Résumé de Cours sur la théorie des automates

1 Automate fini déterministe

1.1 Définition d'un automate fini déterministe

Un automate fini déterministe est un quintuplet $A = (Q, \Sigma, \delta, q_0, F)$ où :

— Q : ensemble fini d'états

— Σ : alphabet (ou ensemble fini de symboles)

— $q_0 \in Q$: l'état initial de l'automate représenté par $\longrightarrow (q_0)$

— $F \subseteq Q$: les états terminaux de l'automate représentés par $((q))$ si $q \in F$

— $\delta : Q \times \Sigma \to Q$: fonction de transition de l'automate.

(q1,a) $\to q2$

q2 est l'état obtenu à partir de l'état q1 en acceptant le symbole a.

1.2 Automate complet

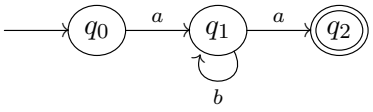
Un automate est complet si à partir d'un état quelconque, on a autant de transitions que de symboles de l'alphabet

1.3 Compléter un automate

Tout automate peut être complété par ajout d'un état piège.

1.4 Table de transition

Considérons Aut = ($\{q_0, q_1, q_2\}$, {a, b}, q_0, $\{q_2\}$)et δ définie par :



La fonction de transition peut se représenter par une table de transition :

	a	b
q_0	q_1	$-$
q_1	q_2	q_1
q_2	$-$	$-$

1.5 Extension de δ

— $\delta^* : Q \times \Sigma^* \mapsto Q$
— $\Delta : \mathcal{P}(Q) \times \Sigma \mapsto \mathcal{P}(Q)$
— $\Delta^* : \mathcal{P}(Q) \times \Sigma^* \mapsto \mathcal{P}(Q)$

1.6 Langage accepté par un automate fini déterministe

$L(A) = \{m \in \Sigma^* | \, \delta^*(q_0, m) \in F \, \}$

1.7 Langage régulier et automates

Théorème : Un langage est régulier si et seulement si il est reconnu par un automate fini déterministe

2 Automate fini non déterministe

2.1 Définition d'un automate fini non déterministe

Un automate fini non déterministe est un quintuplet $A = (Q, \Sigma, \delta, I, F)$ où :
— Q : ensemble fini d'états
— Σ : alphabet (ou ensemble fini de symboles)
— ϵ : symbole $\notin Q \cup \Sigma$ (ϵ transition ou transition arbitraire)
— $I \subseteq Q$: ensemble des états initiaux de l'automate.
— $F \subseteq Q$: ensemble des états terminaux de l'automate
— $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$ où $\mathcal{P}(Q)$ est l'ensemble des parties de Q

2.2 Trois formes de non déterminisme :

— $card(I) > 1$: plusieurs états initiaux
— $\exists q \in Q$, $\exists q' \in Q$ tel que $\delta(q, \epsilon) = q'$ (c'est-à-dire il existe des ϵ transitions)
— $\exists q \in Q$, $\exists \sigma \in \Sigma$ tel que $card(\delta(q, \sigma)) > 1$(c'est-à-dire à partir d'un état quelconque, et d'un symbole quelconque, il peut y avoir plusieurs états possibles)

2.3 Langage accepté par un automate fini non déterministe

$L(A) = \{m \in \Sigma^* | \, \delta^*(q_0, m) \in F \; et \; q_0 \in I\}$

3 Déterminisation

3.1 Théorème

Pour tout automate fini non déterministe, il existe un automate fini déterministe qui accepte le même langage.

3.2 ϵ-fermeture

Soit P un ensemble, on définit l' ϵ-fermeture (P) de la manière suivante :
ϵ-fermeture(P) = P $\cup$ { états accessibles à partir de chaque état de P en appliquant transitivement les ϵ-transitions}

3.3 Algorithme de déterminisation

1. Soit $A = (Q, \Sigma, I, F, \delta)$ un automate fini non déterministe avec $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$
2. Cherchons un automate A' déterministe équivalent à A. Soit $A' = (Q', \Sigma, q'_0, F', \delta')$ avec $\delta' : Q' \times \Sigma \to Q\prime$
   Pour cela :
   (a) déterminons l'état q'_0 :
   — $q'_0 = \epsilon\text{-}fermeture(Q_0)$
   (b) enfin déterminons la fonction de transition δ' :
   — calculons δ' (q'_0, σ) pour tout $\sigma \in \Sigma$. Par définition on a :
   δ' (q'_0, σ) = ϵ-fermeture(Δ (q'_0, σ))
   — on obtiendra ainsi de nouveaux états pour lesquels on calculera la fonction de transition et ainsi de suite ...

4 Langages réguliers et automates

4.1 Propriétés des langages réguliers

Il y a équivalence entre
— les langages réguliers,
— les langages décrits par une expression régulière,
— les langages reconnus par un automate fini déterministe,
— les langages reconnus par un automate fini non déterministe.

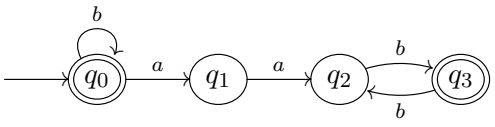
4.2 Trouver une expression régulière correspondant à un automate

4.2.1 Lemme d'Arden

L'équation $x = \alpha x \mid \beta$ a pour solution $x = \alpha^*\beta$
Le lemme d'Arden permet de transformer un automate en une expression régulière.

4.2.2 Système d'équations résolu avec le lemme d'Arden

1. principe
   — On associe à chaque q_i, une expression régulière e_i.
   — On crée un système d'équations qu'on résoud en utilisant le lemme d'Arden
2. exemple : soit l'automate suivant



— A chaque q_i on associe une expression régulière x_i. L'expression régulière recherchée est donc x_0.
— Lorsque un état q_i est terminal, l'expression régulière associée peut être le mot vide Λ
— On écrit donc un système d'équations :
  (a) $x_0 = ax_1 + bx_0 \mid \Lambda$
  (b) $x_1 = ax_2$
  (c) $x_2 = bx_3$
  (d) $x_3 = bx_2 + \Lambda$
— Il suffit de résoudre le système en utilisant le lemme d'Arden

4.3 Trouver un automate associé à une expression régulière

4.3.1 Dérivation d'expressions régulières

1. Définition
   Soit e une expression régulière, soit a un symbole de Σ, on définit la dérivée de e par rapport à a de la manière suivante :
   — $D_a(e) = \{\ \omega\ \ /\ \text{a}\ \omega \in \text{L(}\ \text{e}\ \text{)}\}$
   La méthodes des dérivées permet d'associer un automate à une expression régulière.
2. Propriétés des dérivées
   — $D_a(a) = \Lambda$
   — $D_a(b) = \emptyset$
   — $D_a(\emptyset) = \emptyset$
   — $D_a(\Lambda) = \emptyset$
   — $D_a(e_1 \mid e_2) = D_a(e_1) \mid D_a(e_2)$
   — $D_a(e_1^*) = D_a(e_1) \bullet e_1^*$
   — $D_a(e_1 \bullet e_2) = D_a(e_1) \bullet e_2 \mid D_a(e_2)\ si\ \Lambda \in L(e_1)$
   — $D_a(e_1 \bullet e_2) = D_a(e_1) \bullet e_2$ sinon

4.3.2 Algorithme calculant un automate à partir d'une expression régulière en utilisant les dérivées

— A l'expression régulière initiale e, on associe un état q_0 de l'automate.
— On calcule la dérivée de e par rapport à chaque symbole de l'alphabet, ce qui donne de nouvelles expressions régulières, donc de nouveaux états de l'automate. On répète ce processus pour chaque nouvelle expression régulière obtenue.
— Si pour un symbole a quelconque de l'alphabet, et pour une expression régulière e_i quelconque (associée à un état q_i) on obtient $D_a(e_i) = \emptyset$, cela signifie qu'il n'y a pas de transition à partir de q_i lors de la "lecture" de a.
— les états terminaux sont ceux qui correspondent à une expression régulière contenant le mot vide Λ

5 Opérateurs sur les automates finis

Pour décrire les opérations sur les automates, nous décrirons ceux-ci comme des quintuplets $\langle Q, X, I, F, \delta \rangle$ où $\delta : Q \times (X \cup \{\epsilon\}) \to \mathcal{P}(Q)$ est la **fonction** de transition.

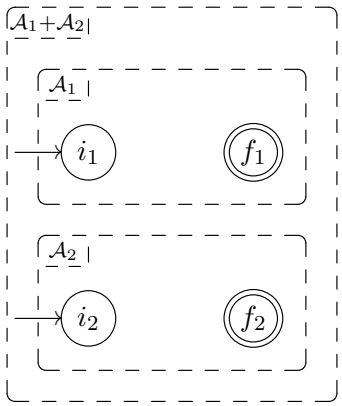
Dans ce qui suit, on supposera que $L(\mathcal{A}_i)$ est le langage reconnu par l'automate $\mathcal{A}_i = \langle Q_i, X, I_i, F_i, \delta_i \rangle$. On supposera (éventuellement après renommage) que les ensembles d'états Q_i sont disjoints.

5.1 Union

On cherche à construire un automate qui reconnaît le langage $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ à partir des automates $\mathcal{A}_1$ et $\mathcal{A}_2$. Cette construction correspond à l'opérateur $e_1 + e_2$ des expressions régulières.

Un mot de $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ est un mot de $L(\mathcal{A}_1)$ ou de $L(\mathcal{A}_2)$, i.e. un mot qui partant de I_1 peut amener $\mathcal{A}_1$ dans F_1 ou bien partant de I_2 peut amener $\mathcal{A}_2$ dans F_2. On propose donc l'automate suivant tel que $L(\mathcal{A}_1 + \mathcal{A}_2) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$:

$$\mathcal{A}_1 + \mathcal{A}_2 : \begin{cases} Q & = Q_1 \cup Q_2 \\ I & = I_1 \cup I_2 \\ F & = F_1 \cup F_2 \\ \delta(q,x) & = \begin{cases} \delta_1(q,x) & \text{si } q \in Q_1 \\ \delta_2(q,x) & \text{si } q \in Q_2 \end{cases} \end{cases}$$



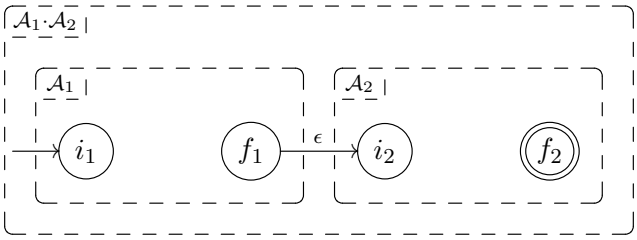
Note : $\mathcal{A}_1 + \mathcal{A}_2$ est le plus souvent indéterministe (plusieurs états initiaux).

5.2 Concaténation

On cherche à construire un automate qui reconnaît le langage $L(\mathcal{A}_1) \bullet L(\mathcal{A}_2)$ à partir des automates $\mathcal{A}_1$ et $\mathcal{A}_2$. Cette construction correspond à l'opérateur $e_1 \bullet e_2$ des expressions régulières.

Un mot de $L(\mathcal{A}_1) \bullet L(\mathcal{A}_2)$ est un mot de $L(\mathcal{A}_1)$ suivi d'un mot de $L(\mathcal{A}_2)$, i.e. un mot qui partant de I_1 peut amener $\mathcal{A}_1$ dans F_1 suivi d'un mot qui partant de I_2 peut amener $\mathcal{A}_2$ dans F_2. On propose donc l'automate suivant tel que $L(\mathcal{A}_1 \bullet \mathcal{A}_2) = L(\mathcal{A}_1) \bullet L(\mathcal{A}_2)$:

$$\mathcal{A}_1 \bullet \mathcal{A}_2 : \begin{cases} Q & = Q_1 \cup Q_2 \\ I & = I_1 \\ F & = F_2 \\ \delta(q,x) & = \begin{cases} \delta_1(q,x) \cup I_2 & \text{si } q \in F_1 \text{ et } x = \epsilon \\ \delta_1(q,x) & \text{si } q \in (Q_1 \setminus F_1) \text{ ou } x \neq \epsilon \\ \delta_2(q,x) & \text{si } q \in Q_2 \end{cases} \end{cases}$$



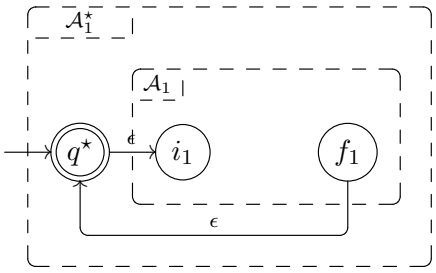
Note : $\mathcal{A}_1 \bullet \mathcal{A}_2$ est indéterministe (ϵ-transitions).

5.3 Étoile

On cherche à construire un automate qui reconnaît le langage $L(\mathcal{A}_1)^\star = \bigcup_{i \in \mathbb{N}} L(\mathcal{A}_1)^i = \{\Lambda\} \cup L(\mathcal{A}_1) \bullet L(\mathcal{A}_1)^\star$ à partir de l'automate $\mathcal{A}_1$. Cette construction correspond à l'opérateur $e_1^\star$ des expressions régulières.

Un mot de $L(\mathcal{A}_1)^\star$ est soit le mot vide, soit un mot de $L(\mathcal{A}_1)$ suivi d'un mot de $L(\mathcal{A}_1)^\star$. On a donc besoin d'un nouvel état initial $q^\star$ qui reconnaît Λ (et est donc également terminal) ou bien un mot de $L(\mathcal{A}_1) \bullet L(\mathcal{A}_1)^\star$. Un mot de $L(\mathcal{A}_1) \bullet L(\mathcal{A}_1)^\star$ est un mot qui partant de I_1 peut amener $\mathcal{A}_1$ dans F_1 suivi d'un mot de $L(\mathcal{A}_1)^\star$. On propose donc l'automate suivant tel que $L(\mathcal{A}_1^\star) = L(\mathcal{A}_1)^\star$:

$$\mathcal{A}_1^\star : \begin{cases} Q & = Q_1 \cup \{q^\star\} \\ I & = \{q^\star\} \\ T & = \{q^\star\} \\ \delta(q,x) & = \begin{cases} \delta_1(q,x) \cup \{q^\star\} & \text{si } q \in F_1 \text{ et } x = \epsilon \\ \delta_1(q,x) & \text{si } q \in (Q_1 \setminus F_1) \text{ ou } x \neq \epsilon \\ I_1 & \text{si } q = q^\star \text{ et } x = \epsilon \\ \emptyset & \text{si } q = q^\star \text{ et } x \neq \epsilon \end{cases} \end{cases}$$



Note : $\mathcal{A}_1^\star$ est indéterministe (ϵ-transitions).

5.4 Complémentaire

On cherche à construire un automate qui reconnaît le langage $\overline{L(\mathcal{A}_1)} = X^\star \setminus L(\mathcal{A}_1)$ à partir de l'automate $\mathcal{A}_1$.

Un mot de $\overline{L(\mathcal{A}_1)}$ est un mot qui partant de I_1 ne peut amener $\mathcal{A}_1$ dans F_1, i.e. un mot qui amène nécessairement $\mathcal{A}_1$ dans $Q_1 \setminus F_1$, quelles que soient les transitions choisies.

Pour simplifier la mise en oeuvre d'une telle définition, on supposera que $\mathcal{A}_1$ est **déterministe** et **complet** . On propose donc l'automate suivant tel que $L(\overline{\mathcal{A}_1}) = \overline{L(\mathcal{A}_1)}$:

$$\overline{\mathcal{A}_1} : \begin{cases} Q & = Q_1 \\ I & = I_1 \\ F & = Q_1 \setminus F_1 \\ \delta(q,x) & = \delta_1(q,x) \end{cases}$$

Note : $\overline{\mathcal{A}_1}$ est déterministe, si $\mathcal{A}_1$ est déterministe.

5.5 Miroir

On définit l'application miroir $\widetilde{\cdot}$ de $X^\star \to X^\star$ défini par : $\widetilde{\Lambda} \triangleq \Lambda$ et $\widetilde{m_1 \cdot m_2} \triangleq \widetilde{m_2} \cdot \widetilde{m_1}$. Cette application est naturellement étendue aux langages.

On cherche à construire un automate qui reconnaît le langage $\widetilde{L(\mathcal{A}_1)}$ à partir de l'automate $\mathcal{A}_1$.

Un mot de $\widetilde{L(\mathcal{A}_1)}$ est le miroir d'un mot qui partant de I_1 peut amener $\mathcal{A}_1$ dans F_1, i.e. un mot qui partant de F_1 peut amener dans I_1, si l'on prend les transitions de $\mathcal{A}_1$ àČÂă l'envers. On propose donc l'automate suivant tel que $L(\widetilde{\mathcal{A}_1}) = \widetilde{L(\mathcal{A}_1)}$:

$$\overline{\mathcal{A}_1} : \begin{cases} Q & = Q_1 \\ I & = F_1 \\ F & = I_1 \\ \delta(q,x) & = \{q' \in Q_1 \mid q \in \delta(q',x)\} \end{cases}$$

Note : $\widetilde{\mathcal{A}_1}$ peut être indéterministe.

5.6 Intersection

On cherche à construire un automate qui reconnaît le langage $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ à partir des automates $\mathcal{A}_1$ et $\mathcal{A}_2$.

Un mot de $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ est un mot de $L(\mathcal{A}_1)$ et de $L(\mathcal{A}_2)$, i.e. un mot qui partant simultanément de I_1 et de I_2 peut amener $\mathcal{A}_1$ (respectivement $\mathcal{A}_2$) dans F_1 et F_2 simultanément. On propose donc l'automate suivant tel que $L(\mathcal{A}_1 \times \mathcal{A}_2) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$:

$$\mathcal{A}_1 \times \mathcal{A}_2 : \begin{cases} Q & = Q_1 \times Q_2 \\ I & = I_1 \times I_2 \\ F & = F_1 \times F_2 \\ \delta(\langle q_1, q_2 \rangle, x) & = \{\langle q_1', q_2' \rangle \in Q_1 \times Q_2 \mid q_1' \in \delta_1(q_1,x), q_2' \in \delta_2(q_2,x)\} \end{cases}$$

Note : On peut également obtenir un automate similaire en utilisant la propriété $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) = \overline{\overline{L(\mathcal{A}_1)} \cup \overline{L(\mathcal{A}_2)}}$. Les automates $\mathcal{A}_1$ et $\mathcal{A}_2$ devront éventuellement avoir été complétés et déterminisés le cas échéant. La déterminisation nécessaire après l'opération d'union (pour calculer le dernier complémentaire) fait alors apparaître les couples d'états $\{q_1, q_2\}$ analogues aux paires (q_1, q_2) considérées dans notre proposition.