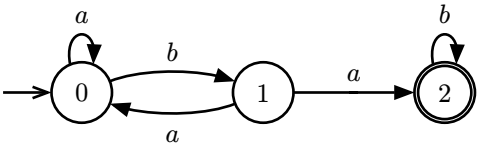
I – Automate non déterministe

Définition 1 : Automate non déterministe

Un *automate* est un 5-uplet $A = (\Sigma, Q, I, F, T)$ où :
- Σ est un alphabet.
- Q est un ensemble fini d'*états*.
- $I \subset Q$ est l'ensemble des *états initiaux*.
- $F \subset Q$ est l'ensemble des *états finaux*, ou acceptants.
- $T \subset Q \times \Sigma \times Q$ est l'ensemble des *transitions*.



$\Sigma = \{a, b\}, Q = \{0, 1, 2\}, I = \{0\}, F = \{2\}$
$T = \{(0, a, 0), (0, b, 1), (1, a, 0), (1, a, 2), (2, b, 2)\}$

Exercice 1

Combien y a t-il d'automates à Σ et Q fixés, en fonction de $n = |Q|$ et $p = |\Sigma|$? .....

Définition 2 : Fonction de transition δ

On peut remplacer T par $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ tel que :

$$\delta(q, a) = \{q' \in Q \mid (q, a, q') \in T\}$$

q	0	1	2
$\delta(q, a)$	.....	.....	.....
$\delta(q, b)$	.....	.....	.....

Il y a un *blocage* lorsque $\delta(q, a) = \varnothing$. $\delta(q, a)$ est aussi noté $q \cdot a$.

Exercice 2 : Implémentation

1. Rappeler la définition des structures de données abstraites de dictionnaire et d'ensemble.
2. Comment s'en servir pour représenter un automate ?
3. Proposer deux implémentations de ces structures de données.

Définition 3 : Fonction de transition étendue δ^*

On définit récursivement l'ensemble $\delta^*(q, m)$ des états atteignables par un chemin d'étiquette $m \in \Sigma^*$ depuis $q \in Q$:

$$\delta^*(q, \varepsilon) = \{q\} \quad \text{et} \quad \delta^*(q, am) = \bigcup_{q' \in \delta(q,a)} \delta^*(q', m)$$

Exercice 3

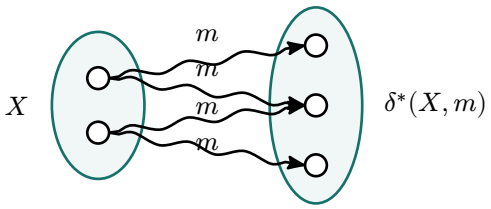
Donner $\delta^*(1, ab)$ pour l'automate ci-dessus.

.....

Définition 4 : Fonction de transition sur un ensemble

On peut aussi étendre δ et δ^* à un ensemble d'états $X \subset Q$:

$$\delta(X, a) = \bigcup_{q \in X} \delta(q, a) \quad \text{et} \quad \delta^*(X, m) = \bigcup_{q \in X} \delta^*(q, m)$$



Lemme 1 : Concaténation de chemins

Avec $u, v \in \Sigma^*$, $X \subset Q$ et en notant $\cdot$ au lieu de δ^* :

$$X \cdot uv = X \cdot u \cdot v$$

Cas particulier $X = \{q\}$: $q \cdot uv = q \cdot u \cdot v$

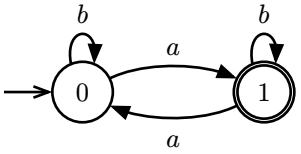
II – Langage reconnaissable

Définition 5 : Langage reconnaissable

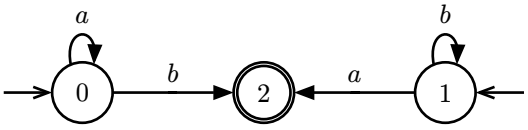
Un mot m est *accepté* par A si $\delta^*(I, m) \cap F \neq \varnothing$, c'est-à-dire s'il est étiquette d'un *chemin acceptant* de I à F.
Le *langage* $\mathcal{L}(A)$ *reconnu* par A est l'ensemble des mots acceptés par A.
On note $\mathcal{L}_{\text{rec}}(\Sigma)$ l'ensemble des *langages reconnaissables* par automate.

Exercice 4

Déterminer les langages reconnus par les automates ci-dessous.



$\mathcal{L}(A_1) =$ .....



$\mathcal{L}(A_2) =$ .....

Exercice 5

Montrer que les langages suivants sont reconnaissables :

1. $ab \mid abc \mid c$
2. Mots de taille paire sur $\Sigma = \{a, b\}$.
3. $(b \mid ab \mid aba)^*$
4. Mots sur $\Sigma = \{a, b\}$ dont l'avant-dernière lettre est a.

Exercice 6 : Test d'appartenance

Soit $A = (Q, \Sigma, I, F, T)$ un automate donné par liste d'adjacence et $m \in \Sigma^*$. Décrire des algorithmes pour tester si $m \in \mathcal{L}(A)$ et comparer leur complexité en temps et mémoire, en fonction de $n = |m|$, $k = |Q|$ et $p = |T|$.

III – Automate complet

Définition 6 : Automates équivalents

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

Définition 7 : Automate complet

Un automate $A = (Q, \Sigma, I, F, \delta)$ est *complet* s'il n'a pas de blocage : $\forall q \in Q, a \in \Sigma,\ \delta(q, a) \neq \varnothing$.

Théorème 1 : Complétion d'un automate

Tout automate est équivalent à un automate complet.

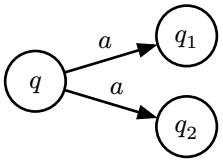
IV – Automate déterministe

Intuitivement, un automate est déterministe s'il n'y a jamais plusieurs possibilités de lecture.

Définition 8 : Automate déterministe (AD)

Un automate $A = (Q, \Sigma, I, F, \delta)$ est *déterministe* si :

- Il possède un unique état initial : $|I| = 1$
- Il y a au plus une transition $q \xrightarrow{a} \ldots : \forall q \in Q, a \in \Sigma,\ |\delta(q, a)| \leq 1$



Automate non déterministe

Un automate déterministe complet (ADC) a une fonction de transition de la forme $\delta : Q \times \Sigma \to Q$ et $\delta^* : Q \times \Sigma^* \to Q$.

Exercice 7 : Propriétés des automates déterministes

Soient $A = (Q, \Sigma, q_0, F, \delta)$ un AD, $q \in Q$, $m \in \Sigma^*$. Quelles sont les propositions vraies ?

1. $\delta(q_1, a) = \delta(q_2, a) \implies q_1 = q_2$.
2. A contient au plus un chemin d'étiquette m partant de q.
3. A contient au plus un chemin de q_0 à q.
4. A est *non ambigu* : chaque mot de $\mathcal{L}(A)$ a un unique chemin acceptant.
5. Si A' est un automate non ambigu alors A' est déterministe.

Exercice 8

Soit $A = (Q, \Sigma, q_0, F, \delta)$ un ADC à n états. Montrer : $\Sigma^{<n} \subset \mathcal{L}(A) \implies \mathcal{L}(A) = \Sigma^*$. La borne est-elle optimale ?

Exercice 9 : Automates à permutations

Soit $A = (Q, \Sigma, q_0, F, \delta)$ un ADC. Pour $a \in \Sigma$, on note $\delta_a(q) = \delta(q, a)$.
On suppose que A est un *automate à permutations*, c'est-à-dire que chaque δ_a est bijective.

1. Peut-on remplacer « bijective » par « injective » ou « surjective » dans cette définition ?
2. Soit $m \in \Sigma^*$. Montrer que $\delta^*(\cdot, m)$ est une permutation de Q.
3. En déduire : $\forall q \in Q, \exists k \geq 1,\ \delta^*(q, m^k) = q$.
4. Soit $a \in \Sigma$. $\Sigma^* a \Sigma^*$ est-il reconnaissable par un automate à permutations ?

Si A est un ADC et m un mot, on peut tester $m \in \mathcal{L}(A)$ en $O(|m|)$ en temps et $O(1)$ en mémoire :

```c
#include <stdbool.h>
typedef struct {
  int initial;
  bool *finaux; // finaux[q] : q est final
  int **delta; // delta[q][a] = q'
} adc;
```

```c
bool accepte(const adc *a, const int m[], int n) {
  int etat = a->initial;
  for (int i = 0; i < n; i++)
    etat = a->delta[etat][m[i]];
  return a->finaux[etat];
}
```

Théorème 2

♡ Soit $A = (\Sigma, Q, I, F, \delta)$ un automate. Alors A est équivalent à un ADC.

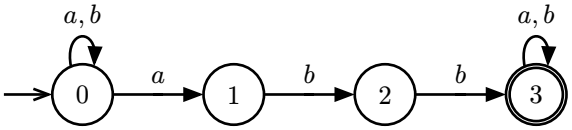
Soit $\mathcal{P}(A) = (\Sigma, \mathcal{P}(Q), \{I\}, F', \delta)$ l'*automate des parties*, où chaque état est un ensemble d'états de A,
$F' = \{X \subset Q \mid X \cap F \neq \varnothing\}$ et $\delta : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q)$ est définie comme dans Définition : Fonction de transition sur un ensemble.
♡ $\mathcal{P}(A)$ a un seul état initial I et chaque $\delta(X, a)$ est un unique ensemble d'états donc A' est un ADC.
$\mathcal{P}(A)$ est équivalent à A car :

$$m \in \mathcal{L}(\mathcal{P}(A)) \iff \delta^*(I, m) \in F' \iff \delta^*(I, m) \cap F \neq \varnothing \iff m \in \mathcal{L}(A)$$

L'automate des parties possède $2^{|Q|}$ états. En pratique, on ne construit que les états accessibles.

Exercice 10

♡ Déterminiser l'automate ci-contre.



Exercice 11

Soit $L_n = \Sigma^* a \Sigma^n$ sur $\Sigma = \{a, b\}$, avec $n \in \mathbb{N}$.

1. Montrer que L_n est reconnu par un automate non déterministe à $n + 2$ états.
2. Montrer que L_n est reconnu par un AD à 2^{n+2} états.
3. Montrer que tout AD reconnaissant L_n possède au moins 2^n états.

V – Accessibilité et émondage

Définition 9 : Accessibilité

Soit $A = (\Sigma, Q, I, F, \delta)$ un automate et $q \in Q$.

- q est *accessible* s'il existe un chemin de I à q dans A.
- q est *co-accessible* s'il existe un chemin de q à F dans A.
- A est *émondé* si tous ses états sont accessibles et co-accessibles.

Exercice 12

Décrire un algorithme déterminant les états accessibles et co-accessibles.

Théorème 3

Tout automate est équivalent à un automate émondé.

Exercice 13

Tout automate est-il équivalent à un ADC émondé ?

VI – Stabilité des langages reconnaissables

VI.1 – Complémentaire

Théorème 4 : Stabilité par complémentaire

♡ Si $L \in \mathcal{L}_{\text{rec}}(\Sigma)$ alors $\Sigma^* \setminus L \in \mathcal{L}_{\text{rec}}(\Sigma)$.

Exercice 14

1. Montrer que la preuve de Théorème : Stabilité par complémentaire nécessite un ADC.
2. Dessiner un automate reconnaissant les mots sur $\Sigma = \{a, b\}$ n'ayant pas aaa comme facteur.

VI.2 – Automate produit

♡ **Théorème 5 : Stabilité par intersection, union et différence**

$\mathcal{L}_{\text{rec}}(\Sigma)$ est stable par intersection, union et différence.

On considère deux ADC $A_k = (Q_k, \Sigma, i_k, F_k, \delta_k)$ de langage L_k, pour $k \in \{1, 2\}$.
Leur *automate produit* est $A_1 \times A_2 = (Q_1 \times Q_2, \Sigma, (i_1, i_2), F, \delta)$ avec $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$. Le produit simule les deux lectures en parallèle. Chaque opérateur correspond à un choix de F :

	$L_1 \cap L_2$	$L_1 \cup L_2$	$L_1 \setminus L_2$
F	.....	.....	.....

Exercice 15

1. Donner une construction plus simple pour reconnaître $L_1 \cup L_2$.
2. En déduire que $\mathcal{L}_{\text{rec}}(\Sigma)$ est stable par intersection et différence, sans utiliser l'automate produit.

Exercice 16

1. Montrer que $L = \{m \mid |m|_a \equiv 0 \mod 2,\ |m|_b \equiv 1 \mod 3\}$ est reconnaissable.
2. Est-il nécessaire de supposer que A_1 et A_2 sont des ADC dans la preuve de Théorème : Stabilité par intersection, union et différence ?

VII – Lemme de l'étoile

♡ **Lemme 2 : Lemme de l'étoile**

Soit L reconnu par un automate à n états. Pour tout $m \in L$ de taille au moins n, il existe des mots x, y, z tels que :

$$m = xyz, \quad |xy| \leq n, \quad y \neq \varepsilon, \quad xy^*z \subset L.$$

Attention :

- La réciproque du lemme de l'étoile est fausse.
- On peut choisir m mais pas sa décomposition xyz.

Méthode : Montrer que L n'est pas reconnaissable.
1. Supposer par l'absurde que L est reconnu par un automate à n états.
2. Choisir $m \in L$ avec $|m| \geq n$.
3. Appliquer le lemme de l'étoile pour obtenir une décomposition $m = xyz$.
4. Trouver k tel que $xy^kz \notin L$ pour aboutir à une contradiction.

Exercice 17

Montrer que les langages suivants ne sont pas reconnaissables :
1. $L_1 = \{a^n b^n \mid n \in \mathbb{N}\}$
2. $L_2 = \{a^n b^p \mid n, p \in \mathbb{N},\ n \neq p\}$

Remarques :

- Les automates sont sans mémoire : le choix d'une transition dépend uniquement de l'état actuel et de la lettre lue, pas du chemin déjà parcouru. Ils ne peuvent donc pas compter un nombre arbitraire, d'où $L_1, L_2 \notin \mathcal{L}_{\text{rec}}(\Sigma)$.
- Mais $\{a^n b^p \mid n \equiv p \bmod 2\} \in \mathcal{L}_{\text{rec}}(\Sigma)$ car il suffit de mémoriser la parité du nombre de a et de b lus.

HP VIII – Compléments

- Théorème de Myhill-Nerode : condition nécessaire et suffisante pour qu'un langage soit reconnaissable.
- Automate minimal : un automate A est équivalent à un unique ADC minimal $M(A)$. Les algorithmes de Moore et Hopcroft calculent une relation d'équivalence par raffinement de partition, Brzozowski calcule $M(A) = \mathcal{P}\left(\left(\mathcal{P}(A^\top)\right)^\top\right)$.
- Machine de Turing : modélise un ordinateur avec un automate et une mémoire infinie.