Automates Avancés et Applications

Auteur
Yago Iglesias

2 octobre 2025

Table des matières

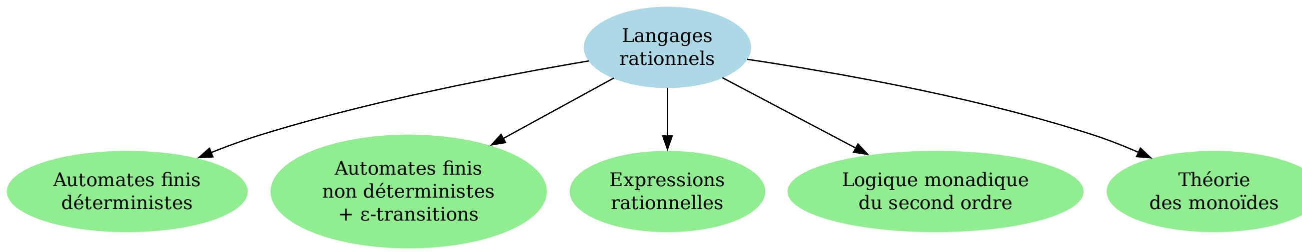
1 Introduction

Ce document est un recueil de notes du cours d'Automates Avancés et Applications de niveau M1. Il est basé sur les cours de Mme. Daniela Petrisan et M. Roberto Mantaci à Université Paris Cité, cependant toute erreur ou inexactitude est de ma responsabilité.

Ce document a été rédigé principalement par Yago Iglesias, mais tout contributeur peut être retrouvé dans la section contributeurs du répertoire GitHub. Un remerciement particulier est adressé à Erin Le Boulc'h pour sa participation active à la rédaction et correction de ce document.

Dans ces notes, les notions sont reliées à leurs définitions grâce à la librairie LaTeX knowledge. La lecture en version électronique est donc recommandée afin de profiter pleinement des liens interactifs.

2 Langages Rationnels



2.1 Définitions de base

Définition 2.1 (alphabet)**.** Un *alphabet* est un ensemble fini de lettres ou de symboles.

Définition 2.2 (mot)**.** Un *mot* sur un alphabet Σ est une séquence de lettres de Σ. On écrit $w = w_1 w_2 \dots w_n$ où $w_i \in \Sigma, \forall i \in \{1, \dots, n\}$. La longueur d'un mot w est notée $|w| = n$, pour $w = w_1 w_2 \dots w_n$.

Notation 2.3. On note ε le mot vide et Σ^* l'ensemble des mots sur Σ.

Définition 2.4 (concaténation)**.** La *concaténation* de deux mots w et v est notée wv. Si $w = w_1 w_2 \dots w_n$ et $v = v_1 v_2 \dots v_m$, alors $wv = w_1 w_2 \dots w_n v_1 v_2 \dots v_m$.

Définition 2.5 (langage)**.** Un *langage* sur un alphabet Σ est un sous-ensemble de Σ^*.

Définition 2.6 (concaténation de langages)**.** Soient L_1, L_2 deux langages sur Σ, leur concaténation est le langage

$$L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$$

Exemple 2.1.1.

$$\Sigma = \{a, b\}, \quad L = \{a, ab, bb\}, \quad K = \{b, ab\}$$

$$LK = \{ab, abb, bbb, aab, abab, bbab\}$$

Définition 2.7 (étoile de Kleene)**.** Soit L un langage sur Σ, son **étoile de Kleene** est le langage

$$L^* = \bigcup_{n\in\mathbb{N}} L^n$$

où $L^0 = \{\varepsilon\}$ et $L^{n+1} = LL^n$.

Remarque 2.1.1. *Est-ce que $L^* = \{w^n \mid w \in L, n \in \mathbb{N}\}$?*
Non, on peut trouver un contre-exemple avec $L = \{a, b\}$.
On a bien que $ab \in L^*$, mais $ab \notin \{w^n \mid w \in L, n \in \mathbb{N}\} = \{a^n \mid n \in \mathbb{N}\} \cup \{b^n \mid n \in \mathbb{N}\}$.

Remarque 2.1.2. *Est-ce que $L(M \cap N) = LM \cap LN$?*
Non, on peut trouver un contre-exemple avec $L = \{a, ab\}$, $M = \{b\}$ et $N = \{\varepsilon\}$.
On a que $M \cap N = \emptyset$, donc $L(M \cap N) = \emptyset$.
Mais $LM = \{ab, abb\}$ et $LN = \{a, ab\}$, donc $LM \cap LN = \{ab\} \neq \emptyset$.

Exercice 2.1.1. Montrer que la concaténation de langages est distributive par rapport à l'union, *i.e.* que $L(M \cup N) = LM \cup LN, \forall L, M, N$ langages.

Démonstration.

$$\begin{aligned}
w \in L(M \cup N) &\iff \exists w_L \in L, \exists w_{M\cup N} \in M \cup N, w = w_L w_{M\cup N} \\
&\iff \exists w_L \in L, (\exists w_M \in M, w = w_L w_M \lor \exists w_N \in N, w = w_L w_N) \\
&\iff (\exists w_L \in L, \exists w_M \in M, w = w_L w_M) \lor (\exists w_L \in L, \exists w_N \in N, w = w_L w_N) \\
&\iff w \in LM \lor w \in LN \\
&\iff w \in (LM \cup LN)
\end{aligned}$$

□

2.2 Langages rationnels : définitions

Définition 2.8 (langage rationnel)**.** Soit Σ un alphabet fini, l'ensemble ERat des *expressions rationnelles* sur Σ est défini comme suit :

— $\varepsilon \in ERat$

— $\emptyset \in ERat$

— $\forall a \in \Sigma, a \in ERat$

— $\forall E, F \in ERat, E + F \in ERat$

— $\forall E, F \in ERat, EF \in ERat$

— $\forall E \in ERat, E^* \in ERat$

Définition 2.9 (sémantique des expressions rationnelles)**.** Soit $r \in$ ERat, on définit le langage $\mathcal{L}(r)$ associé à r par induction sur la structure de r :

— $\mathcal{L}(\varepsilon) = \{\varepsilon\}$

— $\mathcal{L}(\emptyset) = \emptyset$

— $\mathcal{L}(a) = \{a\}$

- $\mathcal{L}(E+F) = \mathcal{L}(E) \cup \mathcal{L}(F)$
- $\mathcal{L}(EF) = \mathcal{L}(E)\,\mathcal{L}(F)$
- $\mathcal{L}(E^*) = \mathcal{L}(E)^*$

Exemple 2.2.1. $\Sigma = \{a, b\}\,,\quad r = (a+b)^*a \in \text{ERat}$, alors $\mathcal{L}(r) = \{wa \mid w \in \Sigma^*\}$

Définition 2.10 (langage rationnel)**.** Un langage L sur un alphabet Σ est dit *rationnel* s'il existe une expression rationnelle $r \in \text{ERat}$ telle que $L = \mathcal{L}(r)$.

Exemple 2.2.2. $\{a^n \mid n \in \mathbb{N}\}$ est un langage rationnel engendré par l'expression rationnelle a^*.
Cependant, $\{a^n b^n \mid n \in \mathbb{N}\}$ n'est pas un langage rationnel.

3 Automates finis

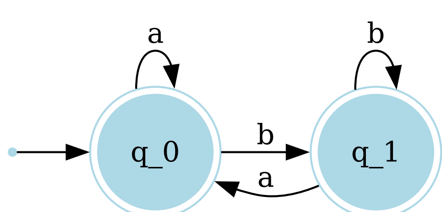
3.1 Automates finis déterministes

Définition 3.1 (automate fini déterministe)**.** Soit Σ un alphabet, un *automate fini déterministe* (AFD) est un tuple $\langle Q, q_0, F, \delta \rangle$ où

- Q est un ensemble fini d'états
- $q_0 \in Q$ est appelé l'état initial
- $F \subseteq Q$ est l'ensemble des états finaux / acceptants
- $\delta : Q \times \Sigma \to Q$ est la fonction de transition

Exemple 3.1.1. Un automate fini déterministe :

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$
- q_0 est l'état initial
- $F = \{q_0, q_1\}$
- $\delta : \begin{cases} (q_0, a) & \mapsto q_0 \\ (q_0, b) & \mapsto q_1 \\ (q_1, a) & \mapsto q_0 \\ (q_1, b) & \mapsto q_1 \end{cases}$



Définition 3.2 (lecture d'un mot par un AFD)**.** Soit $A = \langle Q, q_0, F, \delta \rangle$ un AFD qui a pour alphabet Σ. On définit la fonction δ^* par induction sur la longueur du mot w :

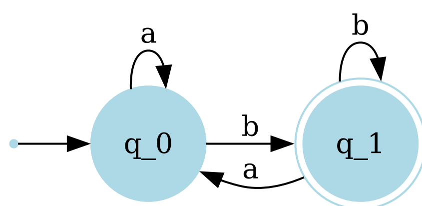
$$\begin{array}{rcl} \delta^* : Q \times \Sigma^* & \to & Q \\ (q, \varepsilon) & \mapsto & q \\ (q, wa) & \mapsto & \delta(\delta^*(q, w), a) \end{array}$$

On a alors que $\delta^*(q, w)$ est l'état atteint par A après avoir lu le mot w depuis l'état q.

Définition 3.3 (langage reconnu par un AFD)**.** Le *langage reconnu* / accepté par un AFD $A = \langle Q, q_0, F, \delta \rangle$ est le langage

$$\mathcal{L}(A) = \{w \mid w \in \Sigma^*, \delta^*(q_0, w) \in F\}$$

Exemple 3.1.2. Pour l'automate A suivant, avec $\Sigma = \{a, b\}$, le langage reconnu est $\mathcal{L}(A) = \mathcal{L}((a+b)^*b) = \mathcal{L}((a+b)^*bb^*) = \{wb \mid w \in \Sigma^*\}$.



Définition 3.4 (chemin)**.** Soit $A = \langle Q, q_0, F, \delta \rangle$ un AFD, et $p, q \in Q$, un *chemin* $p \to q$ est une suite $(p, a_0, q_1), (q_1, a_1, q_2), \ldots, (q_{n-1}, a_{n-1}, q) \in Q^n \times \Sigma^n \times Q^n$ telle que $\forall i \in \{1, \ldots, n-2\}, \delta(q_i, a_i) = q_{i+1}$ et $\delta(p, a_0) = q_1$ et $\delta(q_{n-1}, a_{n-1}) = q$.

Définition 3.5. Un automate $A = \langle Q, q_0, F, \delta \rangle$ est *accesible* si, pour tout état de l'automate, il existe un chemin partant de q_0 qui mène a lui.

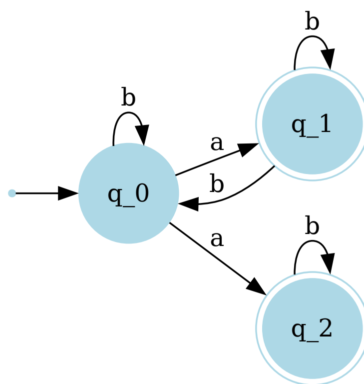
3.2 Automates finis non déterministes + ε-transitions

Dans cette partie, on traite en meme temps les cas avec et sans ε-transitions. Les additions en vert correspondent à la définitions avec ε-transitions. En ignorant cela, on retrouve la definition d'un automate fini non déterministe.

Définition 3.6 (automate fini non déterministe + ε-transitions)**.** Soit Σ un alphabet, un *automate fini non déterministe* (AFN) est un tuple $\langle Q, I, F, \delta \rangle$ où

- 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 / acceptants
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$ est la fonction de transition

Exemple 3.2.1. Un automate fini non déterministe avec $\Sigma = \{a, b\}$:



Définition 3.7 (lecture d'un mot par un AFN)**.** Soit $A = \langle Q, I, F, \delta \rangle$ un AFN qui a pour alphabet Σ. On définit la fonction δ^* par induction sur la longueur du mot w :

$$\begin{aligned} \delta^* : Q \times \Sigma^* &\rightarrow \mathcal{P}(Q) \\ (q, \varepsilon) &\mapsto \{q\} \\ (q, wa) &\mapsto \bigcup_{p \in \delta^*(q,w)} \delta(p, a) \end{aligned}$$

Définition 3.8 (langage reconnu par un AFN)**.** Le langage reconnu / accepté par un AFN $A = \langle Q, I, F, \delta \rangle$ est le langage

$$\mathcal{L}(A) = \{w \in \Sigma^* \mid \exists q_0 \in I, \delta^*(q_0, w) \cap F \neq \emptyset\}$$

3.3 Déterminisation d'un AFN

Soit $A = \langle Q, I, F, \delta \rangle$ un AFN, on considère l'AFD $A' = \langle \mathcal{P}(Q), I, F', \delta' \rangle$ où

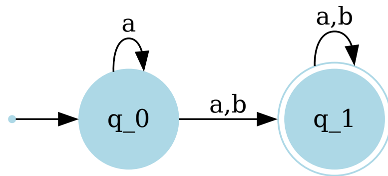
— $F' = \{q \in \mathcal{P}(Q) \mid q \cap F \neq \emptyset\}$

— $$\begin{aligned} \delta' : \mathcal{P}(Q) \times \Sigma &\rightarrow \mathcal{P}(Q) \\ (Q, a) &\mapsto \bigcup_{p \in Q} \delta(p, a) \end{aligned}$$

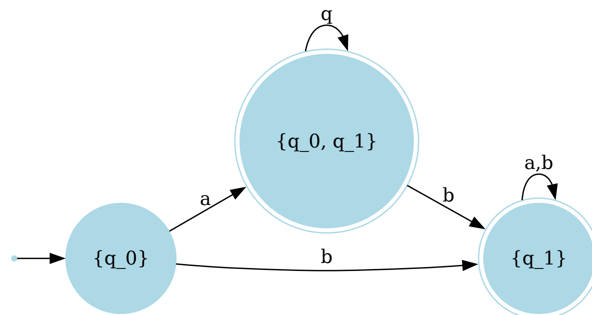
Ce processus est appelé *déterminisation* d'un AFN et nous permet de transformer un AFN en un AFD équivalent.

Exemple 3.3.1. Déterminisation d'un automate non déterministe :

Automate non déterministe :



Automate déterminisé :

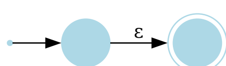


Théorème 3.9. *Soit A un automate fini non déterministe avec des ε-transitions, alors il existe un automate fini déterministe A', tel que $\mathcal{L}(A) = \mathcal{L}(A')$.*

3.4 Équivalence entre expressions rationnelles et automates finis déterministes

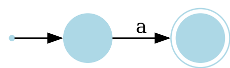
Théorème 3.10 (Injection des expressions rationnelles vers les automates finis déterministes)**.** *Soit $r \in ERat$, alors il existe un automate fini déterministe $N(r)$ tel que $\mathcal{L}(A) = \mathcal{L}(r)$.*

Démonstration. Nous allons construire un tel automate par induction sur la structure de r cependant la preuve du fait que cet automate reconnaît le langage associé à r est omise. Cette construction est appelée **construction de Thompson**.

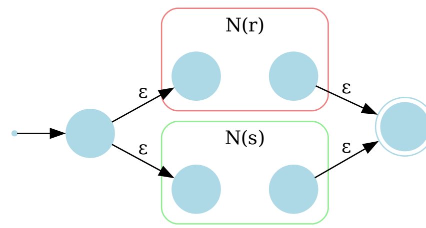


$N(\varepsilon)$

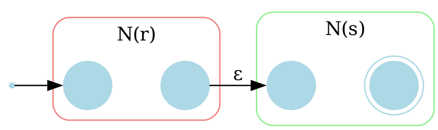
$N(\emptyset)$



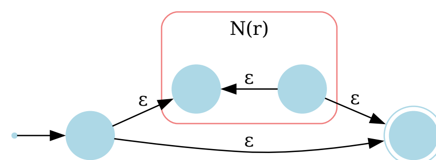
$N(a), a \in \Sigma$



$N(r+s)$



$N(rs)$



$N(r^*)$

En combinant ces constructions, on peut construire un automate fini non déterministe pour n'importe quelle expression rationnelle qui peut être determinisé pour obtenir un automate fini déterministe.

□

4 Minimisation d'automates

4.1 Morphismes d'automates

Définition 4.1 (Morphisme d'automates)**.** Soit $A = \langle Q, q_0, F, \delta \rangle$ et $A' = \langle Q', q_0', F', \delta' \rangle$ sur un alphabet Σ. Alors un *morphisme d'automates* $\varphi : A \to A'$ est une fonction $\varphi : Q \to Q'$ telle que

1. $\varphi(q_0) = q_0'$
2. $\forall q \in Q, \quad \varphi(q) \in F' \iff q \in F$
3. $\forall q \in Q, \forall a \in \Sigma, \quad \delta'(\varphi(q), a) = \varphi(\delta(q, a))$ *i.e.* $\varphi \circ \delta_a = \delta_a' \circ \varphi$

$$\begin{tikzcd}
Q \arrow[r, "\delta_a"] \arrow[d, "\varphi"'] & Q \arrow[d, "\varphi"] \\
Q' \arrow[r, "\delta_a"] & Q'
\end{tikzcd}$$

Exercice 4.1.1. Soit $\varphi : A \to A'$ un morphisme d'automates déterministes, alors $\mathcal{L}(A) = \mathcal{L}(A')$

Démonstration.

$$\begin{aligned}
w \in \mathcal{L}(A) &\iff \delta^*(q_0, w) \in F \\
&\iff \varphi(\delta^*(q_0, w)) \in F' \quad \textit{(par 1)} \\
&\iff \delta^*(\varphi(q_0), w) \in F' \quad \textit{(par 3)} \\
&\iff \delta^*(q_0', w) \in F' \quad \textit{(par 2)} \\
&\iff w \in \mathcal{L}(A')
\end{aligned}$$

□

Définition 4.2 (Quotient d'un automate)**.** Soit $L \subseteq \Sigma^*$ et $w \in \Sigma^*$. Le *quotient* $w^{-1}L$ est défini par

$$w^{-1}L = \{u \in \Sigma^* \mid wu \in L\}$$

Définition 4.3. Soit A $= \langle Q, q_0, F, \delta \rangle$ sur l'alphabet Σ et $q \in Q$ alors on définit

$$L_q = \{w \in \Sigma^* \mid \delta^*(q, w) \in F\}$$

Lemme 4.4. *Soit $A = \langle Q, q_0, F, \delta \rangle$ un automate fini déterministe, $q \in Q$ et $w \in \Sigma^*$. Si $\delta^*(q_0, w) = q$ alors $L_q = w^{-1}\mathcal{L}(A)$*

Démonstration.

$$\begin{aligned}
u \in L_q &\iff \delta^*(q, u) \in F \\
&\iff \delta^*(\delta^*(q_0, w), u) \in F \\
&\iff \delta^*(q_0, wu) \in F \\
&\iff wu \in \mathcal{L}(A) \\
&\iff u \in w^{-1}\mathcal{L}(A)
\end{aligned}$$

□

Corollaire 4.5. *Si $L \subseteq \Sigma^*$ est un langage régulier alors l'ensemble $\left\{w^{-1}L \mid w \in \Sigma^*\right\}$ est fini.*

Démonstration. Si $L = \mathcal{L}(A)$, avec A $= \langle Q, q_0, F, \delta \rangle$, alors $\left|\left\{w^{-1}L \mid w \in \Sigma^*\right\}\right| \leq |Q|$. Car $\forall w \in \Sigma^*$, $w^{-1}L$ est le langage accepté par A à partir de l'état $\delta^*(q_0, w)$. □

Corollaire 4.6. *Si $\langle Q, I, F, \delta \rangle$ accepte un langage L alors $|Q| \geq \#$ quotients de L*

Définition 4.7 (Automate des quotients)**.** Soit $L \subseteq \Sigma^*$ un langage régulier, soit $Q = \left\{w^{-1}L \mid w \in \Sigma^*\right\}$ et $\langle Q, q_0, F, \delta \rangle$. Alors l'*automate des quotients* de L est défini par

- L'état initial est $\varepsilon^{-1}L$
- $F = \left\{w^{-1}L \mid w \in L\right\}$
- $\delta(w^{-1}L, a) = (wa)^{-1}L$

Remarque 4.1.1. *Si $w^{-1}L = (w')^{-1}L$, alors $(wa)^{-1}L = (w'a)^{-1}L$.*

Soit $w, w' \in \Sigma^*$ tel que $w^{-1}L = (w')^{-1}L$. Soit $a \in \Sigma$ et $u \in \Sigma^*$,

$$\begin{aligned} u \in (wa)^{-1}L &\iff wau \in L \\ &\iff au \in w^{-1}L \\ &\iff au \in w'^{-1}L \\ &\iff w'au \in L \\ &\iff (w'a)u \in L \\ &\iff u \in (w'a)^{-1}L \end{aligned}$$

Donc la fonction de transition δ de l'automate des quotients est bien définie.

Proposition 4.8. *Étant donné un automate A qui accepte le langage L, l'automate des quotients accepte aussi L et il est minimal parmi les automates acceptant L.*

Démonstration. $\delta^*(L, w) = w^{-1}L$ et donc $\underbrace{\delta^*(L, w) \in F}_{w \in \mathcal{L}(A)} \iff w \in L$

Et ainsi, par le corollaire 4.6, l'automate des quotients est minimal. □

Lemme 4.9. *Soit $L \subseteq \Sigma^*$ un langage régulier. Soit A_L l'automate des quotients de L et soit B un autre automate acceptant L. Soit reach(B) un sous-automate accessible de B. Alors on a un morphisme surjectif (un quotient) d'automates reach$(B) \twoheadrightarrow A_L$.*

$$\begin{tikzcd} & reach(B) \arrow[dl, two heads] \arrow[dr, hook] & \\ A_L & & B \end{tikzcd}$$

Si $B = < F, t_0, F, \delta >$ alors reach$(B) = < F', t_0, F', \delta' >$ où

- $F' = \{t \in F \mid \exists w \in \Sigma^*, \delta^*(t_0, w) = t\}$
- $F' = F \cap F'$
- $\delta' = \delta \cap (F' \times \Sigma \times F')$

Exercice 4.1.2. On définit $\varphi : \text{reach}(B) \to A$.

Si $t \in F', \exists w \in \Sigma^*$ tel que $t = \delta^*(t_0, w)$ On définit $\varphi(t) = w^{-1}L$

Montrer que φ est bien définie, *i.e.* , $\varphi(t) = w'^{-1}L \implies w^{-1}L = w'^{-1}L$.

Démonstration. Vérifions φ surjective.

Si $w \in \Sigma^*$ et $w^{-1}L$ est un état de A_L alors $\varphi(\delta^*(t_0, w)) = w^{-1}L$

Si $|B| = |A_L|$, alors B est un automate accesible (Sinon $|A| \leq |\text{reach}(B)| < |B|$ ↯).

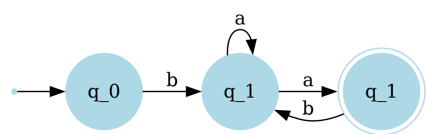
Donc $B = \text{reach}(B)$

De plus, le morphisme φ est une bijection et φ^{-1} est un morphisme d'automates. Donc $B \cong A_L$. □

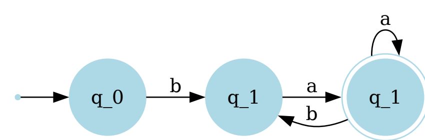
4.2 Minimisation d'automates

Rappel 4.2.1. *Pour tout langage rationnel L, il existe un unique (à isomorphisme près) automate déterministe minimal (avec le plus petit nombre possible d'états) qui reconnait L.*

Remarque 4.2.1. *Ce n'est pas vrai pour les automates non-déterministes.*



$(ba^+)^+$



$(ba^+)^+$

L'automate des résiduels de L est l'automate minimal reconnaissant L.

4.3 Algorithmes de minimisation

Nous allons voir trois algorithmes de minimisation différents :

- Algorithme de Brzozowski
- Algorithme de Moore
- Algorithme de Hopcroft

4.3.1 Algorithme de Brzozowski

Définition 4.10. Soit $A = \langle Q, q_0, F, \delta \rangle$ un automate déterministe, son *automate miroir* est l'automate $A' = \langle Q, F, \{q_0\}, \delta' \rangle$ avec δ' définie par :

$$\forall q, q' \in Q, a \in \Sigma,\ \delta(q, a) = q' \iff q \in \delta'(q', a)$$

Proposition 4.11. *Si A est un automate déterministe et accesible, et $A^\sim = det(mirr(A))$ est l'automate obtenu en déterminisant l'automate miroir de A. Alors $A^\sim$ est l'automate minimal qui reconnait $\widetilde{\mathcal{L}(A)}$.*

Démonstration. Soit $A = (Q, q_0, F)$, $L = \mathcal{L}(A)$. $\mathrm{mirr}(A) = (Q, F, q_0)$, $M = \mathcal{L}(\mathrm{mirr}(A)) = \widetilde{L}$.

On note $X \cdot u$ et $\delta(X, u)$, X étant un ensemble d'états et u un mot, l'ensemble $\bigcup_{q\in X} \delta(q, u)$.

Nous allons montrer que $A^\sim$ est l'automate minimal pour M. Pour cela, il suffit de montrer que $A^\sim$ est isomorphe à l'automate des résiduels de M.

Pour cela, il faut montrer que si u et v sont deux mots quelconques tel que $u^{-1}M = v^{-1}M$, alors $F \cdot u = F \cdot v$ dans $A^\sim$.

Soit p un état de $F \cdot u$. Puisque A est accesible, il existe un chemin de $q_0 \to p$ dans A, et donc il existe un chemin $p \to q_0$ dans $\mathrm{mirr}(A)$. Soit w l'étiquette de ce chemin, alors le mot uw est l'étiquette d'un chemin réussi de $\mathrm{mirr}(A)$. En conséquence, $uw \in M$.

Mais $uw \in M \iff w \in u^{-1}M \iff w \in v^{-1}M \iff vw \in M$.

Donc vw est l'étiquette d'un chemin réussi Γ dans $\text{mirr}(A)$. Étant donné que A est déterministe, tout chemin aboutissant dans q_0 et dont l'étiquette a w comme suffixe doit passer par p.

On en déduit que $p \in Tv$. □

Corollaire 4.12. *Soit A un automate, alors l'automate $det(mirr(det(mirr(A))))$ est l'automate minimal pour $\mathcal{L}(A)$.*

Démonstration. En effet,

$$\underbrace{\text{det}(\text{mirr}(\underbrace{\underbrace{\text{det}(\text{mirr}(A))}_{\text{Un automate déterministe et accesible qui reconnait } \widetilde{\mathcal{L}(A)}}))}_{\text{la proposition précédente garantit que cet automate est minimal pour } \widetilde{\widetilde{\mathcal{L}(A)}}=\mathcal{L}(A)}}$$

□

Complexité. *La complexité de l'algorithme est en $O(2^n)$. En effet, la dernière détermination exécutée travaille sur un automate de taille au plus 2^n et produit un automate de taille 2^n. Sa complexité reste aussi en $O(2^n)$.*

Remarque 4.3.1. *Alors que d'autres algorithmes, tel que Moore, ont une complexité polynomiale, Brzozowski à différence des autres, ne nécessite pas que l'automate donné soit déterministe.*

4.3.2 Algorithme de Moore

Définition 4.13 (Congruence d'automates)**.** Soit $A = (Q, q_0, F, S)$ un automate déterministe et soit $\sim$ une relation d'équivalence définie sur l'ensemble Q. On dit que $\sim$ est une *congruence* si $\sim$ satisfait les conditions suivantes :

1. Compatibilité aves les transitions : Si $q \sim q'$ alors $\forall a \in \Sigma, \delta(q, a) \sim \delta(q', a)$
2. Saturation de A : Si $q \sim q'$ alors $q \in F \iff q' \in F$

Définition 4.14. Si $A = \langle Q, q_0, F, \delta \rangle$ est un AFD et $\sim$ une congruence définie sur Q. On définit l'automate quotient :

$$A/\sim = (Q', q'_0, F', \delta')$$

avec

— $Q' = \{[q] \mid q \in Q\}$ (chaque état est étiqueté par une classe d'équivalence)

— $q'_0 = [q_0]$

— $F' = \{[q] \mid q \in F\}$

— $\delta'([q], a) = [p]$ si et seulement si $p \in [qa]$

Proposition 4.15. *Si A est un automate déterministe et $\sim$ une congruence sur A, alors $\mathcal{L}(A) = \mathcal{L}(A/\sim)$.*

Définition 4.16 (*Congruence de Nérode*)**.** Si $A = (Q, \{q_0\}, F, \delta)$ un AFD, $\forall q, q' \in Q$ on définit $q \cong q' \iff L_q = L_{q'}$.

Rappel 4.3.1. $L_q = \{w \mid \delta^*(q, w) \in F\}$. *Donc*

$$\begin{aligned} q \cong q' \iff L_q = L_{q'} &\iff \forall w \in \Sigma^*, w \in L_q \text{ si et seulement si } w \in L_{q'} \\ &\iff \forall w, \delta^*(q, w) \in F \text{ si et seulement si } \delta^*(q', w) \in F \end{aligned}$$

On en déduit que

$$q \ncong q' \iff \exists w, \delta^*(q,w) \in F \text{ et } \delta^*(q',w) \notin F \tag{1}$$

où $\delta^*(q,w) \notin F$ *et* $\delta^*(q',w) \in F$ *sont deux états non équivalents, dits* séparables.

Si q et q' sont séparables et w est un mot qui satisfait 1, on dira que w *sépare* q et q'.
Pour calculer la congruence de Nérode on introduit une famille de congruences :

$$\cong_i, i \in \mathbb{N}, q \cong_i q' \iff \forall w, |w| \leq i \implies \delta^*(q,w) \in F \iff \delta^*(q',w) \in F$$

Définition 4.17. Définition alternative

— $q \cong_0 q'$ si et seulement si $q \in F \iff q' \in F$

— $q \cong_{i+1} q'$ si et seulement si $q \cong_i q'$ et $\forall a \in \Sigma, \delta(q,a) \cong_i \delta(q',a)$

Remarque 4.3.2. *Par définition* $\cong_{i+1}$ *induit une partition de* Q *au moins aussi fine que celle induite par* $\cong_i$*.*

Les classes de $\cong_{i+1}$ *sont obtenues en partitionnant des classes de* $\cong_i$*.*

Remarque 4.3.3. *Puisque* Q *est fini, le processus de partitionnement doit se stabiliser. Autrement dit,*

$$\exists k \in \mathbb{N}^*, \textit{tel que } \cong_k = \cong_{k+j} \forall j \in \mathbb{N}^*$$

Proposition 4.18. *Si* $k \in \mathbb{N}$ *tel que* $\cong_k = \cong_{k+j}$ $\forall j \in \mathbb{N}$*, alors* $\cong_k = \cong$

Démonstration. Soit $q, q' \in Q$, $q \cong_k q'$, montrer que $L_q = L_{q'}$.

Soit $w \in L_q$.

— Si $|w| \leq k$ alors $\delta^*(q',w) \in F$ car $q \cong_k q'$

— Si $|w| > k$ alors $\exists i$ tel que $|w| = k + i$ et donc comme $\cong_k = \cong_{k+i}$ on a que $\delta^*(q',w) \in F$

Et donc $w \in L_{q'}$ et ainsi $L_q \subseteq L_{q'}$. On peut faire la meme preuve pour montrer que $L_{q'} \subseteq L_q$. On en déduit que que $L_q = L_{q'}$. □

Définition 4.19 (Algorithme de Moore)**.** L'algorithme consiste à créer un automate à partir des classes d'équivalence sur $\cong$. Les états sont les classes d'équivalence, on obtient les transitions en regardant le comportement d'un représentant de la classe et les états finaux sont les classes qui ont un représentant qui est un état final.

Complexité. *L'algorithme effectue* n *étapes et chaque étape dépense* $O(n)$*, donc sa complexité est en* $O(n^2)$ *(au pire).*

En fait, en moyenne $O(n \log n)$ *et même* $O(n \log \log n)$*, [Dav10]*

5 Monoïdes

Définition 5.1 (Monoïde)**.** Un monoïde est un tuple $(M, \times, 1)$ où M est un ensemble, $\times : M \times M \to M$ est une operation binaire sur M, $1 \in M$ tel que

1. $\times$ est une opération associative ($\forall x, y, z \in M, x \times (y \times z) = (x \times y) \times z$)
2. 1 est un élément neutre pour $\times$, c'est-à-dire , $\forall x \in M, 1 \times x = x \times 1 = x$

Exemple 5.0.1.

— $(\mathbb{N}, +, 0)$
— $(\Sigma^*, \cdot, \varepsilon)$
— $(\mathbb{Z}, +, 0)$ est un monoïde, mais aussi un groupe car tout élément possède un symétrique.
— Si Q est un ensemble, alors $(Q^Q, \circ, \mathrm{id})$

Définition 5.2 (Morphisme de monoïdes)**.** Soit $(M, \times_M, 1_M)$ et $(N, \times_N, 1_N)$ des monoïdes. Une fonction $h : M \to N$ est appelée un morphisme de monoïdes si et seulement si

1. $h(1_M) = 1_N$ (*h préserve l'élément neutre*)
2. $\forall x, y \in M, h(x \times_M y) = h(x) \times_N h(y)$ (*h préserve la multiplication*)

Proposition 5.3. *Soit Σ un ensemble et $(M, \cdot, 1)$ un monoïde. Soit $f : \Sigma \to M$ une fonction. Il existe un unique morphisme de monoïdes $\bar{f} : \Sigma^* \to M$ tel que $\bar{f}(a) = f(a), \forall a \in \Sigma$.*

$$\begin{tikzcd}
\Sigma \arrow[r, hook] \arrow[d, "f"] & \Sigma^* \arrow[dl, dashed, "\exists! \bar{f}"] \\
M &
\end{tikzcd}$$

TODO: Add circular arrow

Démonstration.

— Existence :
Soit $w \in \Sigma^*$ de la forme $w = w_1 w_2 \ldots w_n$ où $w_i \in \Sigma$.
Soit $\bar{f}(w) =^{def} f(w_1) \cdot \ldots \cdot f(w_n)$ et $\bar{f}(\varepsilon) = 1_M$. Alors pour $a \in \Sigma$, $\bar{f}(a) = f(a)$.
Si $w = w_1 w_2 \ldots w_n$ et $v = v_1 v_2 \ldots v_m \in \Sigma^*$.

$$\begin{aligned}
\bar{f}(wv) = \bar{f}(w_1 w_2 \ldots w_n v_1 v_2 \ldots v_m) &= f(w_1) \cdot \ldots \cdot f(w_n) \cdot f(v_1) \cdot \ldots \cdot f(v_m) \\
&= (f(w_1) \cdot \ldots \cdot f(w_n)) \cdot (f(v_1) \cdot \ldots \cdot f(v_m)) \\
&= \bar{f}(w_1 w_2 \ldots w_n) \cdot \bar{f}(v_1 v_2 \ldots v_m) \\
&= \bar{f}(w) \cdot \bar{f}(v)
\end{aligned}$$

Donc $\forall w, v \in \Sigma^*, \bar{f}(wv) = \bar{f}(w)\bar{f}(v)$ et $\bar{f}(\varepsilon) = 1_M$ et ainsi, $\bar{f}$ est un morphisme de monoïdes.

— Unicité :
Si $h : \Sigma^* \to M$ est un morphisme de monoïdes tel que $h(a) = f(a), \forall a \in \Sigma^*$. Alors $h(\varepsilon) = 1_M$.
Comme h préserve la multiplication, on a $h(a_1 a_2 \ldots a_n) = h(a_1) \ldots h(a_n) = f(a_1) \ldots f(a_n) = \bar{f}(a_1 a_2 \ldots a_n)$.
Et donc $h = \bar{f}$

□

On dit que Σ^* est librement engendré sur Σ.

Définition 5.4 (Librement engendré)**.** H est librement engendré sur Σ si

$$\forall f : \Sigma \to M \text{ où } M \text{ est un monoïde}, \exists! \bar{f} : H \to M \text{ et } i : \Sigma \to H, \bar{f} \circ i = f$$

$$\begin{tikzcd} \Sigma \arrow[r, hook, "i"] \arrow[d, "f"] & H \arrow[dl, dashed, "\exists! \bar{f}"] \\ M & \end{tikzcd}$$

Et on a donc que $H \cong \Sigma^*$.

Exemple 5.0.2. Montrons que $(\mathbb{N}, +, 0)$ est librement engendré sur $\{1\}$.

$$\begin{tikzcd} \{1\} \arrow[r, hook] \arrow[d, "f"] & (\mathbb{N}, +, 0) \arrow[dl, dashed, "\exists! \bar{f}"] \\ (M, \cdot, 1_M) & \end{tikzcd}$$

Alors on a que $f(1) = m \in M$ et donc

$$f(n) = f(\underbrace{1 + \ldots + 1}_{n \text{ fois}}) = \underbrace{f(1) \cdot \ldots \cdot f(1)}_{n \text{ fois}} = \underbrace{m \cdot \ldots \cdot m}_{n \text{ fois}} = m^n$$

Donc, si on pose $i : \{1\} \to (\mathbb{N}, +, 0)$ tel que $f(1) = 1$, alors quelque soit le choix de m, *i.e.* , quelque soit la fonction f choisie (car elle dépend juste du choix m), alors on peut toujours poser $\bar{f}(\mathbb{N}, +, 0)$ avec $f(0) = 1_M$ et $f(1) = m$ qui vérifie bien que $\bar{f} \circ i = f$. Ainsi, $(\mathbb{N}, +, 0)$ est librement engendré sur $\{1\}$.

Remarque 5.0.1. $\mathbb{N} \cong \{1\}^*$

5.1 Reconnaissance par monoïdes

Définition 5.5 (Reconnaissance par monoïdes)**.** Soit Σ un alphabet fini. Soit $L \subseteq \Sigma^*$ un langage et soit $\varphi : \Sigma^* \to M$ un morphisme de monoïdes où $(M, \circ, 1_M)$ est un monoïde fini. On dit que φ reconnait L si et seulement si il existe $P \subseteq M$ tel que $L = \varphi^{-1}(P)$, c'est-à-dire $L = \{w \in \Sigma^* \mid \varphi(w) \in P\}$.

Exercice 5.1.1. φ reconnaît L si et seulement si $L = \varphi^{-1}(\varphi(L))$.

Démonstration.

— $\boxed{\Leftarrow}$
  Soit $P = \varphi(L) \subseteq M$, alors on a que $\varphi^{-1}(P) = \varphi^{-1}(\varphi(L)) = L$ et donc φ reconnait L.

— $\boxed{\Rightarrow}$
  On sait qu'il existe P tel que $L = \{w \in \Sigma^* \mid \varphi(w) \in P\}$. Regardons ce qu'est $\varphi(L)$. On a que $\varphi(L) = \{\varphi(w) \mid w \in \Sigma^*, \varphi(w) \in P\} = P$. Et donc on a bien $\varphi^{-1}(\varphi(L)) = \varphi^{-1}(P) = L$.

□

Proposition 5.6. *Soit $L \subseteq \Sigma^*$ un langage sur l'alphabet Σ. Alors L est reconnaissable si et seulement si L est reconnu par un morphisme de monoïdes $\varphi : \Sigma^* \to M$ où M est fini.*

Démonstration.

— $\boxed{\Leftarrow}$
  Soit $\varphi : \Sigma^* \to M$ un morphisme de monoïdes avec M fini qui reconnait le langage L. $\exists P \subseteq M, L = \varphi^{-1}(P)$.
  Soit A l'automate suivant : $\langle M, 1_M, P, \delta \rangle$ où

$$\begin{array}{rcl} \delta : M \times \Sigma & \to & M \\ (m,a) & \mapsto & m \cdot \varphi(a) \end{array}$$

On peut montrer par induction sur la longueur du mot que

$$\begin{array}{rcl} \delta^* : M \times \Sigma^* & \to & M \\ (1_M, w) & \mapsto & \varphi(w) \end{array}$$

$$\begin{array}{rcl} A \text{ accepte le mot } w & \iff & \delta^*(1_M, w) \in P \iff \varphi(w) \in P \\ & \iff & w \in \varphi^{-1}(P) \iff w \in L. \end{array}$$

Donc A accepte le langage L.

— $\boxed{\Rightarrow}$
Soit $A = \langle Q, q_0, F, \delta \rangle$ un automate déterministe complet qui accepte L. Soit

$$\begin{array}{rcl} \varphi : \Sigma^* & \to & Q^Q \quad \big((Q^Q, \circ, id_Q) \textit{ est bien un monoïde}\big) \\ w & \mapsto & (q \mapsto \delta^*(q,w)) \end{array}$$

On a bien :

- Q^Q est un monoïde fini.
- $\varphi(\varepsilon)(q) = \delta^*(q, \varepsilon) = q$, donc $\begin{array}{rcl} \varphi(\varepsilon) : Q & \to & Q \\ q & \mapsto & q \end{array}$ est la fonction identité sur Q et donc φ préserve l'identité (élément neutre).
- $\varphi(w) \circ \varphi(w') = \varphi(ww'),\ \forall w, w' \in \Sigma^*$ car $\delta^*(\delta^*(q,w), w') = \delta^*(q, ww')$. Donc φ préserve la multiplication.
  Soit $P \subseteq Q^Q$ défini par $P = \{f : Q \to Q \mid f(q_0) \in F\}$.

$$\begin{array}{rcl} \varphi^{-1}(P) & = & \{w \in \Sigma^* \mid \varphi(w) \in P\} \\ & = & \{w \in \Sigma^* \mid \varphi(w)(q_0) \in F\} \\ & = & \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\} \\ & = & \{w \in \Sigma^* \mid w \text{ est accepté par } A\} \\ & = & L \end{array}$$

Donc L est reconnu par φ.

□

Définition 5.7. On va appeler $\varphi(\Sigma^*) \subseteq Q^Q$ le **monoïde de transition de l'automate** A.

Définition 5.8 (Congruence)**.** Soit $(M, \cdot, 1_M)$ un monoïde. Une congruence sur M est une relation d'équivalence $\sim \subseteq M \times M$ telle que

$$\forall m, m' \in M,\ m \sim m' \iff \forall n, r \in M, n \cdot m \cdot r \sim n \cdot m' \cdot r$$

Définition 5.9 (Congruence syntaxique d'un langage)**.** Soit $L \subseteq \Sigma^*$ un langage. Soit $\sim_L \subseteq \Sigma^* \times \Sigma^*$ la relation d'équivalence définie par :

$$w \sim_L w' \text{ si et seulement si } \forall u, v \in \Sigma^*, uwv \in L \iff uw'v \in L$$

Proposition 5.10. *$\sim_L$ est une congruence sur Σ^*.*

Démonstration. Soit $m, m' \in \Sigma^*$ tels que $m \sim_L m'$, c'est-à-dire $\forall u, v \in \Sigma^*, umv \in L \iff um'v \in L$. Soit $n \in \Sigma^*$, montrons que $nmr \sim_L nm'r$ c'est-à-dire $\forall u, v \in \Sigma^*, u(nmr)v \in L \iff u(nm'r)v \in L$. Soit $u, v \in \Sigma^*$, posons $u' = un, v' = rv$, alors, on veut montrer que $u'mv' \in L \iff u'm'v' \in L$, ce qui est vrai car $m \sim_L m'$. □

Proposition 5.11. *Si $(M, \cdot, 1_M)$ est un monoïde et $\sim \subseteq M \times M$ est une congruence sur M, alors $M/\sim = \{[m] \mid m \in M\}$, où $[m]$ note la classe d'équivalence de m par rapport à $\sim$.*

$M/\sim$ est un monoïde et $\begin{array}{rcl} h : M & \to & M/\sim \\ m & \mapsto & [m] \end{array}$ *est un morphisme de monoïdes.*

Démonstration. $(M/\sim, \cdot, [1_M])$ est un monoïde où $[m][m'] = [mm']$.

À montrer que cette multiplication est bien définie, c'est-à-dire si $m_1 \sim m_2$ et $m'_1 \sim m'_2$ alors $m_1m'_1 \sim m_2m'_2$:

$$\begin{array}{rcll} m_1 & \sim & m_2 & \\ m'_1 & \sim & m'_2 & \\ 1_M \cdot m_1 \cdot m'_1 & \sim & 1_M \cdot m_2 \cdot m'_1 & (\textit{car} \sim \textit{est une congruence}) \\ m_1 \cdot m'_1 & \sim & m_2 \cdot m'_1 & \\ \text{Mais } m_2 \cdot m'_1 & \sim & m_2 \cdot m'_2 & (\textit{car} \sim \textit{est une congruence}) \\ \implies m_1 \cdot m'_1 & \sim & m_2 \cdot m'_2 & \end{array}$$

Donc la multiplication sur $M/\sim$ est bien définie.
$[1_M]$ est un élément neutre car $[m] \cdot [1_M] = [m \cdot 1_M] = [m]$ et pareil pour $[1_M] \cdot [m] = [m]$.
h est un morphisme de monoïdes :

- $h(1_M) = [1_M]$
- $h(m \cdot n) = [m \cdot n] = [m] \cdot [n] = h(m) \cdot h(n)$

□

Remarque 5.1.1. *h est un* ***morphisme surjectif*** *(aussi appelé un quotient).*

Définition 5.12. Soit $L \subseteq \Sigma^*$ un langage et soit $\sim_L$ la congruence syntaxique sur Σ^*. Le monoïde quotient $\Sigma^*/\sim_L$ est appelé le monoïde syntaxique de L.

Proposition 5.13. *Avec les notations ci-dessus, L est reconnu par le morphisme* $\begin{array}{rcl} \varphi : \Sigma^* & \to & \Sigma^*/\sim_L \\ w & \mapsto & [w] \end{array}$.

Remarque 5.1.2. *Si $w \in L$ et $w' \sim_L w$ alors $w' \in L$, parce que $w' \sim_L w$, donc $\varepsilon w \varepsilon \in L \implies \varepsilon w' \varepsilon \implies w' \in L$.*

Démonstration. Soit $P \subseteq \Sigma^*/\sim_L$ donné par $P = \{[w] \mid w \in L\}$.

On doit montrer que $L = \varphi^{-1}(P)$, c'est-à-dire , $L = \{w \in \Sigma^* \mid \varphi(w) \in P\}$

En utilisant la remarque 5.1.2, on a que si $w \in L$ alors $[w] \subseteq L$.

On en conclut que

$$L = \bigcup_{w \in L} [w] \iff L = \varphi^{-1}(P)$$

□

Proposition 5.14. *Le monoïde syntaxique a une propriété similaire à 4.9 :* $\forall (M, \cdot, 1_M)$ *monoïde qui reconnait un langage* L*, on a le diagramme suivant :*

$$\begin{tikzcd} & (N, \cdot, 1_N) \arrow[dl, two heads] \arrow[dr, hook] & \\ (\Sigma^*/\sim_L, \cdot, [\varepsilon]) & & (M, \cdot, 1_M) \end{tikzcd}$$

Terminologie 5.1. *On dit que le monoïde syntaxique de* L *divise tout monoïde qui accepte* L*.*

Lemme 5.15. *Si* N *divise* M *et* N *reconnait un langage* L *alors* M *reconnait* L*.*

Démonstration.

$$\begin{tikzcd} & T \arrow[dl, two heads] \arrow[dr, hook] & \\ N & & M \end{tikzcd}$$

$\varphi : \Sigma^* \to N$ et $P \subseteq N$ tel que $L = \varphi^{-1}(P)$

Exercice : montrer que on peut construire $\varphi'' : \Sigma^* \to M$ qui reconnait L. □

Proposition 5.16. *Un monoïde* M *reconnait un langage* L *si et seulement si le monoïde syntaxique* $\Sigma^*/\sim_L$ *divise* M*.*

Démonstration.

— $\boxed{\Leftarrow}$
  $\Sigma^*/\sim_L$ reconnait L. Par le lemme, si $\Sigma^*/\sim_L$ divise M, alors M reconnait L.

— $\boxed{\Rightarrow}$
  Supposons que M reconnait L. Donc on a $\varphi : \Sigma^* \to M$ et $P \subseteq M$ avec $\varphi^{-1}(P) = L$.
  Soit $N = \varphi(\Sigma^*)$ l'image directe de Σ^* par φ.
  Alors $N \hookrightarrow M$ est un sous-monoïde de M.
  Je voudrais avoir un quotient $h : N \twoheadrightarrow \Sigma^*/\sim$.
  Si $n \in N$ alors $\exists w \in \Sigma^*$ tel que $\varphi(w) = n$. On définit $h(n) = [w]$.
  Il faut montrer que :

  1. h est bien défini (si $\varphi(w') = \varphi(w) \implies [w] = [w']$).
  2. h est un morphisme de monoïdes.
  3. h est surjectif.

  Montrons cela :

- 1 : Supposons que $\varphi(w) = \varphi(w')$.
  On veut montrer que $[w] = [w']$ c'est-à-dire $w \sim_L w'$, c'est-à-dire $\forall u, v \in \Sigma^*, uwv \in L \iff uw'v \in L$.
  Soient $u, v \in \Sigma^*$.

$$
\begin{aligned}
uwv \in L &\iff \varphi(uwv) \in P \\
&\iff \varphi(u)\varphi(w)\varphi(v) \in P \quad (\textit{car } \varphi \textit{ est un morphisme}) \\
&\iff \varphi(u)\varphi(w')\varphi(v) \in P \quad (\textit{car } \varphi(w) = \varphi(w')\ ) \\
&\iff \varphi(uw'v) \in P \\
&\iff uw'v \in L
\end{aligned}
$$

  Donc $w \sim_L w'$.

Exercice : finir la preuve.

□

6 Logique monadique du second ordre (MSO)

Est-ce qu'il existe $w \in \Sigma^*$ tel que $w \models \exists x \exists y (x < y)$ soit vraie ? (x, y sont interprétés comme des positions dans le mot w.) Par exemple, $a \not\models \exists x \exists y (x < y)$ Ici, cette proposition est satisfaite par $w \in \Sigma^*$ si et seulement si $|w| \geq 2$.

Pour une proposition φ, on a un langage $\mathcal{L}_\varphi = \{w \in \Sigma^* \mid w \models \varphi\}$.

Exemple 6.0.1. $\Sigma = \{a, b\}$ $\varphi_2 = \exists x \exists y (\forall z (z \geq x) \wedge Q_a x \wedge \forall z (z \leq y) \wedge Q_b y)$ où $\leq, \geq$ sont interprétés comme les relations habituelles sur $\mathbb{N}$. $Q_a x$ signifie qu'en position x, on retrouve la lettre a. $\mathcal{L}_{\varphi_2} = \left\{w \in \{a, b\}^* \mid w \models \varphi_2\right\} = a(a+b)^* b$

TODO: Comment and add line breaks

Exemple 6.0.2.

$$
\begin{aligned}
\varphi_3 &= \exists X (\forall x (\forall z (z \geq x) \rightarrow X(x)) \\
&\wedge \forall x (\forall z (z \leq x) \rightarrow \neg X(x)) \\
&\wedge \forall x \forall y (((x < y) \wedge \forall z (z > x) \rightarrow (z \geq y)) \rightarrow (X(x) \leftrightarrow \neg X(y))))
\end{aligned}
$$

$X \rightsquigarrow$ un ensemble de positions dans un mot
$X(x) \rightsquigarrow$ vrai si et seulement si $x \in X$
$\mathcal{L}_{\varphi_3} = \{w \in \Sigma^* \mid |w| \equiv 0 (\bmod 2)\}$

6.1 Syntaxe

Définition 6.1 (La *logique du premier ordre*).

— **V** un ensemble de variables : $\{x, y, z, x_1, y_1, z_1, \ldots\}$
— *Prédicats numériques* : $\mathcal{P} = \left\{R_i^j, i > 0, j \geq 0\right\}$

— Formules atomiques :

$$\begin{array}{rcll} \alpha & ::= & Q_a x & \text{Lettre } a \text{ en position } x \\ & | & R_i^j(x_1, \ldots, x_j) & \text{Prédicats numériques} \end{array}$$

— Formules du premier ordre :

$$\begin{array}{rcll} \varphi & ::= & \alpha & \textit{Formules atomiques} \\ & | & \varphi \wedge \varphi & \text{Conjonction} \\ & | & \neg\varphi & \text{Négation} \\ & | & \exists x \varphi & \textit{Quantificateur existentiel} \end{array}$$

Définition 6.2 (La *logique du second ordre*)**.** C'est une extension de la logique du premier ordre :

— un ensemble de *variables du second ordre* : $X, Y, Z, X_1, Y_1, Z_1 \ldots$

— Formules atomiques :

$$\begin{array}{rcll} \alpha & ::= & \cdots & \\ & | & X(x) & \text{Appartenance à } X \end{array}$$

— Formules du second ordre :

$$\begin{array}{rcll} \varphi & ::= & \cdots & \\ & | & \exists X \varphi & \textit{Quantificateur existentiel sur des ensembles} \end{array}$$

Notation 6.3 (*Quantificateur universel*)**.** On note $\forall x \varphi$ la formule $\neg \exists x (\neg \varphi)$. Cette notation s'étend aussi à la logique du second ordre.

6.2 Sémantique

6.2.1 Sémantique de la logique monadique du premier ordre

Pour $n \in \mathbb{N}$ la longueur d'un mot, x est interpreté comme un élément de $\{1, \ldots, n\}$.
$[\![R_i^j]\!]_n \subseteq \{1, \ldots, n\}^j$ est une *interpretation* de chaque prédicat R_i^j pour tout $n \geq 1$.

$$[\![<]\!]_n \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\} = \{(i, j) \mid i < j\}$$

Définition 6.4. Une $\mathcal{V}$-structure, pour $\mathcal{V} \subseteq \mathbf{V}$ est un mot de la forme $(a_1, U_1), \ldots, (a_n, U_n)$ où $a_1, \ldots, a_n \in \Sigma$ et $U_1, \ldots, U_n \subseteq \mathbf{V}$ tels que

1. $U_i \cap U_j = \emptyset, \forall i, j,\ i \neq j$
2. $\bigcup_{i=1}^n U_i = \mathcal{V}$

Définition 6.5 (relation de satisfaction)**.** On définit la relation de satisfaction $w \models \varphi$ où w est une $\mathcal{V}$-structure et φ une formule de premier ordre tel que

1. $\forall x \in \mathrm{FV}(\varphi) \implies x \in \mathcal{V}$ (FV = ensemble des variables libres)
2. Les quantificateurs distincts dans φ lient des variables distinctes

Si φ est un proposition, c'est-à-dire , $FV(\varphi) = \emptyset$ alors on a $\mathcal{L}_\varphi = \{w \text{ les } \emptyset\text{-structures} \mid w \models \varphi\} \subseteq \Sigma^*$.

Pour une interprétation $[\![R_i^j]\!]_{i,j}$ de prédicats numériques, on définit la relation $w \models \varphi$ par induction sur la structure de la formule φ.

- $w \models Q_a x$ ssi w contient une lettre (a, U_i) avec $x \in U_i$
- $w \models R_i^j(x_1, \ldots, x_j)$ ssi $[\![R_i]\!]_{|w|}(k_1, \ldots, k_j)$ est vrai où les $k1, \ldots, k_j$ sont les positions dans w où les variables $x_1, \ldots x_j$ apparaissent.
- $w \models \varphi_1 \wedge \varphi_2$ ssi $w \models \varphi_1$ et $w \models \varphi_2$
- $w \models \neg\varphi$ ssi $w \not\models \varphi$
- si $w = (a_1, U_1) \ldots (a_n, U_n)$ est une $\mathcal{V}$-structure. $w \models \exists\varphi$ ssi $\exists i \in \{1, \ldots, n\}$ $(a_1, U_1) \ldots (a_i, U_i \cup \{x\}) \ldots (a_n, U_n) \models \varphi$ (il s'agit d'une $\mathcal{V} \cup x$-structure.)

Exemple 6.2.1.

$$abc \models_? \exists x \exists y (x < y)$$

$V_1 = (a, \{x\})(b, \emptyset)(c, \emptyset)$ est une $\{x\}$-structure et on se demande si $V_1 \models_? \exists y (x \leq y)$.
On prend maintenant $V_2 = (a, \{x\})(b, \{y\})(c, \emptyset)$ et :

$$\begin{aligned} V_2 \models_? (x < y) \quad &\iff \quad (1, 2) \in [\![<]\!]_3 \\ &\iff \quad 1 < 2 \end{aligned}$$

Et donc $abc \models \exists x \exists y (x < y)$

Soit φ et ψ deux formules, $\varphi \iff \psi$ si et seulement si $\mathcal{L}_\varphi = \mathcal{L}_\psi$.

Abbréviation 6.1. $\forall x \varphi = \neg \exists x (\neg \varphi)$

Exemple 6.2.2.

$$\begin{aligned} abc \quad &\models \quad \exists x \exists y (\forall z (z \geq x) \wedge Q_a x \wedge \forall z (z \leq y) \wedge Q_b y) \\ (a, \{x\}), (c, \emptyset), (b, \emptyset) \quad &\models \quad \exists y (\forall z (z \geq x) \wedge Q_a x \wedge \forall z (z \leq y) \wedge Q_b y) \\ a, \{x\}), (c, \emptyset), (b, \{y\}) \quad &\models \quad (\forall z (z \geq x) \wedge Q_a x \wedge \forall z (z \leq y) \wedge Q_b y) \\ (a, \{x\}), (c, \emptyset), (b, \{y\}) \quad &\models \quad (\neg \exists z (z < x) \wedge Q_a x \wedge \neg \exists z (z > y) \wedge Q_b y) \end{aligned}$$

- $(a, \{x\}), (c, \emptyset), (b, \{y\}) \models \neg \exists z (z < x) \Leftarrow (a, \{x\}), (c, \emptyset), (b, \{y\}) \not\models \exists z (z < x)$
- $(a, \{x\}), (c, \emptyset), (b, \{y\}) \models Q_a x \Leftarrow (a, \{x\})$
- $(a, \{x\}), (c, \emptyset), (b, \{y\}) \models \neg \exists z (z > y) \Leftarrow (a, \{x\}), (c, \emptyset), (b, \{y\}) \not\models \exists z (z > y)$
- $(a, \{x\}), (c, \emptyset), (b, \{y\}) \models Q_b y \Leftarrow (b, \{y\})$

6.2.2 Sémantique de la logique monadique du second ordre

V_1 les variables du premier ordre
V_2 les variables du second ordre

Définition 6.6. Une $(\mathcal{V}_1, \mathcal{V}_2)$-structure est un mot sur $\Sigma \times 2^{\mathcal{V}_1} \times 2^{\mathcal{V}_2}$, c'est-à-dire un mot de la forme $(a_1, U_1, U_1), \ldots, (a_n, U_n, V_n)$ où

- Les U_i sont des ensembles de variables du premier ordre.
- Les V_i sont des ensembles de variables du second ordre.
- $(a_1, U_1), \ldots, (a_n, U_n)$ est une $\mathcal{V}_1$-structure.

Définition 6.7. On définit la relation de satisfaction $w \models \varphi$ où w est une $(\mathcal{V}_1, \mathcal{V}_2) - structure$ et φ une formule de MSO avec les mêmes contraintes sur les variables que pour la logique du premier ordre TODO: Add reference plus $w \models X(x)$ et $w \models \exists X \varphi$

L'induction est la même que pour le premier ordre, avec ces cas en plus :

- $w \models X(x)$ ssi w contient une lettre (a, S, T) où $x \in S$ et $X \in T$
- $w \models \exists X \varphi$ ssi $\exists J$ un ensemble de positions dans w avec la propriété : la $(\mathcal{V}_1, \mathcal{V}2)$-structure w' obtenue en remplaçant (a_i, S_i, T_i) pour $i \in J$ par $(a_i, S_i, T_i \cup \{X\})$ satisfait φ.

Exemple 6.2.3. TODO

Définition 6.8. Un langage L est définissable dans MSO[<] (la logique monadique du second ordre avec un prédicat binaire $<$) si et seulement si il existe une formule $\varphi \in MSO[<]$ telle que $L = \{w \in \Sigma^* \mid w \models \varphi\}$

Définition 6.9. Un langage $L \subseteq \Sigma^*$ est définissable dans $FO[<]$ (la logique du premier ordre avec $<$) si et seulement si $\exists \varphi \in FO[<]$ telle que $L = \{w \in \Sigma^* \mid w \models \varphi\}$

6.3 Relation avec les expressions rationnelles

Théorème 6.10. *Un langage est définissable dans $MSO[<]$ ssi L est régulier.*

Démonstration.

- $\boxed{\Leftarrow}$
  Soit $A = \langle Q, q_0, F, \delta \rangle$ un automate fini déterministe qui accepte un langage L. On peut supposer que $L \subseteq \Sigma^*$ (car $\varepsilon \in L$, on va prendre la disjonction de la formule obtenue pour $L \setminus \{\varepsilon\}$ avec $\forall x \neg(x = x)$)
  Soit $w \in \Sigma^*$, Alors w est reconnu par l'automate A ssi $\exists X_0, \ldots, X_{k-1} \subseteq \{1, \ldots, |w|\}$ tels que les propriétés suivantes soient vérifiés :
  1. $\bigcup_{i=0}^{k-1} X_i = \{1, \ldots, |w|\}$
  2. $\forall i < j, X_i \cap X_j = \emptyset$
  3. $1 \in X_0$
  4. $\forall j \in \{1, \ldots, |w|\}$ si $j \in X_i$ et $j + i \in X_e$ et si a est la lettre en position j dans le mot w, alors $\delta(q_i, a) = q_e$.
  5. Si $|w| \in X$ et a est la dernière lettre de w, alors $\delta(q_j, a) \in F$.

  Supposons que $w = a_1 a_2 \ldots a_n$ est accepté par A. On construit les ensembles $(X_i)_{i \in \{0, \ldots, k-1\}}$ tel que $i \in X_j$ ssi après avoir lu les premiers $i - 1$ lettres de w on arrive a l'état q_j TODO: Diagram ?

  Les ensembles $X_0, \ldots, X_{k-1}$ satisfait les 5 propriétés.

1. $X_0 \cup \ldots \cup X_{k-1} = \{i, \ldots, |w|\}$. L'inclusion a gauche est vraie par définition. Montrons l'autre inclusion. On considère l'unique chemin dans l'automate A obtenu en lisant les premieres $i-1$ lettres de w à partir de l'état q_0. Supposons qu'on arrive dans l'état q_j. Par définitions $u \in X_j$.
2. $\forall i < j, X_i \cap X_j = \emptyset$ est vrai car l'automate est déterministe.
3. $1 \in X_0$ car si on lit les premieres 0 lettres de w on reste a l'état q_0.
4. Après les premieres $j-1$ lettres on arrive a q_i. a est la lettre en position j. Donc après les premieres j lettres on arrive dans l'état $\delta(q_i, a)$. Si après les premières j lettres on arrive dans l'état q_l, alors $\delta(q_i, a) = q_l$.
5. w est accepté par A. Donc si $|w| \in X_j$ alors après avoir lu les premieres $|w|-1$ lettres on arrive dans q_j. Si a est la dernière lettre de w, alors $\delta(q_j, a)$ est un état acceptant car w est accepté.

On construit la formule

$$\varphi = \exists X_0 \ldots \exists X_{k-1}(\varphi_1 \wedge \ldots \wedge \varphi_5)$$

où

- $\varphi_1 = \forall x(X_0(x) \vee \ldots \vee X_{k-1}(x)) = \forall x \bigvee_{i=1}^{k-1} X_i(x)$
- $\varphi_2 = \forall x \bigwedge_{0 \leq i < j \leq k-1} \neg(X_i(x) \wedge X_j(x))$
- $\varphi_3 = \exists x \left((\forall y, x \leq y) \wedge X_0(x)\right)$
- $\varphi_4 = \forall x \left(\forall y(y = x+1) \to \bigwedge_{0 \leq i < l < k}((X_i(x) \wedge X_l(y)) \to \bigvee_{S_l} Q_a(x))\right)$
  Où $S_l = \{a \in \Sigma \mid \delta(q_i, a) = a_l\}$
- $\varphi_5 = \forall x \left(\forall y(x \geq y) \to \bigwedge_{i=0}^{k-1}(X_i(x) \to \bigvee_{T_i} Q_a(x))\right)$
  Où $T_i = \{a \in \Sigma \mid \delta(q_i, a) \in F\}$

— $\boxed{\Rightarrow}$
On suppose que L est défini par une formule de $MSO[<]$. La preuve est par induction sur la structure de la formule.
$L \subseteq (\Sigma \times 2^{\mathcal{V}_1} \times 2^{\mathcal{V}_2})^*$.
TODO: Rappel? $\mathcal{L}$ est l'ensemble de toutes les $(\mathcal{V}_1, \mathcal{V}_2)$-structures, pour $\mathcal{V}_1$ en ensemble de variables du premier ordre et $\mathcal{V}_2$ un ensemble de variables du second ordre.

Exercice 6.3.1. Trouver un automate sur l'alphabet $\Sigma \times 2^{\mathcal{V}_1} \times 2^{\mathcal{V}_2}$ qui accepte $\mathcal{L}$

Exercice 6.3.2. Trouver un automate qui accepte les $(\mathcal{V}_1, \mathcal{V}_2)$-structures tel qu'une variable de premier ordre x apparait dans une lettre de la forme (a, S, T). Le but est de montrer que le $\mathcal{L}\left(Q_a(x)\right)$ est régulier.

Exercice 6.3.3. Montrer que $\mathcal{L}_{X(x)}$ est régulier.

Exercice 6.3.4. Montrer que $\mathcal{L}_{x<y}$ est régulier.

Soit $\varphi = \exists x \psi$ et supposons que L_ψ est régulier et donc accepté par un automate $A = \langle Q, q_0, F, \delta \rangle$ sur l'alphabet $\Sigma \times 2^{\mathcal{V}_1} \times 2^{\mathcal{V}_2}$.
On définit l'automate $A' = \langle Q \times \{0,1\}, (q_0, 0), F \times \{1\}, \delta' \rangle$ où

$$\begin{aligned}
\delta' &= \{((q,u),(a,S,T),(q',u)) \mid u \in \{0,1\}, x \notin S, (q,(a,S,T),q') \in \delta\} \\
&\cup \{((q,0),(a,S \setminus \{x\},T),(q',1)) \mid x \in S, (q,(a,S,T),q') \in \delta\}
\end{aligned}$$

TODO: diagram ? ? ?

A' est un automate sur l'alphabet $\Sigma \times 2^{\mathcal{V}_1 \setminus \{x\}} \times 2^{\mathcal{V}_2}$.

$(a_1, S_1, T_1), \ldots, (a_n, S_n, T_n) \in (\Sigma \times 2^{\mathcal{V}_1 \setminus \{x\}} \times 2^{\mathcal{V}_2})^*$ est accepté par A' si et seulement si $\exists i \in \{1, \ldots, n\}$ tel que le mot $w^* = (a_1, S_1, T_1), \ldots, (a_i, S_i \cup \{X\}, T_i), \ldots, (a_n, S_n, T_n)$ est accepté par A.

□

Théorème 6.11. *Un langage est définissable dans FO$[<]$ ssi il existe une expression rationnelle sans étoile r définissant un langage ssi le monoïde syntaxique du langage est apériodique (c'est-à-dire il ne contient aucun sous-groupe non-trivial).*

6.3.1 Éléments idempotents

Définition 6.12 (Éléments idempotents)**.** Soit $(M, \cdot, 1)$ un monoïde. Un élément $e \in M$ est *idepmpotent* si et seulement si $e \cdot e = e$.

Exercice 6.3.5. Soit $e \in M$ est idempotent, alors $\forall i \in \mathbb{N}^*.\ e^i = e$.

Lemme 6.13. *Soit $(M, \cdot, 1_M)$ un monoïde fini et soit $m \in M$. Si $|M| = n$ alors $m^{n!}$ est idempotent.*

Démonstration. Parmi les puissances de m : $m, \ldots, m^{n+1}$ il y deux qui sont égales, car $|M| = n$. Soient $k \in \{1, \ldots, n\}$ et $l \geq 1$ tel que $m^k = m^{k+l}$ $(k + l \in \{1, \ldots, n+1\})$. $m^k = m^{k+l} \implies m^{k+2l} = m^{k+l}m^l = m^k m^l = m^{k+l} = m^k$

Par induction on a que $\forall i \geq m^{k+il} = m^k \implies \forall i \geq 1, j \geq 1, m^{k+il+j} = m^{k+j}$

Soit $j = kl - k$ et $i = k$, alors on obtient

$$\begin{aligned} m^{k+kl-k} = m^{kl} &\implies m^{2kl} = m^{kl} \\ &\iff m^{kl}m^{kl} = m^{kl} \\ &\iff m^{kl} \text{es un idempotent} \end{aligned}$$

Donc $\exists x \in \mathbb{N}$, $m^{n!} = m^{klx} = (m^{kl})^x = m^{kl} \implies m^{n!}$ est idempotent. □

Définition 6.14 (Puissance idempotente)**.** Une *puissance idempotente* d'un élément $m \in M$ est un élément de la forme m^i $(i \geq 1)$ tel que m^i est idempotent.

Lemme 6.15. *Une puissance idempotente de $m \in M$ est unique, dans le sens suivant : si m^i et m^j sont des idempotents, alors $m^i = m^j$.*

Démonstration. m^i idempotent $\implies (m^i)^j = m^i$, voire 6.3.5.

Pareil pour m^j idempotent $\implies (m^j)^i = m^j$. Donc $m^i = m^{ij} = m^{ji} = m^j$ □

6.3.2 Relations de Green

Définition 6.16. Soit $(M, \cdot, 1_M)$ un monoïde. On défini les relations suivantes.

- *prefixe, ou $>_R$, ou* R-préordre : On dit que m est un préfixe de n si et seulement si $\exists x \in M,\ n = mx$. On le note $m >_R n$.

  Un ensemble de la forme $y \cdot M = \{y \cdot z \mid z \in M\}$ pour $y \in M$ est appelé un idéal à droite de M.

  m est un préfixe de n ssi $m \cdot M \supseteq n \cdot M$ (à vérifier en exercice).

— *suffixe, ou* $>_L$ *, ou* L-préordre : On dit que m est un suffixe de n ssi $\exists y \in M,\ n = ym$. On le note $m >_L n$.
TODO: exercice (same as before)

— *infixe, ou* $>_J$ *, ou* J-préordre : On dit que m est un infixe de n ssi $\exists x, y \in M,\ n = xmy$. On le note $m >_J n$.
TODO: exercice (same as before)

Exercice 6.3.6. Montrer que $>_R, >_J, >_L$ sont transitives et réflexives et donc des préordres.

Notation 6.17. On défini les relations d'équivalence associées à ces préordres L

— mRn ssi $m >_R n$ et $n >_R m$ (On dit m est *R-équivalent* n)
On note par $[m]_R$ la R-classe d'équivalence de m, *i.e.* , $[m]_R = \{n \in M \mid mRn\}$

— mLn ssi $m >_L n$ et $n >_L m$ (On dit m est *L-équivalent* n)
On note par $[m]_L$ la L-classe d'équivalence de m, *i.e.* , $[m]_L = \{n \in M \mid mLn\}$

— mJn ssi $m >_J n$ et $n >_J m$ (On dit m est *J-équivalent* n)
On note par $[m]_J$ la J-classe d'équivalence de m, *i.e.* , $[m]_J = \{n \in M \mid mJn\}$

Exercice 6.3.7. $\forall m \in M, [m]_R \subseteq [m]_J$

Démonstration. Soit $m \in M$ et soit $x \in [m]_R$, alors xRm et donc $x >_R m$ et donc $\exists y$ tel que $m = xy$ et donc $m = 1xy$ et ainsi $x >_J m$ et donc $x \in [m]_J$.1 ☐

Lemme 6.18 (Eggbox lemma)**.** *Soit M un monoïde fini et $m, n \in M$,*

1. *Si $m >_R n$ et mJn alors $n >_R m$ et donc mRn*
2. *Si $m >_L n$ et mJn alors $n >_L m$ et donc mLn*

Démonstration.

1. Soit $m, n \in M$ tels que $m >_R n$ et mJn alors

$$m >_R n \implies \exists x \in M,\ n = mx$$

$$mJn \implies \exists y, z \in M,\ m = ynz$$

TODO: make proper Tout cela implique $m = ymxz$

$$\begin{aligned} m = ymxz \quad &\implies \quad m = y(ymxz)xz \\ &\implies \quad m = y^2m(xz)^2 \end{aligned}$$

Par induction on peut montrer que

$$\forall i \geq 1, m = y^i m(xz)^i \tag{2}$$

Soit $i \geq 1$ tel que $(xz)^i$ est idempotent, alors

$$\begin{aligned} m &= y^i m(xz)^i \quad (2) \\ &= y^i m(xz)^i(xz)^i \quad \big((xz)^i \text{ idempotent}\big) \\ &= m(xz)^i \quad (2) \\ &= mx(xz)^{i-1}z \quad (\text{pour } i > 0) \\ &= n(xz)^{i-1}z \end{aligned}$$

Donc $m = n(xz)^{i-1}z \implies n$ est un préfixe de $m \implies n >_R m$.
On a alors que mRn.

2. Exercice

□

Définition 6.19. On dit mHn ssi mRn et mLn. On a $H = R \cap L$.

Exercice 6.3.8. L'intersection d'une R-classe avec une L-classe d'équivalence est soit vide soit une H-classe d'équivalence.

Démonstration. Si $[m]_R \cap [n]_L \neq \emptyset$, alors l'énoncé est vrai.
Soit $s \in [m]_R \cap [n]_L$. On va montrer que $[m]_R \cap [n]_L = [s]_H$

- Soit $s' \in [m]_R \cap [n]_L$. On sait $s'Rs$ car $s' \in [m]_R$ donc $s'RmRs$. Pareil, $s'Ls$ car $s'LnLs$. Et donc $s'Hs$. Et donc $[m]_R \cap [n]_L \subseteq [s]_H$
-

$$\begin{aligned} s' \in [s]_H &\implies sRs' \text{ et } sLs' \\ &\implies s'Rm \text{ et } s'Ln \\ &\implies s' \in [m]_R \cap [n]_L \end{aligned}$$

□

Lemme 6.20. *Soit $\mathbb{H}$ une H-classe incluse dans une J-classe $\mathbb{J}$. Alors, soit*

1. $\forall m, n \in \mathbb{H}, mn \notin \mathbb{J}$
2. *$\mathbb{H}$ est un groupe (Attention!! L'élément neutre de $\mathbb{H}$ n'est pas forcément 1_M)*

Démonstration. Supposons que $\exists m_0, n_0 \in \mathbb{H}$ tels que $m_0, n_0 \in \mathbb{J}$. On suppose donc la négation de 1. Montrons que $\mathbb{H}$ est un groupe :

- Étape 1 : Soit $m, n \in \mathbb{H}$, montrons que $mn \in \mathbb{J}$.
- Étape 2 : Soit $m, n \in \mathbb{H}$, montrons que $mn \in \mathbb{H}$.

  Exercice 6.3.9. Montrer que $mnLn$.

  Et donc $mn \in \mathbb{H}$
- Étape 3 : Soit $m \in \mathbb{H}$ on va montrer que $\mathbb{H}$ a un élément neutre et que ses éléments ont une inverse. Par 6.15, il existe $i > 1$ tel que m^i est idempotent. Par l'étape 2, on sait que $m^i \in \mathbb{H}$, donc on peut conclure que $\mathbb{H}$ contient un élément idempotent, notons le par e. On va montrer que e est l'élément neutre de $\mathbb{H}$, c'est-à-dire , $\forall m \in \mathbb{H},\ m \cdot eme \cdot m = m$.
  Soit $n \in \mathbb{H}$, donc $nLe \implies \exists x,\ n = x \cdot e \implies ne = (xe)e = x(ee) = xe = n$
  De la même manière, $n \in \mathbb{H} \implies nRe \implies \exists y,\ n = ey \implies en = eey = ey = n$ Donc $en = n$.
  Il reste à montrer que chaque élément possède un inverse : $\forall n \in \mathbb{H}, \exists n' \in \mathbb{H}$ tel que $nn' = e = n'n$
  On observe que $\mathbb{H}$ contient un unique idempotent. Soit e' un autre idempotent, par le même argument que ci-dessus, e' est un élément neutre donc : TODO. $e ='e$
  Soit $n \in \mathbb{H}$. Par le lemme 6.15, il existe $j \geq 1$ tel que n^j est idempotent, mais $n^j \in \mathbb{H}$ donc $n^j = e$. Et ainsi, n^{j-1} est l'inverse de n.

□

Définition 6.21. Un monoïde $(M, \cdot, 1_M)$ est apériodique ssi $\forall m \in M, \exists i \geq 1,$ tel que $m^i = m^{i+1}$

Théorème 6.22 (Schützenberger)**.** *Soit $L \subseteq A^*$ un langage régulier. Les affirmations suivantes sont équivalentes :*

1. *L est définissable par une expression rationnelle sans-étoile.*
2. *L est définissable dans la logique $FO[<]$.*
3. *Le monoïde syntaxique de L est apériodique.*

TODO

7 Apprentissage de langages rationnels par queries et contrexemples

Nous allons étudier un algorithme qui permet de "deviner" un automate reconnaissant un langage rationnel inconnu L, en suivant un algorithme à complexité polynomiale "raisonnable". L'idée est que le *learner* (ou étudiant), une entité dont l'objectif est de retrouver le langage, pose des questions à un *oracle* (ou teacher) qui connait le langage cherché.

Les questions sont de deux types :

— Les *membership queries* : est-ce qu'un mot w donné appartient au langage L ?

— Les *conjectures* : on présente un automate conjecture. Si cet automate reconnaît L, alors l'oracle répond "Ok". Sinon, il retourne un mot *contre-exemple* w. Ce mot vérifie soit $w \in L$ (alors qu'il n'est pas accepté par notre automate), soit $w \notin L$ (alors qu'il est accepté par notre automate).

Cet algorithme découle du travail de [Ang87] et est connu comme l'algorithme L^*.

7.1 Définitions

Définition 7.1 (Clôture par préfixes / (suffixes))**.** Un ensemble S est *clos par préfixes* (suffixes), si

$$\forall w \in S, \text{ tout préfixe (suffixe) de } w \text{ est dans } S.$$

Exercice 7.1.1. Indiquez lesquels des ensembles suivants sont clos par préfixes :

1. $\{0, 2, 10, 010\}$ $\times$
2. $\{110, 1, 0, \varepsilon, 11\}$ ✓
3. $\{1110, 10, 1\}$ $\times$
4. $\{011, 0, \varepsilon, 11, 01\}$ $\times$
5. $\{111, \varepsilon, 11, 1, 0\}$ ✓

Pendant le processus, le learner maintient une *table*, qui est un triplet (S, E, T) où S est un ensemble clos par préfixes, E est un ensemble clos par suffixes, et T une fonction $(S \cup S\Sigma) \times E \to \{0, 1\}$. Une table peut être représentée par une matrice où :

— les colonnes sont indexées par l'ensemble E de mots clos par suffixes ;

— les lignes sont indexées par l'ensemble de mots $S \cup S \cdot \Sigma$;

— la case de la ligne s et de la colonne e contient $T(s,e)$.

Définition 7.2. Si $s \in S \cup S\Sigma$, soit $n = |E|$, on définit

$$\begin{array}{rccc} \textit{row}: & S \cup S\Sigma & \to & [0,1]^n \\ & s & \mapsto & (T(s,e_1),\ldots,T(s,e_n)) \end{array}$$

Définition 7.3. Une table est close si

$$\forall t = sa \in S\Sigma, \exists s' \in S, \mathrm{row}\,(t) = \mathrm{row}\,(s')$$

Définition 7.4. Une table est cohérente si

$$\forall s_1, s_2 \in S, \mathrm{row}\,(s_1) = \mathrm{row}\,(s_2) \implies \forall a \in \Sigma, \mathrm{row}\,(s_1 a) = \mathrm{row}\,(s_2 a)$$

Exemple 7.1.1.

— La table suivante n'est pas close car $\nexists s \in S, \mathrm{row}\,(ab) = \mathrm{row}\,(s)$

		E	
		ε	a
S	ε	0	1
	a	1	1
	b	0	0
$S\Sigma$	aa	0	0
	ab	1	0
	bb	0	1

— La table suivante est close et cohérente

	ε	a
ε	0	0
a	1	1
b	1	1
aa	0	0
ab	0	0

— La table suivante n'est pas cohérente car $\mathrm{row}\,(\varepsilon) = \mathrm{row}\,(a)$ mais $\mathrm{row}\,(\varepsilon a) = \mathrm{row}\,(a) \neq \mathrm{row}\,(aa)$

	ε	a
ε	0	1
a	0	1
b	0	0
aa	0	0
ab	0	1
bb	1	1
bb	1	0

7.2 L'algorithme

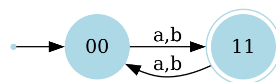
Construction 7.1. *À partir de chaque table (S, E, T) close et cohérente, on peut associer un automate déterministe $A = (Q, q_0, F, \delta)$ où :*

$$\begin{aligned} Q &= \{row(s) \mid s \in S\} \\ q_0 &= \{row(\varepsilon)\} \\ F &= \{row(s) \mid T(s) = 1\} \\ \delta(row(s), a) &= row(sa). \end{aligned}$$

On note cet automate $A(S, E, T)$.

Exemple 7.2.1. L'automate associé à la table close et cohérente de l'exemple 7.1.1 est :

	ε	a
ε	0	0
a	1	1
b	1	1
aa	0	0
ab	0	0



L'algorithme suivant permet de retrouver le langage cherché.

Algorithme 7.1. *Initialement, on peut conjecturer l'un des automates des Figures 1 et 2, selon si $\varepsilon \in L$ ou non.*

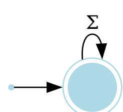


FIGURE 1 – Automate pour $\varepsilon \in L$

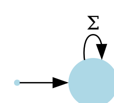


FIGURE 2 – Automate pour $\varepsilon \notin L$

Tant que l'oracle donne un contre-exemple

— *w = contre-exemple.*

— *w et tous ses préfixes sont ajoutés à S.*

— *Les nouvelles lignes sont remplies en posant des membership queries.*

— *Tant que T n'est pas cohérente ou n'est pas close :*

- *Si T n'est pas cohérente (i.e. , $\exists s_1, s_2 \in S, \exists a \in \Sigma, row(s_1) = row(s_2)$ et $row(s_1 a) \neq row(s_2 a)$ i.e. $\exists e \in E, T(s_1 ae) \neq T(s_2 ae)$)*
  alors on ajoute ae et tous ses suffixes à E.
- *Si T n'est pas close, alors $\exists t \in S\Sigma, \forall s \in S, row(t) \neq row(s)$ et on ajoute t à S.*

— *On propose à l'oracle l'automate conjecture associé à T.*

7.3 Propriétés des tables closes et cohérentes

Notation 7.5. On note (S, E, T) une table, où S est l'ensemble des lignes. E l'ensemble des mots des colonnes et T la fonction de vérité. Parfois, par abus de notation, on dira que la table est T.

Définition 7.6. Un automate déterministe (Q, q_0, F, S) est *en accord* avec une table (S, E, T) si

$$\forall s \in S \cup S\Sigma, \forall e \in E, \delta(q_0, se) \in F \iff T(se) = 1$$

Théorème 7.7. *Si $A(S, E, T)$ est l'automate issu d'une table (S, E, T) alors A est en accord avec T et tout autre automate en accord avec T et non isomorphe à A possède au moins un état de plus que A.*

La preuve de ce théorème est un corollaire des trois lemmes suivants.

Lemme 7.8. *Si (S, E, T) est close et cohérente alors dans l'automate $A = \mathcal{A}(S, E, T)$*

$$\forall s \in S \cup S\Sigma, \delta(q_0, s) = row(s)$$

Démonstration. Induction sur la taille de s.

— Si $|s| = 0$, alors $s = \varepsilon$ et donc $\delta(q_0, \varepsilon) = \mathrm{row}(\varepsilon)$ est vrai par construction de l'automate.

— Supposons l'énoncé vrai pour tout mot appartenant à $S \cup S\Sigma$ de longueur inférieure ou égale à n et soit t un mot de $S \cup S\Sigma$ de longueur $n + 1$, alors $t = s \cdot x$, avec $s \in S$, car si $t \in S\Sigma$, $s \in S$ trivialement et si $t \in S$, comme S est clos par préfixes, on a $s \in S$.

$$\begin{aligned}
\delta(q_0, t) &= \delta(\delta(q_0, s), x) \\
&= \delta(\mathrm{row}(s), x) \quad (\textit{Par hypothèse d'induction}) \\
&= \mathrm{row}(sx) \quad (\textit{Par la définition de } \delta \textit{ dans 7.1}) \\
&= \mathrm{row}(t)
\end{aligned}$$

□

Lemme 7.9. *Si la table (S, E, T) est close et cohérente, alors l'automate $A = \mathcal{A}(S, E, T)$ est en accord avec la fonction T, i.e.*

$$\forall s \in S \cup S\Sigma, \forall e \in E, \delta(q_0, se) \in F \iff T(se) = 1$$

Démonstration. Induction sur la taille de e.

— Si $|e| = 0$, alors $e = \varepsilon$. Soit $s \in S \cup S\Sigma$, et donc $se = s$.

- Si $s \in S$, alors par le lemme précédent (7.8), $\delta(q_0, s) = \mathrm{row}(s)$, et comme $\mathrm{row}(s) \in F \iff T(s) = 1$ par construction de l'automate, on retrouve le résultat cherché.
- Si $s \in S\Sigma$, comme la table est close, il existe $s' \in S$, tel que $\mathrm{row}(s') = \mathrm{row}(s)$ qui vérifie $\mathrm{row}(s') \in F \iff T(s') = 1$ par construction de l'automate.

— Supposons l'énoncé vrai pour tout mot de E de longueur inférieure ou égale à n. Soit $e \in E$ tel que $|e| = n+1$, et on pose $e = xe'$, avec $x \in \Sigma$. Soit $s \in S \cup S\Sigma$, il existe $s' \in S$ tel que $\mathrm{row}(s) = \mathrm{row}(s')$ car la table est close.

$$\begin{aligned}
\delta(q_0, se) &= \delta(\delta(q_0, s), e) \\
&= \delta(\delta(q_0, s), xe') \\
&= \delta(\mathrm{row}(s), xe') \quad \textit{(par le lemme 7.8)} \\
&= \delta(\mathrm{row}(s'), xe') \quad (\textit{car } row(s) = row(s')) \\
&= \delta(\delta(\mathrm{row}(s'), x), e') \\
&= \delta(\mathrm{row}(s'x), e') \quad (\textit{par définition de } \delta) \\
&= \delta(\delta(q_0, s'x), e') \quad \textit{(par le lemme 7.8)} \\
&= \delta(q_0, s'xe')
\end{aligned}$$

Comme $|e'| = n$ on peut appliquer l'hypothèse d'induction et on a :

$$\delta(q_0, se) = \delta(q_0, s'xe') \in F \iff T(s'xe') = 1$$

Puisque $\mathrm{row}(s) = \mathrm{row}(s')$ et $e = xe'$, on a que $T(s'xe') = T(sxe') = T(se)$. □

Lemme 7.10. *Soit (S, E, T) une table close et cohérente, alors si l'automate $A = \mathcal{A}(S, E, T)$ a n états, tout automate A' en accord avec T et ayant au plus n états est isomorphe à A.*

Démonstration. Soit $A' = (Q', q_0', F', \delta')$ un autre automate en accord avec T et ayant au plus n états.

L'objectif est d'exhiber un isomorphisme entre les deux automates. La construction et la vérification de cet isomorphisme suivent les arguments détaillés dans la preuve du Lemme 4 de [Ang87]. □

7.4 Terminaison de l'algorithme

Lemme 7.11. *Si (S, E, T) est une table, alors tout automate en accord avec T possède au moins autant d'états que le nombre de valeurs distinctes de l'ensemble $R = \{row(s) \mid s \in S\}$.*

Démonstration. Soit $A = (Q, q_0, F, \delta)$ un automate en accord avec T et définissons $f : R \to Q$, $f(\mathrm{row}(s)) = \delta(q_0, s)$.

Soient s_1 et s_2 tel que $\mathrm{row}(s_1) \neq \mathrm{row}(s_2)$, donc $\exists e \in E, T(s_1e) \neq T(s_2e)$. Comme A est en accord avec T, seulement l'un de deux mots est reconnu.

Sans perte de généralité, on suppose s_1e reconnu et s_2e pas reconnu, donc $\delta(q_0, s_1e) \in F$, alors que $\delta(q_0, s_2e) \notin F$. Cela implique que $\delta(q_0, s_1) \neq \delta(q_0, s_2)$.

Pour deux rows distincts correspondent deux états distincts de A, et donc A a au moins autant d'états que $|R|$. □

Soit n le nombre d'états de l'automate minimal reconnaissant le langage inconnu L cherché.

Remarque 7.4.1. *Le nombre d'états des différents automates conjectures présentés à l'oracle ne peut qu'augmenter.*

— *si on ajoute un mot à E parce que la table n'était pas cohérente, le nombre de valeurs rows distincts augmente d'une unité.*

— *si on ajoute un mot à S parce que la table n'était pas close, le nombre de valeurs de rows distincts augmente d'une unité.*

On déduit qu'on peut rencontrer une table non close ou non cohérente au plus $n-1$ fois tout au long de l'algorithme.

En particulier, on déduit qu'après un nombre fini d'étapes de l'algorithme, on arrive toujours à trouver une table close et cohérente et à émettre une conjecture.

Combien de conjectures le learner émet-il avant de donner la bonne ?

Soit (S, E, T) une table close et cohérente et $A(S, E, T)$ l'automate associé. Supposons que A soit une conjecture fausse, et donc que l'oracle fournit un contre-exemple t.

L'automate $A(S, E, T)$ et l'automate du langage cherché sont en désaccord sur le mot t, donc ces deux automates ne sont pas équivalents, et donc $A(S, E, T)$ a au moins un état de moins que l'automate cherché. Donc $A(S, E, T)$ a au plus $n-1$ états. On en déduit que le nombre de fausses conjectures émises pas le learner est au plus $n-1$.

On en déduit que l'algorithme s'arrête toujours, au pire après avoir rendu $n-1$ tables closes et cohérentes et après avoir émis au plus $n-1$ fausses conjectures.

7.5 Analyse de la complexité

Elle dépend du nombre n et de la longueur du plus long contre-exemple fourni par l'oracle, notée m.

— A chaque fois qu'on trouve une table non cohérente, on ajoute un mot à E.

— A chaque fois qu'on trouve une table non close, on ajoute un mot à S.

— A chaque fois qu'on émet une conjecture fausse, on ajoute au plus m mots à S (le contre-exemple et ses préfixes).

On en déduit que $|E| \leq n$.

La longueur maximale des mots de E augmente au plus d'une unité à chaque fois qu'on rend la table cohérente (on ajoute à E un mot xe avec $e \in E$). On déduit que la longueur maximale d'un mot de E est aussi $< n$.

De même, on a que $|S| \leq n + m(n-1)$. TODO: explanation of each number

La longueur maximale des mots de S est inférieure à $m - n - 1$.

On déduit que le cardinal maximal de $(S \cup S\Sigma) \times E$ (le nombre maximal de classes de la table) est : $(k+1)(n + m(n-1))n \in O(mn^2)$.

Considérons le coût de chaque type d'opérations :

— Vérifier si une table est close et cohérente se fait en temps polynomial relativement à la taille de la table.

— Ajouter un mot à S ou à E nécessite au plus nm membership queries.

— Construire une conjecture à partir d'une table cohérente et close est faisable en temps polynomial relativement à la taille de la table.

— Un contre-exemple nécessite l'addition d'au plus m mots à S et cette opérations est effectuée au plus $n-1$ fois.

On en déduit le théorème suivant :

Théorème 7.12. *L'algorithme produit toujours un automate isomorphe à l'automate minimal du langage cherché. De plus, le temps de calcul peut être exprimé par un polynôme en n et en m.*

Remarque 7.5.1. *Si l'oracle fournit toujours un contre-exemple de taille $\leq n$ (ce qui est toujours possible), alors le polynôme en question dépend de n seulement.*

8 String matching

Définition 8.1. Étant donné deux chaines de caractères, T le texte et M le motif, on dit que T présente une occurrence de M si

$$\exists i, 0 \leq i \leq |T| - |M|, \forall j \in \{1, \ldots, |M| - 1\}, T[i+j] = M[j]$$

Remarque 8.0.1. *Le texte T contient le facteur u si et seulement si T a un préfixe qui appartient à $\Sigma^* u$.*

Remarque 8.0.2. *L'algorithme trivial qui teste toutes les positions de T et vérifie s'il y a une occurrence de u en position i est en $O(|T|\,|u|)$.*

```
function NAIF(T, u)
    for i in 0, ..., |T| − |u| + 1 do
        j = 0
        while j < |u| et (u[j] == t[i + j]) do
            j ++
        if j == |u| then
            Afficher l'occurrence de u en position i
```

Remarque 8.0.3. *(Un premier algorithme meilleur que le naïf) Pour chercher les occurrences de u dans T, on peut construire un automate qui reconnait $\Sigma^* u$ et lui faire analyser T. À chaque fois qu'on passe par un état final, on vient de voir une occurrence de M.*

Remarque 8.0.4. *Avec l'algorithme naïf, chaque caractère de T pourrait être analysé $|u|$ fois, car le motif est décalé d'une position à chaque fois. Avec les automates, chaque caractère est analysé une seule fois, ce qui explique l'amélioration.*

Remarque 8.0.5. *L'approche par automates présente cependant quelques problèmes d'efficacité :*

- *Si l'automate est non déterministe, le temps de calcul peut devenir très lourd : il faut analyser un nombre potentiellement exponentiel de chemins.*
- *Si on le déterminise avant, alors on risque de trouver un automate de taille exponentielle.*

8.1 Knuth-Morris-Pratt

L'algorithme Knuth-Morris-Pratt, [KMP77], améliore l'algorithme naïf en introduisant des décalages d'amplitude > 1 et fait en sorte que chaque caractère de T soit analysé une seule fois.

Définition 8.2. Si w est un mot, on appelle *bord* de w le mot le plus long qui est en même temps préfixe et suffixe propre de w .

Exemple 8.1.1. "A" et "ABA" sont les bords de "ABABA".

Pour tout mot $w \in \Sigma^*$, on note Bord(w) le plus long bord de w, c'est-à-dire le plus long préfixe propre de w qui est aussi un suffixe propre de w. On note également $b(w)$ la longueur de Bord(w). De plus, pour tout j tel que $0 \leq j \leq |w|$, on note w_j le préfixe de w de longueur j.

La fonction KMP repose sur le calcul préalable de Bord(w_j) et $b(w_j)$ pour tous les j, dans un temps polynomial en $|w|$. Cette phase de prétraitement permet d'optimiser les décalages à effectuer lors de la recherche d'un motif u dans un texte T.

Supposons qu'au cours de la recherche de u dans T, un échec se produit au niveau de la position j du motif u alors que l'on examine le texte à partir de la position i. Cet échec indique que le caractère $T[i+j]$ ne correspond pas à $u[j]$.

Pour déterminer le prochain décalage efficace de u, on observe que :

— Décaler u d'une amplitude k, $1 \leq k \leq j$, n'a de sens que si, après ce décalage, un préfixe de u peut s'aligner avec la portion correspondante de T.

— Il est inutile de considérer des décalages $k < b(u_{j-1})$, car Bord(u_{j-1}) est précisément le plus long suffixe de u_{j-1} qui est aussi un préfixe de u.

Ainsi, après un échec à la position j, le décalage optimal consiste à aligner Bord(u_{j-1}) au début de u avec sa précédente occurrence dans T. Autrement dit, on recule le curseur j à $b(u_{j-1})$ sans avoir à comparer à nouveau les $b(u_{j-1})$ premiers caractères, car ils sont garantis égaux en raison de la définition du bord.

Ce mécanisme permet de limiter les comparaisons inutiles et assure que chaque caractère de T est comparé à un nombre constant de caractères de u en moyenne, rendant ainsi le nombre total de comparaisons linéaire en $|T| + |u|$.

Exemple 8.1.2. TODO: explain

$$\begin{aligned} T &= ABABAABCBABABACAB \\ u &= ABABACA \end{aligned}$$

L'algorithme effectue un pré-traitement dans lequel pour tout j avec $0 \leq j \leq |u| - 1$ calcule le plus grand bord du préfixe de u de longueur j.

Exemple 8.1.3. Pour le motif "ABABACA" :

— Bord$(\varepsilon) = \varepsilon$

— Bord$(A) = \varepsilon$

— Bord$(AB) = \varepsilon$

— Bord$(ABA) = A$

— Bord$(ABAB) = AB$

— Bord$(ABABA) = ABA$

— Bord$(ABABAC) = \varepsilon$

— Bord$(ABABACA) = A$

Il suffit de calculer une fonction qui à tout $j \in \{0, \ldots, |u| - 1\}$ associe la longueur du plus long bord du préfixe de longueur j du motif, c'est la fonction dite préfixe.

Cet algorithme peut aussi se retrouver comme un algorithme issu des automates.

Soit u le motif à chercher.

Définition 8.3. Pour tout $w \in \Sigma^*$ on défini $q(w) = \{$ le plus long $X \mid X$ est suffixe de w et X est préfixe de $u\}$

Proposition 8.4. *Soit $\cong$ la congruence de Nérode induite par le langage $\Sigma^* u$, alors*

$$w_1 \cong w_2 \iff q(w_1) = q(w_2)$$

Démonstration.

$\boxed{\Rightarrow}$
Supposons $w_1 \cong w_2$ $(\forall y \in \Sigma^*, w_1 y \in \Sigma^* u \iff w_2 y \in \Sigma^* u)$

Notons z le plus long suffixe de w_1 qui est aussi préfixe de u et notons y le mot $z^{-1}u$ (le mot obtenu de u en effaçant z au début) alors $w_1 y$ se termine par u, c'est-à-dire est dans $\Sigma^* u$.

Le mot y est le mot de longueur minimale tel que $w_1 y$ se termine par u, car s'il existait un mot y' plus court que y tel que $w_1 y'$ se termine par u alors z ne serait pas le plus long suffixe de w_1 qui est aussi préfixe de u.

Puisque $w_1 \cong w_2$ on a que $w_2 y$ est dans $\Sigma^* u$ aussi, c'est-à-dire se termine par u, de plus y est aussi le mot de longueur minimale tel que $w_2 y$ est dans $\Sigma^* u$. En effet si un mot y'' tel que $w_2 y''$ est dans $\Sigma^* u$ existait alors $w_1 y''$ serait aussi dans $\Sigma^* u$ à cause de l'équivalence, en contradiction avec la minimalité de y.

Il en découle que z est aussi un suffixe de w_2 et que c'est le plus long suffixe de w_2 qui aussi préfixe de u, donc $q(w_1) = q(w_2)$.

$\boxed{\Leftarrow}$
Supposons que $q(w_1) = q(w_2)$ et nous voulons montrer que

$$\forall y, \quad w_1 y \in \Sigma^* u \iff w_2 y \in \Sigma^* u$$

Ceci est trivialement vrai si $|y| \geq |u|$, dans ce cas

$$\begin{array}{rcl} w_1 y \text{ se termine par } u & \text{ssi} & y \text{ se termine par } u \\ & \text{ssi} & w_2 y \text{ se termine par } u \end{array}$$

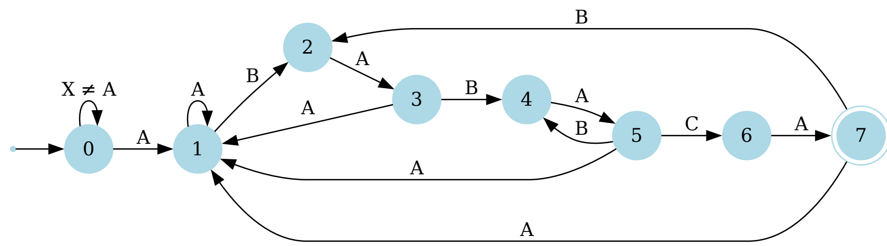
Supposons donc que $|y| < |u|$ et considérons le mot $w_1 y$

$w_1 y \in \Sigma^* u \implies \exists$ un suffixe z de w_1 qui est préfixe de u et z est un suffixe de $q(w_1) = q(w_2)$ donc z est aussi un suffixe de $q(w_2)$ et en particulier il est un suffixe de $w_2 \implies w_2 y$ se décompose en $w_2' z y = w_2 y \in \Sigma^* u$ $\square$

Cette proposition suggère une nouvelle manière pour calculer l'automate minimal pour $\Sigma^* u$

$$\begin{array}{rcl} Q & = & \{\text{les préfixes de } u\} \\ q_0 & = & <\varepsilon> \\ F & = & \{<u>\} \\ \delta(<v>, a) & = & \left\{ \begin{array}{ll} <v \cdot a> & \text{si } v \cdot a \text{ est un préfixe de } u \\ q(v \cdot a) & \text{sinon} \end{array} \right. \end{array}$$

Exemple 8.1.4. Pour le mot "ABABACA" on construit l'automate suivant :



L'automate est complet et tout transition "manquante" va vers l'état 0.

Remarque 8.1.1. *Si dans cet automate on remplace les préfixes de u par leur longueur, on se rend compte que cet automate contient la même information que la fonction préfixe.*

9 Évaluation de fonctions booléennes

Notation 9.1. On note $\mathbb{B} = \{0, 1\}$.

On veut évaluer les fonctions $f : \mathbb{B}^n \to \mathbb{B}$

Exemple 9.0.1. Une telle fonction est $f(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee x_3 : \mathbb{B}^3 \to \mathbb{B}$

Une fonction $f : \mathbb{B}^n \to B$ est représentable avec 2^k bits (on donne les images des 2^n n-uplets sur $\mathbb{B}$). Donc au total il y a 2^{2^n} fonctions $\mathbb{B}^n \to \mathbb{B}$.

Idée. *Utiliser des automates qui reconnaissent le langage des* n*-uplets* $(x_1, x_2, \ldots, x_n)$ *pour lesquels* $f(x_1, x_2, \ldots, x_n) = 1$.

Un automate qui fait cela est l'arbre de décision de f.

Exemple 9.0.2. Si $f(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee x_3$, $Sol(f) = 111, 110, 001, 011, 101$
TODO: image

Pour éviter cet inconvénient, plutôt que de travailler avec les automates génériques on travaille avec des BDD (Binary Decision Diagrams).

Définition 9.2. Un BDD est un graphe avec une seule racine, acyclique, il possède deux feuilles, une étiquetée par 0 (false) et une étiquetée par 1 (true).

Les nœuds internes (nœuds de décision) on toujours deux fils. Notamment, si n est un nœud interne et x_i la variable relative à ce nœud, on nomme $\text{low}\,(n)$ le fils de n correspondant à l'assignation $x_i = 0$ et $\text{high}\,(n)$ celui correspondant à l'assignation $x_i = 1$.

Chaque nœud a une étiquette qui est un entier ≥ 0 et on a toujours $\text{etiquette}\,(n) > \text{etiquette}\,(\text{low}\,(n))$ et $\text{etiquette}\,(n) > \text{etiquette}\,(\text{high}\,(n))$.

Définition 9.3. Un BDD est dit réduit si :

1. On n'a jamais $\text{low}\,(n) = \text{high}\,(n)$

2. Il n'y a pas deux sous arbres isomorphes.

A partir d'un arbre de décision on peut obtenir un BDD réduit par la méthode suivante :

— Les feuilles qui ne se correspondent pas à des états terminaux ont comme étiquette 0.
— Les feuilles qui se correspondent à des états terminaux ont comme étiquette 1.
— Si n est un nœud interne :
  - si etiquette (low (n)) = etiquette (high (n)) = e, alors etiquette $(n) = e$.
  - s'l existe un nœud n' déjà étiqueté et tel que

$$\begin{aligned} \text{etiquette}\,(\text{low}\,(n')) &= \text{etiquette}\,(\text{low}\,(n)) \\ \text{etiquette}\,(\text{high}\,(n')) &= \text{etiquette}\,(\text{high}\,(n)) \end{aligned}$$

    Alors on pose etiquette (n) = etiquette (n')
  - Sinon l'étiquette de n est le plus petit entier qui n'a pas encore été utilisé.

Une fois qu'on a attribué une étiquette à tous les nœuds, on identifie les états ayant la même étiquette.

10 Langages de mots infinis et automates de Büchi

10.1 Langages de mot infinis

Notation 10.1. Soit A un alphabet.

On note A^ω comme l'ensemble des mots infinis sur A (un mot infini est une fonction : $\mathbb{N} \to A$).

On note $A^\infty = A^* \cup A^\omega$.

On étend la définition de certaines notions :

Soit $L \subseteq A^*$ un langage (de mots finis), on définit $L^\omega = \{x_0x_1\cdots \mid x_i \in L \setminus \{\varepsilon\}\}$ (généralisation de l'opérateur $*$ à une concaténation d'un nombre infini de mots de L.)

Définition 10.2 (Langages ω-rationnels)**.** La classe des langages ω-rationnels de A^∞ est la *plus petite* classe R de sous-ensembles de A^∞ qui satisfait :

1. $\emptyset \in R$ et $\forall a \in \Sigma, \{a\} \in R$;
2. Close par union finie ;
3. $\forall X \in R \cap A^*, \forall Y \in R \cap A^\infty, X \cdot Y \in R$;
4. $\forall X \in R \cap A^*, X^* \in R$ et $X^\omega \in R$.

Remarque 10.1.1. *La classe R contient en particulier tous les langages rationnels de A^*.*

Exemple 10.1.1. $A = \{a, b\}$.

— $L = \{w \in A^\omega \mid w$ a un nombre fini de $b\}$, on peut le définir par l'expression ω-rationnelle $(a+b)^*a^\omega$ ou $(a^*b)^*a^\omega$,
— $L = \{w \in A^\omega \mid w$ commence par a et a un nombre infini de $b\}$, avec l'expression : $a(a^*b)^\omega$.

Proposition 10.3 (Caractérisation des langages ω-rationnels)**.** *Un langage $L \subseteq A^\infty$ est w-rationnel si et seulement si L est l'union finie de langages de la forme XY^ω avec $X, Y \in R \cap A^*$ (X, Y sont des langages rationnels "classiques").*

Démonstration.

Soit Δ la classe des langages obtenue comme union finie de langages de la forme $XY^\omega, X, Y \in Rat(A^*)$.

— $\boxed{\Delta \subseteq R}$
 Par définition de R
 $$Y^\omega \in R \quad \textit{(par 4)}$$
 et
 $$XY^\omega \in R \quad \textit{(par 3)}$$
 et puisque R est fermé relativement à l'union finie, on a que tous les langages de Δ sont dans R, on a $\Delta \subseteq R$.

— $\boxed{R \subseteq \Delta}$

 Pour montrer l'inclusion inverse, on définit une classe $\mathcal{E}$. Avec
 $$L \in \mathcal{E} \iff \begin{cases} L \cap A^* \in \mathrm{Rat}(A^*) \\ L \cap A^\omega \in \Delta \end{cases}$$
 et on montre que $R \subseteq \mathcal{E} \subseteq \Delta$. Il faut vérifier que
 1. $\emptyset \in \mathcal{E}$
 2. $\forall a \in A, \{a\} \in \mathcal{E}$
 3. $\mathcal{E}$ est fermé par union finie
 4. $\forall X \in \mathcal{E} \cap A^*, \forall Y \in \mathcal{E} \cap A^\infty, X \cdot Y \in \mathcal{E}$
 5. $\forall X \in \mathcal{E} \cap A^*, X^* \in \mathcal{E}$ et $X^\omega \in \mathcal{E}$

 La vérification de ces propriétés est laissée en exercice.

 Puisque R est définie comme la plus petite classe qui satisfait les propriétés précédentes, on en déduit que $R \subseteq \mathcal{E}$.

□

10.2 Automates de Büchi

Définition 10.4. Un *automate de Büchi* est un quadruplet (Q, I, T, δ), avec $|Q| < \infty$, où Q, I, T, δ ont la même signification que pour les automates finis.

Définition 10.5. Un chemin infini est une suite infinie (q_i, a_i, q_{i+1}) tel que $\forall i, (q_i, a_i, q_{i+1}) \in \delta$ et le mot infini $a_0 a_1 \ldots$ est dit l'étiquette du chemin.

Définition 10.6. Un chemin $(q_0, a_0, q_1), (q_1, a_1, q_2), \ldots$ est *réussi* (ou acceptant) si $q_0 \in I$ et si $\exists$ une suite infinie d'indices $q_{i_J} \in T$.

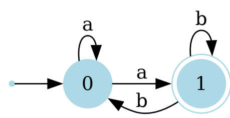
Donc un chemin est réussi si et seulement s'il passe un nombre infini de fois par des états terminaux.

On note en particulier qu'il doit exister un état $t \in T$ par lequel le chemin passe un nombre infini de fois.

Définition 10.7. Un mot $w \in A^\omega$ est accepté par un automate de Büchi si et seulement s'il est l'étiquette d'un chemin réussi.

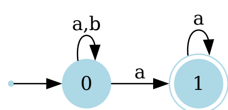
Définition 10.8. Un langage L de A^ω est reconnaissable s'il existe un automate de Büchi qui accepte exactement les mots de L.

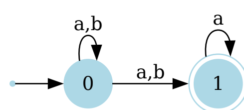
Exemple 10.2.1. L'automate suivant reconnait les mots qui commencent par a et qui ont un nombre infini de b :

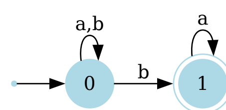


Exercice 10.2.1. Écrire un automate de Büchi qui reconnait les mots infinis sur $\{a, b\}$ ayant un nombre fini de b.

Trois automates possibles :







Définition 10.9. Un automate est dit émondé si tous ses états sont accessibles (on peut y accéder à partir d'un état initial) et co-accessibles (à partir de n'importe quel état, on peut arriver à un état final).

Les notions d'automate émondé, complet, déterministe se généralisent aux automates de Büchi.

Mais alors que les algorithmes pour rendre émondé ou complet un automate marchent aussi pour les automates de Büchi, il n'existe pas d'algorithme pour déterminiser un automate de Büchi.

En fait, il existe des langages reconnaissables de A^ω qui ne sont pas reconnus par un automate de Büchi déterministe.

La classe des langages reconnaissables par un automate de Büchi est fermée relativement à l'union finie et la méthode pour construire l'automate qui reconnait l'union est la même que pour les automates finis (on "met ensemble" les automates).

On verra que cette classe est aussi fermée relativement au complémentaire et à l'intersection.

Définition 10.10. Un automate est dit normalisé s'il a

— 1 seul état initial sans flèches rentrantes

— 1 seul état final sans flèches sortantes

Théorème 10.11 (lien avec le théorème de Kleene)**.** *$L \subseteq A^\omega$ est ω-rationnel si et seulement s'il est reconnu par un automate de Büchi.*

Démonstration.

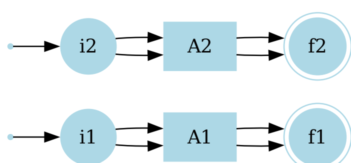
— $\boxed{\Leftarrow}$
Soit L reconnaissable, il existe $A = (Q, I, T, \delta)$ tel que $L = \mathcal{L}^\omega(A)$.
Soient $q, q' \in Q$. Pour chaque couple (q, q'), on définit $A_{qq'} = (Q, q, q', \delta)$
Soit $\mathcal{L}^*(A_{qq'})$ le langage reconnu par $A_{qq'}$ et $\mathcal{L}^+(A_{qq'}) = \mathcal{L}^*(A_{qq'}) \setminus \{\varepsilon\}$
Alors $L = \bigcup_{(q,q') \in I \times T} \mathcal{L}^*(q, q')(\mathcal{L}^+(q', q'))^\omega$. Montrons cette égalité.

$$\begin{aligned} w \in \bigcup_{(q,q') \in I \times T} \mathcal{L}^*(q, q')(\mathcal{L}^+(q', q'))^\omega \quad &\Longleftrightarrow \quad \exists i_0 \in I, \exists t_0 \in T, \text{ tels que } w \in \mathcal{L}^*(i_0, t_0)(\mathcal{L}^+(t_0, t_0))^\omega \\ &\Longleftrightarrow \quad w \text{ est l'étiquette d'un chemin qui passe} \\ &\qquad\ \text{un nombre infini de fois par l'état final } t_0 \\ &\Longleftrightarrow \quad w \in \mathcal{L}^w(A) = L \end{aligned}$$

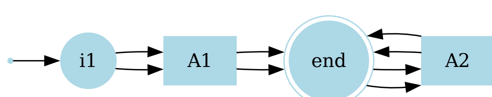
— $\boxed{\Rightarrow}$
Au vu de la caractérisation et du fait que la classe Rec est fermée relativement à l'union finie, il suffit de montrer que si $X, Y \in \mathrm{Rat}(A^+)$, alors $XY^\omega \in \mathrm{Rec}$.
D'après le théorème de Kleene, $X, Y \in \mathrm{Rat}(A^*)$, alors X, Y sont reconnaissables par des automates finis, donc

$$\exists A_1 \text{ tel que } X = \mathcal{L}(A_1)$$
$$\exists A_2 \text{ tel que } Y = \mathcal{L}(A_2)$$

De plus on peut choisir A_1 et A_2 normalisés.
Soit $A_1 = (Q_1, i_1, f_1, \delta_1)$ et $A_2 = (Q_2, i_2, f_2, \delta_2)$. Donc les automates ont la forme suivante :



Supposons d'abord que $\varepsilon \notin X$.
Alors on identifie les états f_1, i_2 et f_2 :



L'automate obtenu reconnait bien les mots XY^ω et il es défini formellement comme suit :

$$A = (Q_1 \cup Q_2 \setminus \{i_2, f_2\}, \{i_1\}, \{f_1\}, \delta_1 \cup \delta_2 \cup \eta_0 \cup \eta_1 \cup \eta_2)$$

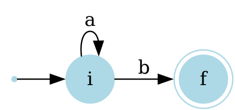
où :

$$\begin{aligned}
\eta_0 &= \{(f_1, a, f_1) \mid (i_2, a, f_2) \in \delta_2\} \\
\eta_1 &= \{(f_1, a, q) \mid (i_2, a, q) \in \delta_2\} \\
\eta_2 &= \{(q, a, f_1) \mid (q, a, f_2) \in \delta_2\}
\end{aligned}$$

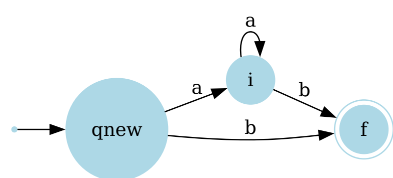
Dans le cas où $\varepsilon \in L_1$, alors on rend f_1 aussi initial.

□

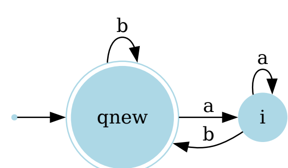
Exemple 10.2.2. $L_1 = \varepsilon, L_2 = \mathcal{L}(a^*b)$
L'automate classique pour L_2 :



On normalise l'automate en ajoutant un nouvel état initial *qnew* avec les mêmes transitions sortantes que l'ancien état initial.



Ensuite on identifie les états *qnew* et f :



Définition 10.12. Soit $L \subseteq \Sigma^*$ un langage de mots finis. On note :

$$\overrightarrow{L} = \{w \in \Sigma^\omega \mid w \text{ a une infinité de préfixes dans } L\}$$

Exemple 10.2.3.

1. $L = a^*b$, $\overrightarrow{L} = \emptyset$ (aucun mot infini)
2. $L = (ab)^*$, $\overrightarrow{L} = (ab)^\omega$ (un seul mot)

3. $L = (a^*b)^+$, $\overrightarrow{L} = (a^*b)^\omega$

Proposition 10.13 (Admis)**.** *Soit $A = (Q, I, T, \delta)$ un automate de Büchi déterministe, alors $\mathcal{L}^\omega(A) = \overrightarrow{\mathcal{L}^+(A)}$ où $\mathcal{L}^+(A)$ est le langage des mots finis reconnus par A comme un automate fini traditionnel.*

Corollaire 10.14. *$X \subseteq \Sigma^*$ est reconnu par un automate de Büchi déterministe si et seulement si $\exists L \in Rat(\Sigma^*)$ tel que $X = \overrightarrow{L}$.*

Proposition 10.15. *Le langage $X = (a+b)^*a^\omega$ $(|w|_b < \infty)$ ne peut pas être reconnu par un automate de Büchi déterministe.*

Démonstration. Par l'absurde, s'il y a un automate déterministe $\exists L \in \mathrm{Rat}(\Sigma)$ tel que tout mot de X a un nombre infini de préfixes dans L.

- — Prenons le mot ba^ω, il a un préfixe $ba^{i_1} \in L$.
- — Prenons le mot $ba^{i_1}ba^\omega$, il a un préfixe $ba^{i_1}ba^{i_2} \in L$.
- — Prenons le mot $ba^{i_1}ba^{i_2}a^\omega$, il a un préfixe $ba^{i_1}ba^{i_2}ba^{i_3} \in L$.

Alors on peut construire un mot infini :

$$ba^{i_1}ba^{i_2}\dots ba^{i_k}\dots$$

qui est l'étiquette d'un chemin qui passe un nombre infini de fois par un état final.

Ce mot serait reconnu par l'automate, or il contient un nombre infini de b. $\lightning$ □

10.3 Clôture sous intersection et complémentaire

Proposition 10.16. *La famille des langages reconnaissables par un automate de Büchi est fermée relativement à l'intersection.*

Démonstration. Soit L_1 et L_2 deux langages reconnaissable et soient

$$\begin{array}{rcl} A_1 & = & (Q_1, I_1, T_1.\delta_1) \\ A_2 & = & (Q_2, I_2, T_2.\delta_2) \end{array}$$

tels que $L_1 = \mathcal{L}^\omega(A_1)$ et $L_2 = \mathcal{L}^\omega(A_2)$.

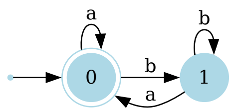
Alors l'automate $A = (Q, I, T, \delta)$:

$$\begin{array}{rcl} Q & = & Q_1 \times Q_2 \times \{1, 2\} \\ I & = & \{(i_1, i_2, 1) \mid i_1 \in I_1, i_2 \in I_2\} \\ T & = & \{(t, q, 1) \mid t \in T_1, q \in Q\} \\ \delta((q_1, q_2, x), a) & = & \begin{cases} (\delta_1(q_1, a), \delta_2(q_2, a), 2) & \text{si } q_1 \in T_1 \text{ et } x = 1 \\ (\delta_1(q_1, a), \delta_2(q_2, a), 1) & \text{si } q_2 \in T_2 \text{ et } x = 2 \\ (\delta_1(q_1, a), \delta_2(q_2, a), x) & \text{sinon} \end{cases} \end{array}$$

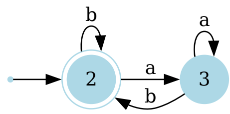
□

Exemple 10.3.1. Soit

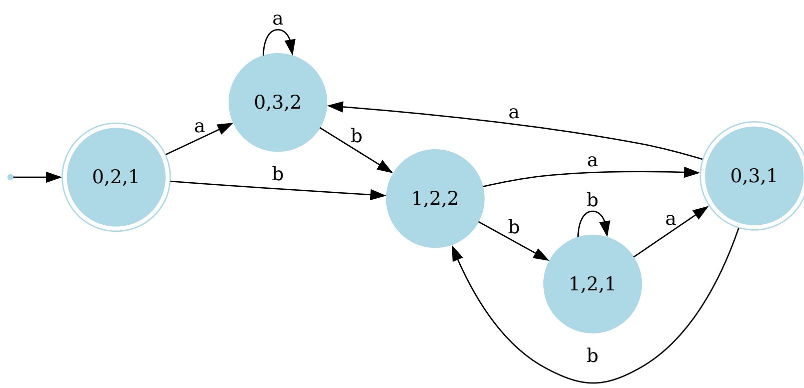
— $L_1 = \{u \in \{a,b\}^\omega \mid |u|_a = \infty\}$



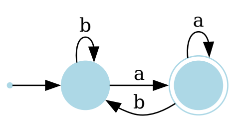
— $L_2 = \{u \in \{a,b\}^\omega \mid |u|_b = \infty\}$



— On cherche $L_3 = L_1 \cap L_2$



Remarque 10.3.1. *Le langage $L_3 = \{u \in \{a,b\}^\omega \mid |u|_a = \infty$ et $|u|_b = \infty\}$, i.e. , le langage de mots de $\{a,b\}^\omega$ avec un nombre infini de a et de b, reconnu par l'automate suivant :*



Notation 10.17. Soit $w \in \Sigma^*$, on note $\inf(w) = \{x \in \Sigma \mid |w|_x = \infty\}$

Théorème 10.18 (Factorisation Ramseyenne)**.** *Soit $f : \mathbb{N} \times \mathbb{N} \to C$, où C est un ensemble fini,* *alors il existe une suite infinie d'entiers $E = i_0, i_1, \ldots, i_n, \ldots$ telle que $f(E \times E)$ soit un singleton, i.e. $\forall i_{j_1}, i_{e_1}, i_{j_2}, i_{e_2}$ dans E, $f(i_{j_1}, i_{e_1}) = f(i_{j_2}, i_{e_2})$.*

Démonstration. On construit une suite de sous-ensembles de $\mathbb{N}$, $T_0, T_1, \ldots$, de la maniere suivante : $T_0 = \mathbb{N}$. Si on a déjà T_i, on construit T_{i+1} comme suit.

Puisque les couleurs sont en nombre fini, il existe une couleur c_0 et une suite $u_0, v_1, v_2, \ldots$ tel que $f(u_0, v_i) = c_0 \forall i \geq 1$ (il existe un nombre infini d'arcs sortant de u_0 de couleur c_0) alors $T_{i+1} = \{v_1, v_2, \ldots\}$

Soit $u_i = \min(T_1)$ et considérons $u_0, u_1, \ldots$. Par construction $\forall i \in \mathbb{N}, \forall j \in \mathbb{N}, f(u_i, u_{i+j}) = c_i$ puisque les couleurs sont en nombre fini, il doit exister une suite $i_0, i_1, \ldots$ telle que $f(u_{i_J}, u_{i_{J+1}}) = c_{i_0}$ et donc la suite $u_{i_0}, u_{i_1}, \ldots$ est la suite cherchée. □

Notation 10.19. Soit $A = \langle Q, I, F, \delta \rangle$ un automate, alors pour un d'états $p, q \in Q$ et un mot $w \in \Sigma^*$, alors on *note* $p \xrightarrow{w} q$ s'il existe un chemin de p à q avec w comme étiquette et on note $p \xrightarrow[t]{w} q$ si le chemin passe par au moins un état terminal.

Définition 10.20. Soit X un langage reconnaissable et $A = (Q, I, T, \delta)$ un automate de Büchi qui reconnait X.

On définit une relation sur Σ^* :

$$\begin{aligned} w \sim w' \iff & \forall p, q \in Q, p \xrightarrow{w} q \iff p \xrightarrow{w'} q \\ & \text{et } p \xrightarrow[t]{w} q \iff p \xrightarrow[t]{w'} q \end{aligned}$$

Proposition 10.21. *1. La relation $\sim$ est une relation d'équivalence.*

2. La relation $\sim$ est une relation d'équivalence d'index fini, son nombre de classes ne peut pas dépasser $3^{|Q|^2}$.

3. La relation $\sim$ est une congruence.

Démonstration.

1. trivial
2. Chacune des $|Q|^2$ coupes d'états distingue 3 types de mots :
   — les mots qui ne sont pas l'étiquette d'un chemin de p à q,
   — les mots qui sont l'étiquette d'un chemin s de p à q qui ne passe par aucun état terminal,
   — les mots qui sont l'étiquette d'un chemin s de p à q qui passe par au moins 1 un état terminal.
3. $\sim$ est une congruence.
   Si $w \sim w'$ et $u, v \in \Sigma^*$ alors $uwv \sim uw'v$
   TODO

□

Remarque 10.3.2. $\Sigma^* / \sim$ *est un monoïde fini.*

Et donc $\forall w \in \Sigma^*, [w]$ est un langage rationnel, en effet, $[w]$ est l'image inverse d'un singleton de $\Sigma^* / \sim$ par le morphisme de projection $\varphi : \Sigma^* \to \Sigma^* / \sim$ tel que $\forall w \in \Sigma^*,\ f(w) = [w]$. Un singleton est rationnel et l'image inverse d'un rationnel par un morphisme est aussi un rationnel.

On considère la famille de langages de Σ^ω de la forme $[u][v]^\omega$ où $[u]$ et $[v]$ sont des classes d'équivalence de $\sim$.

On va montrer que les langages $[u][v]^\omega$ constituent une partition de Σ^ω plus fine que la partition $< X, \mathscr{C}(X) >$, donc que X et $\mathscr{C}(X)$ sont l'union finie d'ensembles de la forme $[u][v]^\omega$.

Proposition 10.22. *Soit $Y = [u][v]^\omega$ pour un choix quelconque de u et v, alors $Y \cap X = \emptyset$ ou $Y \subseteq X$.*

Démonstration. Supposons $Y \cap X \neq \emptyset$ et soit $x \in Y \cap X$, alors x est l'étiquette d'un chemin réussi de A de la forme

$$q_0 \xrightarrow{u_0} q_1 \xrightarrow{v_1} q_2 \xrightarrow{v_2} q_3 \dots$$

avec $u_0 \in [u]$ et $\forall i, v_i \in [v]$, de plus, une infinité de chemins $q_i \xrightarrow{v_i} q_{i_1}$ passent par au moins un état terminal.

Soit $y \in Y = [u][v]^\omega$. On peut donc écrire y comme :

$$y = u_0' v_1' v_2' \dots$$

avec $u_0' = [u]$ et $\forall i, v_i' = [v]$.

Mais alors $u_0 \sim u_0'$ et $\forall i, v_i \sim v_i'$ et donc

$$q_0 \xrightarrow{u_0'} q_1 \xrightarrow{v_1'} q_2 \xrightarrow{v_2'} q_3 \dots$$

avec une infinité de chemins $q_i \xrightarrow{v_i'} q_{i_1}$ passent par au moins un état terminal grâce à l'équivalence $v_i \sim v_i'$.

Ainsi y est aussi l'étiquette d'un chemin réussi de A, c'est-à-dire A reconnait y, donc $y \in X$, donc $Y \subseteq X$. □

Proposition 10.23. $\forall x \in \Sigma^\omega, \exists u, v \in \Sigma^*, x \in [u][v]^\omega$.

Démonstration. Écrivons $x = a_0 a_1 \dots a_n \dots$ où $a_i \in \Sigma$.

On définit une fonction $f : \mathbb{N} \times \mathbb{N} \to \Sigma / \sim$

$$f(i,j) = [a_i a_{i+1} \dots a_{j-1}], i < j$$

par le théorème de Ramsey, il existe une suite $I = i_0, i_1, \dots$ telle que $f(I)$ est un singleton. En particulier,

$$f(i_0, i_1) = f(i_1, i_2) = f(i_2, i_3) \ \dots$$

Mais alors, soit $u - q_0 \dots q_{i_0 - 1}$, $v_j = a_{i+j} \dots a_{i_{j+1}-1}$ et $x = a_0 a_1 \dots a_{i_0 - 1} a_{i+0}$ mais

$$\begin{array}{rcl} f(i_0, i_1) & = & [a_{i_0} a_{i_0+1} \dots a_{i_1 - 1}] \\ f(i_1, i_2) & = & [a_{i_1} a_{i_1+1} \dots a_{i_2 - 1}] \\ f(i_2, i_3) & = & [a_{i_2} a_{i_2+1} \dots a_{i_3 - 1}] \end{array}$$

en fait, on a que $v_j \sim v_1 \forall j \geq 1$ et donc $x \in [u][v_1]^\omega$. □

Théorème 10.24. *$\forall X$ ω-rationnel, on a*

$$\mathscr{C}(X) = \bigcup_{\substack{(u,v) | \\ [u][v]^\omega \cap X = \emptyset}} [u][v]^\omega$$

Démonstration.

— $\boxed{\supseteq}$
Évidente car on fait une union d'ensembles qui sont tous contenus dans $\mathscr{C}(X)$.

— $\boxed{\subseteq}$
Soit $y \in \mathscr{C}(X)$.

$$\exists u, v, \text{ tels que } y \in [u][v]^\omega \quad (\textit{Propositon 10.23})$$

alors $[u][v]^\omega \cap \mathscr{C}(X) \neq \emptyset$ et donc par la Proposition 10.22 on a $[u][v]^\omega \cap X = \emptyset$ et donc y est bien dans l'un des ensembles de l'union.

□

Corollaire 10.25. *La classe des langages reconnaissables par un automate de Büchi est fermée relativement au passage au complémentaire.*

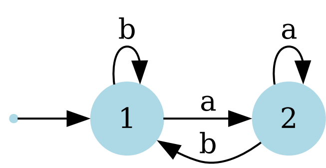
10.4 Automates de Muller

Pour résoudre le problème de l'absence d'un algorithme de détermination des automates de Büchi on introduit des généralisations comme les automates de Muller.

Définition 10.26. Un automate de Muller est un quadruplet $(Q, I, \mathcal{T}, \sigma)$ où $\mathcal{T}$ (la table de l'automate) est un ensemble de sous-ensembles de Q : $\mathcal{T} \subseteq \mathcal{P}(Q)$.

Soit $w \in \Sigma^\omega$, w est accepté par A si w est l'étiquette d'un chemin γ tel que $\inf(\gamma) \in \mathcal{T}$ où $\inf(\gamma)$ est l'ensemble d'états de Q par qui γ passe infiniment souvent.

Exemple 10.4.1. L'automate suivant



avec la table $\mathcal{T} = \{\{2\}\}$ est un automate de Muller déterministe qui reconnait les mots avec un nombre fini de b (ceci n'est pas possible avec un automate déterministe de Büchi).

11 Automates cellulaires

Définition 11.1. Un automate cellulaire est défini par :

— Un entier positif d représentant la dimension de l'espace $(\mathbb{Z}^d)$

— Un ensemble fini S d'états

— Un *voisinage*, *i.e.* un ensemble $V = (\vec{v_1}, \vec{v_1}, \ldots, \vec{v_m})$ de vecteurs de $\mathbb{Z}^d$ qui représente la position relative des voisins (relative à la position de la cellule). Les voisins de la cellule $\vec{r}$ sont $\vec{r} + \vec{v_1}, \ldots, \vec{r} + \vec{v_m}$.

— Une fonction $f : S^m \to S$ de transition locale qui définit l'état d'une cellule à l'instant $t + 1$ en fonction des états de ses voisins à l'instant t.

Les automates cellulaires sont synchrones, et homogènes, dans le temps et dans l'espace, *i.e.* , ils appliquent la règle simultanément a toutes les cellules, la règle locale de transition ne change pas avec le temps et ils appliquent la meme règle a toutes les cellules.

Définition 11.2. Une configuration est une fonction $\mathbb{Z}^d \to S$ qui décrit l'état courant de chaque cellule.

Si C est l'ensemble des configurations, alors un automate cellulaire définit une fonction $G : C \to C$ où si $c \in C$, alors $G(c) = c'$ est la configuration obtenue de c en appliquant la règle f à toutes les cellules.

En général, on itère la fonction G et pour une configuration c on s'intéresse à la suite :

$$G^0(c) = c, G^1(c), G^2(c), \dots \text{ où } G^i(c) = G(G^{i-1}(c))$$

Définition 11.3. Un point fixe (ou nature morte) est une configuration c telle que $\forall k, G^k(c) = c$.

Définition 11.4. Une configuration c est dite périodique (dans le temps), ou oscillateur, si $\exists k$ tel que $G^k(c) = c$. Le plus petit entier k tel que $G^k(c) = c$ est dit la période d'oscillation.

Définition 11.5. Une configuration est dite ultimement fixe si $\exists k$ tel que $\forall j,\ G^{k+j}(c) = G^k(c)$.

Définition 11.6. Une configuration est dite ultimement périodique si $\exists k, \exists l > 0$ tel que $G^{k+l}(c) = G^k(c)$, *i.e.* à partir de la k-ème itération on rentre dans un cycle de longueur l.

Exemple 11.0.1. TODO

11.1 Unicité des automates cellulaires

Théorème 11.7. *Si deux automates cellulaires A et B réalisent la même fonction G, alors A et B ne peuvent différer que par leurs voisinages et les deux voisinages ne peuvent être différents que par la présence (ou pas) de voisins inutiles (c'est-à-dire des voisins dont l'état n'a aucune influence sur la valeur de f).*

Exemple 11.1.1. TODO

11.2 Le jeu de la vie

Exemple 11.2.1. Pour $d = 2$, Game of Life de Conway [Gar70] est défini par :

— $S = \{0, 1\}$. L'état 0 correspond à l'état d'une cellule "morte" et 1 à celui d'une cellule "vivante".

— $V = ((0, 1), (1, 0), (0, -1), (-1, 0), (1, 1), (1, -1), (-1, 1), (-1, -1))$, dit le voisinage de Moore de rayon 1.

— Fonction de transition :

- une cellule vivante reste vivante si et seulement si elle a 2 ou 3 voisins vivants ;
- une cellule morte devient vivante si et seulement si elle a 3 voisins vivants.

Une implémentation sur navigateur est disponible sur : `https://conwaylife.com/`.

Définition 11.8. On appelle automates cellulaires élémentaires les automates cellulaires qui respectent :

— $d = 1$

— $S = \{0, 1\}$

— $V = (-1, 0, 1)$

Il y a exactement $2^8 = 256$ automates élémentaires.

Wolfram a introduit une numérotation [Wol83] pour les automates cellulaires élémentaires où chaque automate est associé à un entier entre 0 et 255.

Puisque on peut définir l'automate juste en étudiant son comportement sur les contextes constitués de trois cellules voisines, il suffit d'associer une étiquette à ce comportement pour caractériser tout l'automate cellulaire. Ainsi, on commence par trier les huit contextes possibles de trois cellules ainsi :

■■■ ■■□ ■□■ ■□□ □■■ □■□ □□■ □□□

Si on considère chaque contexte de trois cellules comme un nombre binaire encodé sur 3 bits, ce tri correspond à un tri décroissant sur la valeur de ces nombres (de 7 à 0). Pour chacun de ces contextes on peut donc regarder quel sera l'état de la cellule du milieu à la génération suivante.

Par exemple, pour l'automate cellulaire qui réalise la fonction $f(x_{i-1}, x_i, x_{i+1}) = x_{i-1}$ **xor** x_{i+1} (la valeur de la cellule i à l'instant $t + 1$ est le ou exclusif des valeurs des cellules $i - 1$ et $i + 1$ à l'instant t) on a :

■■■	■■□	■□■	■□□	□■■	□■□	□□■	□□□
□	■	□	■	■	□	■	□

Maintenant, on peut voir ceci comme un nombre binaire sur 8 bits. Ainsi, pour cet automate cellulaire on obtient :

■■■	■■□	■□■	■□□	□■■	□■□	□□■	□□□
□	■	□	■	■	□	■	□
0	1	0	1	1	0	1	0

Ainsi cet automate est numéroté par $01011010_2 = 90_{10}$. C'est l'automate élémentaire noté W_{90}.

11.3 Configurations finies et configurations périodiques

Définition 11.9. Soit $s \in S$ un état, et c une configuration, on appelle s-support de c l'ensemble des cellules qui ont un état différent de s.

Notation 11.10. Souvent, on impose qu'un état q_0 soit un état *quiescent* (généralement, c'est l'état 0). Un état quiescent est aussi stable par f, *i.e.* $f(\underbrace{q_0, \ldots, q_0}_{m}) = q_0$.

Définition 11.11. Si q_0 est un état quiescent on appelle configurations finies (ou plus précisément q_0-finies) les configurations ayant un q_0-support fini.

Puisque un état quiescent est par stable par définition, il en suit que si c est une configuration finie alors $G(c)$ l'est aussi. Si donc $F \subset C$ dénote l'ensemble de toutes les configurations finies, la restriction de G à F, notée G_F, est une fonction de F à F.

Définition 11.12. Soit $\vec{v}$ un vecteur de $\mathbb{Z}^d$, la translation $\tau_{\vec{v}}$ est une fonction d'automate cellulaire où le voisinage contient seulement le vecteur $-\vec{v}$ et la fonction de transition locale f est l'identité, autrement dit, la valeur d'une cellule $\vec{r}$ à l'instant $t+1$ est simplement la valeur de la cellule $\vec{r} - \vec{v}$ à l'instant t.

Définition 11.13. On note $\vec{\sigma}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ le vecteur unitaire qui a la i-ème composante égale à 1 et toutes les autres égales à 0. Les translation $\tau_{\vec{\sigma}_i}$ (pour i dans $\{1, \ldots, d\}$) sont nommées shifts élémentaires.

Puisque tout vecteur $\vec{v}$ est une combination linéaire des vecteurs $\vec{\sigma}_1, \vec{\sigma}_2, \ldots, \vec{\sigma}_d$, il en suit que toute translation peut être exprimée comme une combinaison linaires de shifts élémentaires.

Définition 11.14. Si $\vec{v}$ est un vecteur, on dit qu'une configuration c est $\vec{v}$-périodique si $c(\vec{n}) = c(\vec{n} + \vec{v})$ pour tout $\vec{n} \in \mathbb{Z}^d$.

Exemple 11.3.1. TODO

Définition 11.15. Une configuration c est dite spatialement périodique si $\exists \vec{v}$ tel que c est $\vec{v}$-périodique.

Définition 11.16. Une configuration c est dite totalement périodique si $\forall i \in \{1, \ldots, d\} \, \exists k_i \in \mathbb{Z}$ tel que c est $k_i\sigma_i$-périodique.

Proposition 11.17. *Si G est la fonction d'un automate cellulaire et τ une translation, alors $G \circ \tau = \tau \circ G$.*

Puisque les fonctions d'automates cellulaires commutent avec les translations, il en découle que si c est totalement périodique, alors $G(c)$ est aussi totalement périodique.

Donc si P représente l'ensemble de toutes les configurations (totalement) périodiques, alors la restriction de G à P, notée G_P, est une fonction de P à P.

Les configurations finies ainsi que les configurations (totalement) périodiques sont importantes parce que elles sont les seules sur lesquelles on peut tester le comportement d'un automate cellulaire. De plus, comme on le verra dans la suite, les deux fonctions G_F et G_P donnent des renseignements sur la surjectivité et l'injectivité de G.

11.4 Une topologie sur l'espace des configurations $S^{\mathbb{Z}^d}$

Définition 11.18. Soit $c_1, c_2, \ldots$ une suite de configurations, on dit que la suite converge vers la configuration c (ou qu'elle a comme limite la configuration c) si

$$\forall \vec{n} \in \mathbb{Z}^d, \exists i \text{ tel que : } c_j(\vec{n}) = c(\vec{n}) \forall j > i$$

Proposition 11.19. *(Compacité de $S^{\mathbb{Z}^d}$.)*

Toute suite de configurations possède une sous-suite convergente.

Démonstration. La démonstration est similaire à celle de la factorisation de Ramsay. (Une configuration c est une fonction de $\mathbb{Z}^d \to S$ avec S fini, la preuve qu'on a vue est pour les fonctions $\mathbb{N}^2 \to M$, avec M fini). $\square$

Proposition 11.20. *(Les fonctions d'automates cellulaires sont continues.)*

Soit $c_1, c_2, \ldots$ *une suite de configurations qui converge vers* c*, alors* $G(c_1), G(c_2), \ldots$ *est une suite de configurations qui converge vers* $G(c)$*.*

Démonstration. La démonstration n'est pas compliquée et elle est laissée en exercice. □

Proposition 11.21. *(Les configurations finies et totalement périodiques sont denses.)*

Pour toute configuration $c \in S^{\mathbb{Z}^d}$*, il existe une suite* $c_1, c_2, \ldots$ *de configurations finies et une suite* $p_1, p_2, \ldots$ *de configurations (totalement) périodiques telles que :*

$$\lim_{i\to\infty} c_i = c \text{ et } \lim_{i\to\infty} p_i = c$$

Démonstration. Soit c une configuration quelconque. Soit $\vec{r_0}, \vec{r_1}, \ldots$ un dénombrement des cellules de $\mathbb{Z}^d$ (c'est-à-dire une fonction qui attribue un numéro ordinal à chaque d-uplet de $\mathbb{Z}^d$, ceci est possible car $\mathbb{Z}^d$ est dénombrable) et pour tout $i \in \mathbb{N}$, soit c_i la configuration définie par :

$$c_i(r_j) = \begin{cases} c(r_j) & \text{si } j \leq i \\ q & \text{sinon} \end{cases}$$

où q est l'état quiescent.

Il est évident que $c_i \in F$ pour tout $i \in \mathbb{N}$ et que $\lim_{i\to\infty} c_i = c$.

Par ailleurs pour obtenir une suite de configurations périodiques qui convergent vers c, on peut construire une suite de configurations $p_1, p_2, \ldots$ où p_i coïncide avec c dans l'hypercube centré à l'origine et de coté $2i+1$ et a une période $2i+1$ dans toutes les directions (on répète cet hypercube à l'infini dans toutes les directions).

Les configurations p_i sont périodiques par construction et évidemment $\lim_{i\to\infty} p_i = c$. □

11.5 Injectivité et surjectivité des automates cellulaires

Définition 11.22. Un automate cellulaire est dit injectif (respectivement surjectif, bijectif) si sa fonction G est injective (respectivement surjective, bijective) .

Théorème 11.23. *On a les implications suivantes :*

1. G *injective* $\Longrightarrow G_F, G_P$ *injectives*
2. G_F *surjective ou* G_P *surjective* $\Longrightarrow G$ *surjective*
3. G_P *est injective* $\Longrightarrow G_P$ *surjective*

Démonstration.

1. Trivial, la restriction d'une fonction injective est toujours injective.
2. Dans le cas où G_F est surjective :

   Soit c une configuration quelconque, montrons que c possède une anti-image. On sait par la Proposition 11.21 qu'il existe une suite de configurations finies $c_1, c_2, \ldots, c_m$ telle que $\lim_{i\to\infty} c_i = c$. Puisque chaque c_i est une configuration finie et comme G_F est surjective,

   $$\forall i \in \mathbb{N}, \exists e_i \in F \text{ telle que } G_F(e_i) = c_i = G(e_i)$$

   Soit la suite $e_1, e_2, \ldots$, d'après la proposition 11.19, cette suite doit posséder une sous-suite convergente. Soit $e_{i_1}, e_{i_2}, \ldots, e_{i_j} \ldots$ une sous-suite convergente et soit e sa limite. G est continue par la proposition 11.20, donc la limite de la suite $G(e_{i_1}), G(e_{i_2}), \ldots, G(e_{i_j}) \ldots$ est $G(e)$.

Mais par ailleurs,

$$\begin{aligned} \lim_{j\to\infty} (G(e_{i_j})) &= \lim_{j\to\infty} (c_{i_j}) \\ &= \lim_{i\to\infty} (c_i) \\ &= c \end{aligned}$$

On a donc $G(e) = c$ et c possède donc une anti-image.
La preuve est analogue dans l'hypothèse que G_P est surjective.

3. Soit c une configuration totalement périodique. Par définition $\exists k_1, k_2, \ldots, k_d \in \mathbb{Z}$ tels que :

$$\forall i \in \{1, \ldots, d\}\,,\ \text{la configuration } c \text{ est } k_i\sigma_i\text{-périodique, c'est à dire, } c(\vec{n}) = c(\vec{n} + k_i\sigma_i) \quad (3)$$

Soit K l'ensemble de toutes les configurations qui satisfont (3). On a que $K \subset P$ et K est fini, car chaque configuration de K est uniquement identifiée par l'état des cellules dans un hyper-parallélépipède d-dimensionnel de cotés $k_1, k_2, \ldots, k_d$ (donc $|K| = |S|^{k_1 k_2 \ldots k_d}$)
Puisque G commute avec les translations, on a que $G(K) \subseteq K$.
Considérons la restriction $G_{\restriction K}$. Cette fonction est injective car c'est une restriction de G_P à un sous-ensemble de P et G_P est injective par hypothèse. Mais toute fonction injective sur un ensemble fini est aussi surjective, donc chaque élément de K (dont c) a un antécédent dans K, donc $\exists e \in K \subsetneq P$ telle que $G(e) = c$, et donc G_P est surjective.

□

Corollaire 11.24. *Si un automate cellulaire est injectif, alors il est surjectif (et donc bijectif).*

Démonstration.

$$\begin{aligned} G \text{ injective} &\implies G_P \text{ injective} \quad (1) \\ &\implies G_P \text{ surjective} \quad (3) \\ &\implies G \text{ surjective} \quad (2) \end{aligned}$$

□

Définition 11.25. On dit qu'un automate cellulaire est réversible s'il est bijectif et si G^{-1} est une fonction d'automate cellulaire.

Remarque 11.5.1. *Par définition, si un automate cellulaire est réversible alors il est bijectif.*

Proposition 11.26. *Si A est un automate cellulaire bijectif, alors il est réversible.*

Démonstration. La preuve montre que la fonction G^{-1} est aussi une fonction d'automate cellulaire (on peut définir la règle locale d'évolution de G^{-1} en fonction de l'état de cellules appartenant à un voisinage *borné*). □

Corollaire 11.27. *G injective $\implies$ G_F surjective.*

Démonstration.

$$\begin{aligned} G \text{ injective} \quad &\Longrightarrow \quad G \text{ bijective} \quad (\textit{Cor. 11.24}\ ) \\ &\Longrightarrow \quad G \text{ réversible} \quad (\textit{Prop. 11.26}\ ) \end{aligned}$$

Si q est l'état quiescent pour G, alors q est aussi l'état quiescent de G^{-1}. (Si toutes les cellules du voisinage sont à q, alors G^{-1} met la cellule à l'état q.)

Donc si c est une configuration finie quelconque, alors la configuration $e = G^{-1}(c)$ (la configuration obtenue en appliquant l'automate inverse) est aussi finie, et alors $G(e) = c$ et c a bien une anti-image finie. □

TODO: add graph

TODO: RM : peut-être mettre le graphe à la fin en sorte de récapitulatif. En particulier l'implication due au théorème du jardin d'Eden (GF inj ssi G surj), n'est pas encore démontrée à ce point.

11.5.1 Surjectivité et équilibre

Définition 11.28. Une configuration qui n'a pas d'anti-image est dite un *Jardin d'Éden* (JDE, ou GOE pour Garden of Eden) .

Remarque 11.5.2. *L'existence de Jardins d'Éden est équivalente à la non surjectivité de l'automate cellulaire.*

Exemple 11.5.1. L'automate élémentaire W_{110} n'est pas surjectif. TODO

Définition 11.29. On appelle motif tout ensemble fini D de cellules ainsi que la description des états des cellules de D. Formellement, un motif est un couple (D, φ) où D est un ensemble fini de cellules (dit le domaine du motif) et φ une fonction $D \to S$.

Définition 11.30. Un motif est dit un orphelin si toute configuration le contenant est un JDE.

Exercice 11.5.1. Montrer que toute configuration contenant le motif 01010 est un Jardin d'Éden pour W_{110} (le motif 01010 est un orphelin).

Dans le cas de W_{110}, on a vu que la non-surjectivité et due au fait que la distribution des images des huit contextes n'est pas équilibrée (il y a cinq contextes ayant comme image 1 mais seulement trois ayant comme image 0). Ceci est vrai en général, en effet, d'après la proposition suivante, l'équilibre de la distribution est une condition nécessaire (mais pas suffisante) pour la surjectivité,

Proposition 11.31. *Si un automate cellulaire est surjectif alors pour tout état $q \in S$ on a : $\left|f^{-1}(q)\right| = |S|^{n-1}$ où S est l'ensemble de ses états et n est la taille du voisinage. En d'autres termes, les images par f des $|S|^n$ possibles contextes sont distribuées de façon équilibrée sur le $|S|$ états.*

Il existe des automates cellulaires (même élémentaires) qui ont une distribution équilibrées $\left|f^{-1}(0)\right| = \left|f^{-1}(1)\right| = 2^{3-1} = 4$ mais qui ne sont pas surjectifs, par exemple parce qu'ils n'ont pas une distribution équilibrée pour les quatre patterns de longueur 2 (00, 01, 10, 11).

Proposition 11.32. *Il existe un JDE si et seulement si il existe un orphelin.*

Démonstration.

— $\boxed{\Leftarrow}$
S'il existe un orphelin D, alors toute configuration qui inclut l'orphelin est un JDE par définition.

— $\boxed{\Rightarrow}$
Si un JDE existe, alors la fonction G de l'automate n'est pas surjective, et donc par la partie 2 du Théorème 11.23, G_F n'est pas surjective, c'est à dire, $\exists$ une configuration finie qui n'a pas d'anti-image. Le support de cette configuration est un orphelin.

□

Définition 11.33. Deux motifs finis J_1 et J_2 sont dits jumeaux si ils ont le même domaine et si, en remplaçant J_1 par J_2 dans n'importe quelle configuration qui contient J_1, on génère la même configuration.

Théorème 11.34 (du Jardin d'Éden [Moo62, Myh63])**.** *G_F injective $\iff$ G surjective. Autrement dit, un automate cellulaire possède un JDE si et seulement si il possède deux jumeaux.*

Démonstration.

— $\boxed{\Rightarrow}$
Supposons que G ne soit pas surjective, alors il existe un JDE et donc un orphelin. A partir de l'existence de cet orphelin, on doit montrer l'existence d'un couple de jumeaux. On ne donnera pas une preuve complète mais on prendra comme exemple l'automate élémentaire W_{110} pour comprendre l'argument de la preuve, qui est un simple argument de comptage.
On a déjà montré que *01010* est un orphelin pour W_{110}. Soit k un entier quelconque et considérons l'ensemble C_k de toutes les configurations finies qui ont un support de taille inférieur ou égale à $5k-2$, alors $|C_k| = 2^{5k-2} = \frac{32^k}{4}$
Si c est une configuration de C_k, alors son image $G(c)$ est une configuration avec un support de taille $5k$. Si on découpe ce support de taille $5k$ en k segments de taille 5, il y a au plus 31 choix pour chaque bloc car *01010* est interdit. Alors l'ensemble $G(C_k)$ a une taille d'au plus 31^k. Or $\frac{32^k}{4} > 31^k$ pour k suffisamment grand. On a donc que pour k suffisamment grand il y aura plus de configurations dans C_k que dans son image $G(C_k)$, donc il existe des configurations de C_k qui ont la même image (des jumeaux).
TODO: RM : Add figure
On pourra s'inspirer de cette exemple pour donner une preuve complète, pour n'importe quel automate qui possède un orphelin dans n'importe quel nombre de dimensions $d > 1$.

— $\boxed{\Leftarrow}$
On donnera la preuve pour le cas $d = 2$ car dans ce cas il est possible faire des dessins qui aident à comprendre la demonstration, cependant la généralisation à un nombre quelconque de dimension est immédiate.
Soient J_1 et J_2 deux jumeaux, montrons qu'il existe un orphelin.
Soit n un entier tel que :
  1. le domaine de J_1 et J_2 est contenu dans un carré de taille n ;
  2. le voisinage de chaque cellule ne contient que des cellules à distance inférieure à n.

Soit m un autre entier et considérons l'ensemble E de tous les motifs ayant un domaine inclus dans un carré de taille $mn \times mn$. On veut établir que pour m suffisamment grand au moins l'un de ces motifs n'a pas de prédécesseur (il est donc un orphelin et toute configuration qui le contient est un Jardin d'Éden).

Le nombre total de ces motifs ayant un domaine inclus dans un carré de taille $mn \times mn$ est évidemment $|S|^{mn\times mn} = \left(|S|^{n\times n}\right)^{m^2}$.

Par ailleurs, les prédécesseurs potentielles des motifs de E ont tous un domaine contenu dans un carré de taille $(m+2)n \times (m+2)n$. En effet, si une cellule se trouve en dehors de ce carré, tous ses voisins sont dans l'état quiescent et la cellule restera donc dans l'état quiescent (l'état quiescent est stable).

TODO: ADD FIGURE

Soit P l'ensemble des motifs de taille $(m+2)n \times (m+2)n$. Puisque J_1 et J_2 ont le même successeur, il suffit de considérer comme prédécesseurs potentiels des motifs de E uniquement les motifs de P qui ne contiennent pas J_2. Notons P' l'ensemble de des motifs de taille $(m+2)n \times (m+2)n$ qui ne contiennent pas J_2.

L'ensemble P' contient au plus $\left(|S|^{n\times n} - 1\right)^{(m+2)\times(m+2)}$ motifs. En effet, ceci est le nombre total de "pavages" de la grille de taille $(m+2)n \times (m+2)n$ avec des motifs de taille $n \times n$ différents de J_2 et P' est un sous-ensemble des motifs obtenus par ces pavages.

Or, pour toute constante $k > 1$ (et en particulier pour la constante $k = |S|^{n\times n}$) on a :

$$\lim_{m\to+\infty} \frac{k^{m\times m}}{(k-1)^{(m+2)\times(m+2)}} = +\infty$$

Ceci peut être vérifié facilement en passant aux exponentielles :

$$\lim_{m\to+\infty} \frac{k^{m\times m}}{(k-1)^{(m+2)\times(m+2)}} = \lim_{m\to+\infty} \frac{\left(e^{\log k}\right)^{m\times m}}{\left(e^{\log(k-1)}\right)^{(m+2)\times(m+2)}} = \lim_{m\to+\infty} e^{m^2\log k-(m+2)^2\log(k-1)}$$

Il suffit donc d'étudier la limite d'un polynôme de degré 2 : $\lim\limits_{m\to+\infty} m^2\log k-(m+2)^2\log(k-1)$, et ce n'est pas difficile de conclure que cette limite est $+\infty$ (le coefficient du terme de degré 2 est positif).

On en déduit qu'au grandir de m la taille de E grandit beaucoup plus vite que la taille de P' et donc que pour un m suffisamment grand, il existe des motifs de E qui n'ont pas de prédécesseur dans P', c'est-à-dire des orphelins.

□

Références

[Ang87] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2) :87–106, 1987.

[Dav10] Julien David. The average complexity of moore's state minimization algorithm is o(n log log n). In *International Symposium on Mathematical Foundations of Computer Science*, 2010.

[Gar70] Martin Gardner. Mathematical games. *Scientific American*, 223(4) :120–123, 1970.

[KMP77] Donald E. Knuth, James H. Morris, Jr., and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2) :323–350, 1977.

[Moo62] Edward F. Moore. Machine models of self-reproduction. 1962.

[Myh63] John R. Myhill. The converse of moore's garden-of-eden theorem. 1963.

[Wol83] Stephen Wolfram. Statistical mechanics of cellular automata. *Rev. Mod. Phys.*, 55 :601–644, Jul 1983.