})$.

Monoïdes et morphismes

Rappels:

- Monoïde M: structure algébrique avec une opération associative $\cdot$ et élément neutre e.
- $\phi : \Sigma^* \to M$ morphisme satisfait $\phi(uv) = \phi(u) \cdot \phi(v)$.
- Du coup, ϕ est entièrement défini par les $\phi(a)$, $a \in \Sigma$.
- Par nécessité, $\phi(\varepsilon) = e$.

Exemple d'un monoïde

Exemple:

- $M = \{A, B, C, D, E\}$ avec élément neutre E et

$u \setminus v$	A	B	C	D	E
A	A	C	C	C	A
B	D	B	C	D	B
C	C	C	C	C	C
D	D	C	C	C	D
E	A	B	C	D	E

- Avec $\Sigma = \{a, b\}$ et $\phi(a) := A$, $\phi(b) := B$ on obtient

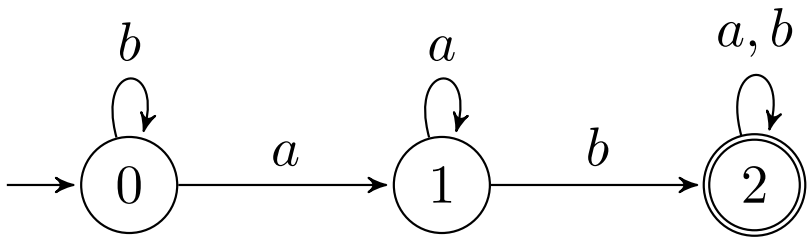
$$\phi^{-1}(A) = a^+ \quad \phi^{-1}(B) = b^+ \quad \phi^{-1}(C) = \Sigma^* ab \Sigma^*$$

$$\phi^{-1}(D) = b^+ a^+ \quad \phi^{-1}(E) = \varepsilon$$

- ϕ reconnaît $\bigcup_{m \in M'} \phi^{-1}(m)$, pour tout $M' \subseteq M$.

Exemple: Monoïde de transitions

- Automate DC :



- On va écrire les fonctions Q^Q sous forme de vecteur :
  - élément neutre: $\langle 0, 1, 2\rangle$
  - $\phi(a) = \langle 1, 1, 2\rangle$ (à savoir $0 \xrightarrow{a} 1$, $1 \xrightarrow{a} 1$, $2 \xrightarrow{a} 2$)
  - $\phi(b) = \langle 0, 2, 2\rangle$
  - p.ex. $\phi(ab) = \langle 1, 1, 2\rangle * \langle 0, 2, 2\rangle = \langle 2, 2, 2\rangle$

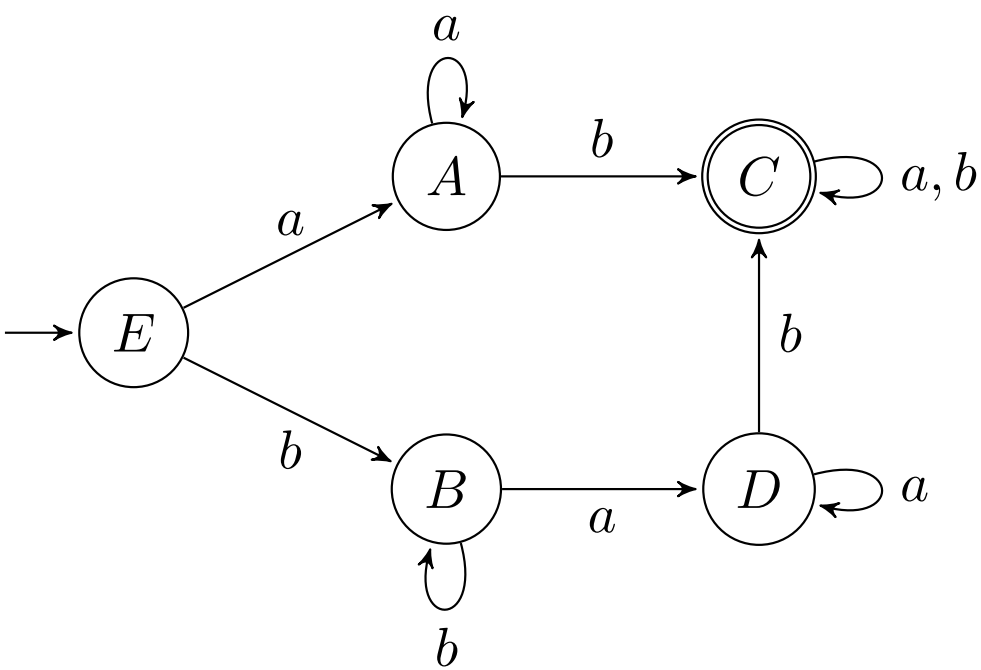
- $\mathcal{L}(\mathcal{A}) = \phi^{-1}(\{\, m \mid m(i) \in F \,\})$

- Énumérer le monoïde de transitions : construire $\phi(\Sigma^{\leq n})$, pour $n \geq 0$ croissant, jusqu'à un n t.q. $\phi(\Sigma^{\leq n}) = \phi(\Sigma^{\leq n+1})$

Morphisme $\rightarrow$ Automate

Soit L reconnu par $\phi : \Sigma^* \rightarrow M$. L'automate $\mathcal{A} = \langle M, \delta, \phi(\varepsilon), \phi(L) \rangle$ avec $\delta(m, a) = m \cdot \phi(a)$ reconnaît L.

Exemple: $M = \{A, B, C, D, E\}$ et ϕ comme auparavant.



Morphismes

Théorème :

Soit $L \subseteq \Sigma^*$. L est reconnaissable par morphisme ssi L est reconnaissable par automate.

Corollaire :

$\mathrm{Rec}(\Sigma^*)$ est fermée par morphisme inverse.

Exemple :

Si L est reconnaissable alors $\sqrt{L} = \{v \in \Sigma^* \mid v^2 \in L\}$ est aussi reconnaissable.

Exercices :

1. Montrer que $\mathrm{Rec}(\Sigma^*)$ est fermée par union, intersection, complémentaire.
2. Montrer que $\mathrm{Rec}(\Sigma^*)$ est fermée par quotients.
   Si $L \in \mathrm{Rec}(\Sigma^*)$ et $K \subseteq \Sigma^*$ alors $K^{-1}L$ et LK^{-1} sont reconnaissables.
3. Montrer que $\mathrm{Rec}(\Sigma^*)$ est fermée par concaténation (plus difficile).

Ferméture par union etc.

Soient $\phi : \Sigma^* \to M$ et $\xi : \Sigma^* \to N$ des morphismes reconnaissant K resp. L.

- Complément: $\Sigma^* \setminus K$ est reconnu par ϕ car L est saturé par ϕ:
  Si $M' := M \setminus \phi(K)$, alors $\phi(u) \in M'$ ssi $u \notin K$.

- Intersection: Soit $O = M \times N$ et $\chi : \Sigma^* \to O$ avec $\chi(a) = \langle \phi(a), \xi(a) \rangle$, $\forall a \in \Sigma$, et $\langle m, n \rangle \cdot \langle m', n' \rangle = \langle m \cdot m', n \cdot n' \rangle$. Alors χ reconnaît $K \cap L$:
  Si $O' := \phi(K) \cap \xi(L)$, alors $\chi(u) \in O'$ ssi $u \in K \cap L$.

- Union: Avec χ comme avant, ξ reconnaît $K \cup L$:
  Si $O'' := (\phi(K) \times N) \cup (M \times \xi(L))$, alors $\chi(u) \in O''$ ssi $u \in K \cup L$.

Ferméture par concaténation

Soient $\phi : \Sigma^* \to M$ et $\xi : \Sigma^* \to N$ des morphismes reconnaissant K resp. L.
S.p.d.g. soit ε le seul mot t.q. ϕ, ξ donnent l'élément neutre.

- Concaténation: Soit $P := 2^{M \times N}$ et $\pi : \Sigma^* \to P$ avec

$$\forall a \in \Sigma : \pi(a) = \{\langle \phi(a), \xi(\varepsilon)\rangle, \langle \phi(\varepsilon), \xi(a)\rangle\}$$

et

$$\begin{aligned} X \cdot Y \quad = \quad & \{ \langle m \cdot m', n' \rangle \mid \langle m, \xi(\varepsilon) \rangle \in X,\ \langle m', n' \rangle \in Y \} \\ & \cup \{ \langle m, n \cdot n' \rangle \mid \langle m, n \rangle \in X,\ \langle \phi(\varepsilon), n' \rangle \in Y \} \end{aligned}$$

Alors $\pi(w) = \{ \langle \phi(u), \xi(v) \rangle \mid w = uv \}$
et $w \in K \cdot L$ ssi $\pi(w) \cap (\phi(K) \times \xi(L)) \neq \emptyset$.

Congruences

Définition : Congruence

Une relation d'équivalence $\equiv$ sur Σ^* s'appelle *congruence*
si $u \equiv v$ implique $\forall x, y : xuy \equiv xvy$.

Définition : Saturation

Soit $L \subseteq \Sigma^*$ et $\equiv$ une congruence sur Σ^*.
L est *saturé* par $\equiv$ si $\forall u, v \in \Sigma^*$, $u \equiv v$ implique $u \in L \iff v \in L$.

Théorème :

Soit $L \subseteq \Sigma^*$. L est reconnaissable ssi L est saturé par une congruence d'index fini.

Preuve: Conséquence de la preuve du prochain théorème

Congruence syntaxique

Définition : Congruence syntaxique

Soit $L \subseteq \Sigma^*$. $u \equiv_L v$ si $\forall x, y \in \Sigma^*, xuy \in L \iff xvy \in L$.

P.ex., pour $L = \Sigma^* ab \Sigma^*$, on a $b \equiv_L b^n$, pour tout $n \geq 1$.

Théorème :

Soit $L \subseteq \Sigma^*$.

- $\equiv_L$ est une congruence et $\equiv_L$ sature L.
- $\equiv_L$ est la plus *grossière* congruence qui sature L.
- L est reconnaissable ssi $\equiv_L$ est d'index fini.

Preuve: Congruence syntaxique

- $\equiv_L$ est une congruence:

$$\begin{aligned} u \equiv_L v &\implies \forall x, y : xuy \in L \Leftrightarrow xvy \in L \\ &\implies \forall x, x', y, y' : x'xuyy' \in L \Leftrightarrow x'xvyy' \in L \\ &\implies \forall x, y : xuy \equiv_L xvy \end{aligned}$$

- $\equiv_L$ sature L:

$$\begin{aligned} u \equiv_L v &\implies \forall x, y : xuy \in L \Leftrightarrow xvy \in L \\ &\implies u \in L \Leftrightarrow v \in L \quad \text{(avec } x = y = \varepsilon\text{)} \end{aligned}$$

- $\equiv_L$ est la plus grossière congruence qui sature L:
  Soit $\equiv$ une congruence qui sature L et $u \equiv v$, on montre alors $u \equiv_L v$.

$$\begin{aligned} u \equiv v &\implies \forall x, y : xuy \equiv xvy \\ &\implies \forall x, y : xuy \in L \Leftrightarrow xvy \in L \quad (\equiv \text{ sature L}) \\ &\implies u \equiv_L v \end{aligned}$$

Preuve: Congruence syntaxique

- Si $\equiv_L$ d'index fini, alors L reconnaissable:

  On considère le monoïde fini $M_L = \Sigma^* / \equiv_L$ avec $[u] \cdot [v] = [uv]$ (bien donné car il s'agit d'une congruence), et le morphisme $\phi : \Sigma^* \to M_L$ défini par $\phi(a) = [a]$, pour tout $a \in \Sigma$. Reste à montrer que ϕ reconnaît L.

  Montrons $\phi^{-1}(\phi(L)) \subseteq L$ (l'autre direction étant triviale).
  Soit $u \in \phi^{-1}(\phi(L))$, il existe alors $m \in M_L$ et $v \in L$ t.q. $\phi(u) = \phi(v) = m$.
  Du coup, $u \equiv_L v$, et $u \in L$.

- Si L reconnaissable, alors $\equiv_L$ d'index fini:

  Soit L reconnu par $\phi : \Sigma^* \to M$, avec M fini. Définissons $u \equiv v$ ssi $\phi(u) = \phi(v)$ et montrons que $\equiv$ est une congruence saturant L.

  - $\equiv$ est une congruence: Soit $\phi(u) = \phi(v)$, alors:

    $$\forall x, y : \phi(xuy) = \phi(x) \cdot \phi(u) \cdot \phi(y) = \phi(x) \cdot \phi(v) \cdot \phi(y) = \phi(xvy)$$

  - $\equiv$ sature L: Soit $u \equiv v$, alors $u \in L$ implique $v \in \phi^{-1}(\phi(L))$ et du coup $v \in L$ (car L saturé par ϕ), et inversement.

Monoïde syntaxique

Définition : Monoide syntaxique

Soit $L \subseteq \Sigma^*$. $M_L = \Sigma^* / \equiv_L$.

Théorème :

Soit $L \subseteq \Sigma^*$.

- M_L est le monoïde de transitions de l'automate minimal de L.
- M_L divise (est quotient d'un sous-monoïde) tout monoïde qui reconnaît L.

On peut effectivement calculer le monoïde syntaxique d'un langage reconnaissable.

Preuve: Monoïde syntaxique

Soit $\mathcal{A} = \langle Q, \delta, i, F \rangle$ l'automate DC minimal qui reconnaît L.
Son monoïde de transistions ϕ engendre une congruence $\equiv_{\mathcal{A}}$ saturant L par $u \equiv_{\mathcal{A}} v$ ssi $\phi(u) = \phi(v)$ (voir preuve précédente). On montre alors que $\equiv_{\mathcal{A}} = \equiv_L$.

Rappel: $\quad u \equiv_{\mathcal{A}} v \iff \forall q \in Q : \delta(q, u) = \delta(q, v)$
$\equiv_L$ est plus grossière que $\equiv_{\mathcal{A}}$, on montre le sens invers: si $u \equiv_L v$ alors $u \equiv_{\mathcal{A}} v$.

$$\begin{aligned} u \equiv_L v &\implies \forall x, y : xuy \in L \Leftrightarrow xvy \in L \\ &\implies \forall x, y : \delta(i, xuy) \in F \Leftrightarrow \delta(i, xvy) \in F \\ &\implies \forall x, y : \delta(\delta(\delta(i, x), u), y) \in F \Leftrightarrow \delta(\delta(\delta(i, x), v), y) \in F \end{aligned}$$

Puisque tout état dans $\mathcal{A}$ est accessible, on peut trouver, pour tout état q, un mot x avec $\delta(i, x) = q$. Du coup, ce dernier implique:

$$\forall q, y : \delta(\delta(q, u), y) \in F \Leftrightarrow \delta(\delta(q, v), y) \in F$$

Pour un q quelconque, notons $q' := \delta(q, u)$ et $q'' = \delta(q, v)$. Alors

$$\forall y : \delta(q', y) \in F \Leftrightarrow \delta(q'', y) \in F$$

du coup $\mathcal{L}(\mathcal{A}, q') = \mathcal{L}(\mathcal{A}, q'')$ et $q' \sim q''$ (équivalence de Nérode).
Or, $\mathcal{A}$ est minimal, du coup $\mathcal{A} = \mathcal{A}/\sim$ et $q' = q''$. Du coup $u \equiv_{\mathcal{A}} v$.

Automate bidirectionnel (boustrophédon)

Définition :

$\mathcal{A} = \langle Q, T, I, F \rangle$ avec $T \subseteq Q \times \Sigma \times Q \times \{L, R\}$;
automate (non-déterministe) qui peut aller à gauche ou à droite.

Configurations de $\mathcal{A}$: éléments de $\Sigma^* Q \Sigma^*$.

Calcul de $\mathcal{A}$, pour $u, v \in \Sigma^*$ quelconque:

- $uqav \to uaq'v$ si $\langle q, a, q', R \rangle \in T$
- $ubqav \to uq'bav$ si $\langle q, a, q', L \rangle \in T$, pour tout $b \in \Sigma$.

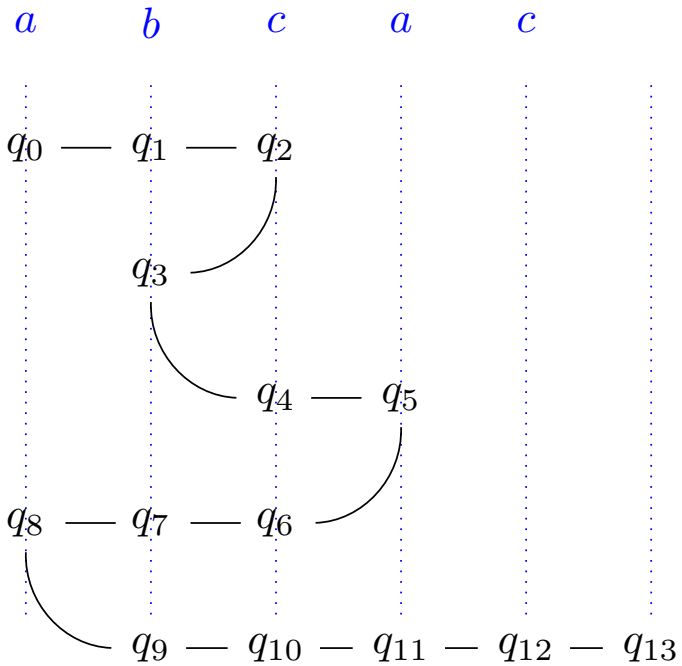
$\mathcal{A}$ accepte un mot u si $qu \to^* uq'$, pour un $q \in I$ et $q' \in F$.

Exemple :

On peut construire un automate bidirectionnel avec $\mathcal{O}(n)$ états qui accepte $\Sigma^* a \Sigma^n$.

Calcul d'un automate bidirectionnel

Le calcul d'un automate bidirectionnel peut serpenter:



Observation 1: Le calcul peut être décomposé en n 'tours', une tour par lettre plus une tour finale.

Observation 2: S'il existe un calcul acceptant, il en existe un avec hauteur de tours borné par $|Q|$.

Équivalence des automates bidirectionnels

Théorème :

$L \subseteq \Sigma^*$ est reconnaissable ssi L accepté par un automate bidirectionnel.

Preuve (idée) :

- Trivial dans le sens unidirectionnel $\rightarrow$ bidirectionnel.
- Dans l'autre sens: Étant donné un automate bidirectionnel $\mathcal{A}$, on construit un automat (unidirectionnel) non-déterministe avec $\mathcal{O}(n^n)$ états qui dévine les tours. La relation de transitions assure que deux tours consécutives vont bien ensemble, les états finaux sont les tours de hauteur 1 avec état final.