Chapitre 2 : Automates et langages réguliers

12 septembre

1 Mots

1.1 Alphabet et mots

Définition 1. *Un alphabet Σ est un ensemble fini. Les éléments d'un alphabet sont appelés lettres.*

Exemple 1.
1. *En informatique on utilise souvent des alphabets de taille 2 :* $\{0,1\}$ *ou* $\{a,b\}$.
2. *En français, l'alphabet courant (sans accentuation) comporte 26 caractères,*

$$\Sigma = \{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\}.$$

3. *Le code génétique est écrit sur l'alphabet des nucléotides* $\{A,T,G,C\}$ *pour l'ADN.*

Définition 2. *Un mot u sur un alphabet Σ est soit le mot vide ε, soit une suite finie $u_1u_2\dots u_n$ avec, pour tout $k \in [\![1,n]\!], u_k \in \Sigma$, c'est-à-dire, chaque u_k est un caractère de Σ.*

La longueur d'un mot u est l'entier, noté $|u|$, qui vaut 0 si $u = \varepsilon$ et n si $u = u_1u_2\dots u_n$ avec, pour tout $k \in [\![1,n]\!], u_k \in \Sigma$.

Définition 3. *L'ensemble des mots de longueur n sur Σ est noté Σ^n. L'ensemble de tous les mots sur Σ est noté Σ^*. Par construction,*

$$\Sigma^* = \bigcup_{n\in\mathbb{N}} \Sigma^n = \{\varepsilon\} \cup \bigcup_{n\in\mathbb{N}^*} \Sigma^n.$$

L'ensemble de tous les mots non vides sur Σ est noté Σ^+. Par construction,

$$\Sigma^+ = \bigcup_{n\in\mathbb{N}^*} \Sigma^n.$$

Exemple 2. *Si $\Sigma = \{a,b\}$ alors $\Sigma^3 =$*

1.2 Concaténation

Définition 4. *La concaténation de deux mots $u = u_1u_2\dots u_n$ et $v = v_1v_2\dots v_p$ est le mot uv de longueur $n+p$ dont le $i$eme caractère est u_i si $i \leq n$ et v_{i-n} sinon.*

Exemple 3. *La concaténation de $abba$ et de aab est*

Théorème 1. *La concaténation est une loi de composition interne associative sur Σ^* d'élément neutre ε.*

Remarques :

1. Grâce à l'associativité de la concaténation, on peut simplifier les écritures : on notera a^n pour le mot $aa\dots a$ où a apparaît n fois.
2. Il n'y a pas de symétrique pour la concaténation (sauf pour le mot vide), toutefois les éléments sont simplifiables à gauche comme à droite : $uv = uw$ ou $vu = wu$ entraîne $v = w$.
3. On a, par définition, l'additivité de la longueur pour la concaténation :
   Soit $u, v \in \Sigma^*$ alors $|uv| = |u| + |v|$.

1.3 Préfixes, suffixes, facteurs

Définition 5. *Soit $u, x \in \Sigma^*$. On dit que :*

- *x est un préfixe de u s'il existe un mot $y \in \Sigma^*$ tel que $u = xy$;*
- *x est un suffixe de u s'il existe un mot $y \in \Sigma^*$ tel que $u = yx$.*
- *x est un facteur de u s'il existe des mots $y, z \in \Sigma^*$ tels que $u = yxz$.*
- *Le mot x de longueur k est un sous-mot du mot $u = u_1 \dots u_n$, s'il existe des indices $1 \leq i_1 < i_2 < \dots < i_k \leq n$ tels que $x = u_{i_1} u_{i_2} \dots u_{i_k}$.*

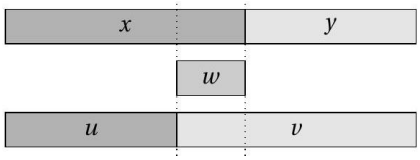
Exemple 4. *Les préfixes de MPI sont $\{\epsilon, M, MP, MPI\}$ et ses suffixes sont $\{\epsilon, I, PI, MPI\}$.*
Le mot $aoba$ est un facteur de $baobab$. Le mot bob est un sous mot de $baobab$ mais ce n'est pas un facteur.

1.4 Quelques résultats combinatoires

Théorème 2. *Soit x, y, u et $v \in \Sigma^*$ tels que $xy = uv$. Alors, il existe un unique mot w tel que l'une des conditions suivantes est réalisée*

1. *$x = uw$ et $v = wy$;*
2. *$u = xw$ et $y = wv$.*

La compréhension de ce résultat est assez immédiate avec le schéma suivant.



preuve

Corollaire 1. *Soit x, y deux préfixes (respectivement suffixes) d'un mot u. Alors, x est un préfixe (respectivement suffixe) de y ou y est un préfixe (respectivement suffixe) de x.*

On peut en déduire par exemple les résultats suivants :

Théorème 3. *Soient x, y et z dans Σ^* tels que $xy = yz$ et $x \neq \epsilon$. Alors il existe deux mots u et v et un entier $k \in \mathbb{N}$ tels que : $x = uv, y = (uv)^k u = u(vu)^k, z = vu$.*

preuve :

Théorème 4. *Soient x et y dans Σ^* tels que $xy = yx$, avec $x \neq \epsilon$ et $y \neq \epsilon$ alors il existe un mot $u \in \Sigma^*$ et deux entiers i et j tels que $x = u^i$ et $y = u^j$.*

preuve :

2 LANGAGES

Définition 6. *Un langage sur l'alphabet Σ est une partie de Σ^*.*

Exemple 5. *Voici quelques exemples de langages sur l'alphabet $\{a, b\}$*

- $\{a, ab, abba, ababbab\}$
- $\{a, b\}^*$
- *l'ensemble des mots se terminant par ba*
- *l'ensemble des mots de longueur paire*
- *l'ensemble des mots palindromes (égaux à leur symétrique miroir).*

Les langages étant des ensembles, ils sont susceptibles d'être soumis aux opérations ensemblistes : réunion, intersection, passage au complémentaire, différence symétrique. On ajoute à cette liste l'opération de concaténation et la puissance de langages.

Définition 7. *Soient L_1, L_2 deux langages sur l'alphabet Σ. La concaténation de L_1 et L_2 est le langage L_1L_2 défini par*

$$L_1L_2 = \{uv, u \in L_1, v \in L_2\}.$$

Exemple 6. *Si L_1 est le langage des mots comprenant uniquement la lettre a et si L_2 est le langage des mots comprenant uniquement la lettre b, alors L_1L_2 est le langage des mots s'écrivant comme une succession de a puis une succession de b.*

Soit L_1 l'ensemble des mots de longueur paire et L_2 l'ensemble des mots de longueur impaire. Que représente L_1L_2 ?

Définition 8. *Soit L un langage.*

Pour un entier $n \in \mathbb{N}$,on définit L^n par :

- $L^0 = \{\epsilon\}$
- $L^{n+1} = LL^n = L^nL$ *si* $n \in \mathbb{N}$.

On définit l'étoile de Kleene de L par $L^* = \bigcup_{n\in\mathbb{N}} L^n$ et $L^+ = \bigcup_{n\in\mathbb{N}^*} L^n$.

Attention : Il ne faut pas confondre L^2 et $\{u^2 : u \in L\}$. Déterminer ces deux ensembles quand $L = \{ab, ba\}$.

3 Langages réguliers et expressions régulières

Définition 9. *Soit Σ un alphabet. L'ensemble des langages réguliers sur Σ est défini inductivement de la façon suivante :*

- *$\emptyset$ est régulier.*
- *Pour tout $a \in \Sigma$, $\{a\}$ est régulier.*
- *Si L_1 et L_2 sont deux langages réguliers alors $L_1 \cup L_2$ ainsi que L_1L_2 sont aussi réguliers.*
- *Si L est régulier alors L^* l'est aussi.*

On note $\mathrm{Reg}(\Sigma)$, l'ensemble des langages réguliers sur Σ.

Exemple 7. *Montrer que $\{\epsilon\}$ est régulier.*

Exemple 8. *Montrer que tout ensemble fini est régulier.*

Définition 10. *Une expression régulière est définie inductivement par :*

- *$\emptyset$ et ϵ sont des expressions régulières.*
- *Pour tout $a \in \Sigma$, a est une expression régulière.*
- *Si E_1 et E_2 sont deux expressions régulières alors $E_1|E_2$ ainsi que E_1E_2 sont aussi régulières.*
- *Si E est une expression régulière alors E^* l'est aussi.*

Les expressions régulières permettent d'avoir une description des langages réguliers.

Définition 11. *Soit E une expression régulière sur un alphabet Σ, alors on définit inductivement le langage associé à E (noté $\mathcal{L}(E)$) de la manière suivante :*

- $\mathcal{L}(\emptyset) = \emptyset$.
- $\mathcal{L}(\epsilon) = \{\epsilon\}$.
- $\forall a \in \Sigma, \mathcal{L}(a) = \{a\}$.
- $\mathcal{L}(E_1|E_2) = \mathcal{L}(E_1) \cup \mathcal{L}(E_2)$ *et* $\mathcal{L}(E_1E_2) = \mathcal{L}(E_1)\mathcal{L}(E_2)$.
- $\mathcal{L}(E^*) = \mathcal{L}(E)^*$

Exemple 9. *Quel est le langage associé à l'expression :* $0^*10^*10^*10^*$ *?*

Donner une expression régulière correspondant à l'écriture en base deux d'un entier sans 0 non significatif.

4 Automates finis déterministes

4.1 Définition et premiers exemples

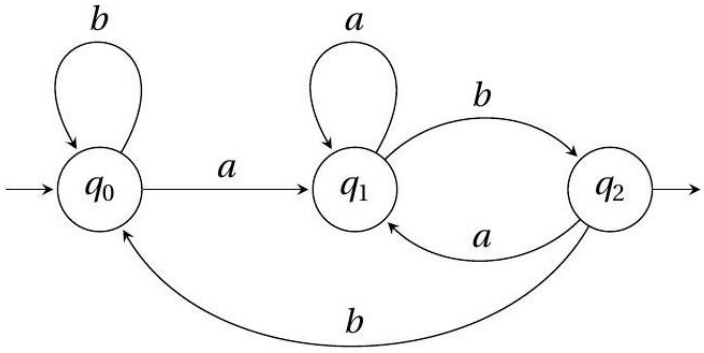
Définition 12. *Un automate fini déterministe (ou AFD) sur un alphabet Σ est un quadruplet (Q, q_0, F, δ) composé de :*

- *Q, un ensemble fini dont les éléments sont appelés états de l'automate ;*
- *$q_0 \in Q$, appelé état initial ;*
- *$F \subset Q$, appelé ensemble des états finaux (ou acceptants) ;*
- *δ une application d'une partie de $Q \times \Sigma$ dans Q, appelée fonction de transition.*

Considérons l'automate $(\{q_0, q_1, q_2\}, q_0, \{q_2\}, \delta)$ sur $\Sigma = \{a, b\}$, où δ est définie par

	q_0	q_1	q_2
a	q_1	q_1	q_1
b	q_0	q_2	q_0

On le représente de la façon suivante.



Remarquons la façon de signaler l'état initial et les états acceptants avec une flèche (entrante ou sortante). Il existe de nombreuses conventions distinctes : par exemple, certains auteurs représentent les états acceptants avec un double cercle plutôt qu'avec une flèche.

Définition 13. *Soit un automate (Q, q_0, F, δ) sur l'alphabet Σ. La fonction de transition δ^* étendue aux mots est la fonction partielle $\delta^* : Q \times \Sigma^* \to Q$ définie récursivement par*

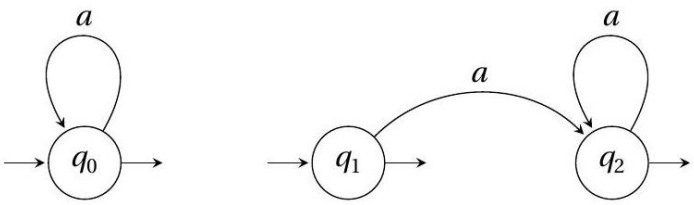
- *pour tout $q \in Q, \delta^*(q, \varepsilon) = q$,*
- *pour tout $q \in Q, u \in \Sigma^*$ et $a \in \Sigma$, $\delta^*(q, ua) = \delta\left(\delta^*(q, u), a\right)$*

Ainsi, L'état $\delta^*(q, u)$ est l'unique état (s'il n'y a pas blocage) où l'on aboutit en lisant le mot u depuis l'état q.

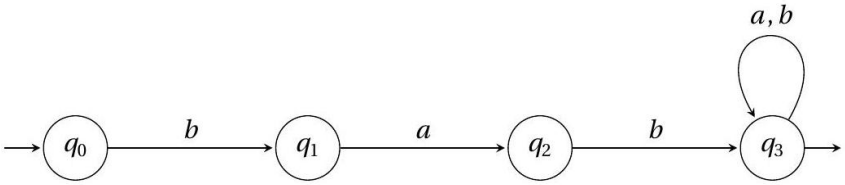
Définition 14. *Soit un automate $A = (Q, q_0, F, \delta)$ sur l'alphabet Σ et δ^* la fonction de transition étendue.*

1. *Un mot $u \in \Sigma^*$ est dit reconnu par A si $\delta^*(q_0, u) \in F$.*
2. *Le langage reconnu par A, noté $\mathcal{L}(A)$, est l'ensemble des mots reconnus par A.*
3. *Un langage L est dit reconnaissable s'il existe un AFD A tel que $L = \mathcal{L}(A)$. On note $\mathrm{Rec}(\Sigma)$ l'ensemble des langages reconnaissables sur Σ.*

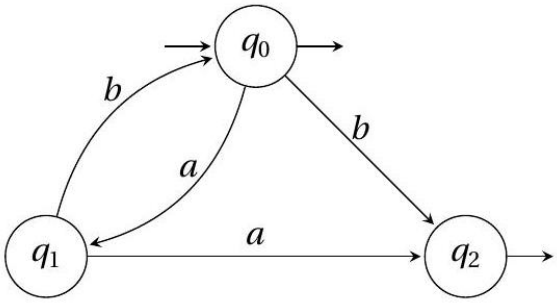
Exemple 10. *Les deux automates suivants sur l'alphabet $\Sigma = \{a\}$ reconnaissent le langage Σ^*.*



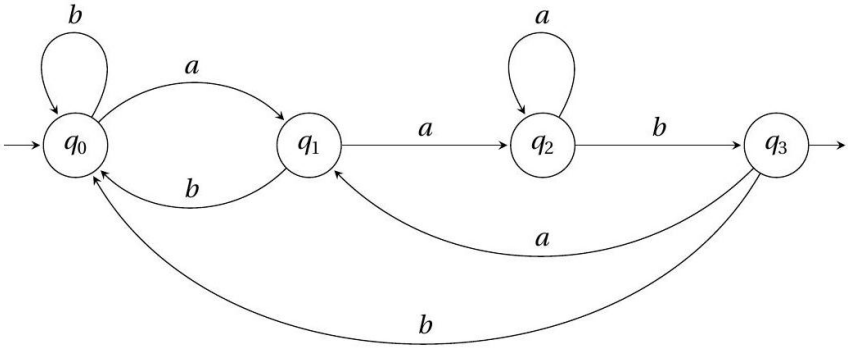
Exemple 11. *L'automate suivant reconnaît le langage $\{bab\}\Sigma^*$, c'est-à-dire l'ensemble des mots commençant par bab.*



Exemple 12. *L'automate suivant reconnaît le langage* $\{ab\}^*\{aa, b, \varepsilon\}$.



Exemple 13. *Quel est le langage reconnu par l'automate suivant ?*



Exemple 14. *Construire un automate qui reconnait le langage des mots de longueur paire sur l'alphabet* $\{a, b\}$.

Exemple 15. *Construire un automate qui reconnait le langage des mots sur l'alphabet* $\{a, b\}$ *où deux lettres identiques ne se suivent jamais :* $ababa$ *est dans le langage mais pas* $aabab$.

Exemple 16. *Construire un automate qui reconnait le langage des mots sur l'alphabet* $\{a, b\}$ *où une occurrence d'une même lettre ne peut pas apparaitre plus de deux fois de suite.*

Attention : Tous les langages ne sont pas reconnaissables : en effet, l'ensemble des automates est dénombrable alors que l'ensemble des langages ne l'est pas.

En effet, Σ^* est un ensemble infini, donc $\mathcal{P}(\Sigma^*)$, qui est l'ensemble des langages est non dénombrable.

D'autre part, pour chaque $n \in \mathbb{N}^*$, il n'y a qu'un nombre fini de façons de construire un automate dont les états sont inclus dans $\{0, \dots n-1\}$.

4.2 Implémentation

Définissons un type automate avec un enregistrement (l'ensemble des états est implicitement celui des entiers).

```
type automate ={finaux: bool array; delta: int -> char -> int};
```

où l'état initial est toujours l'état 0, `finaux.(i)` vaut `true` si, et seulement si, l'état `i` est final. `delta` est une fonction pas nécessairement définie pour tous les états et lettres.

L'exception `Blocage` sert alors à pallier le problème de la définition de la fonction de transition.

```
exception Blocage;;
```

Écrivons désormais le calcul de $\delta^*(q_0, m)$ pour un mot m.

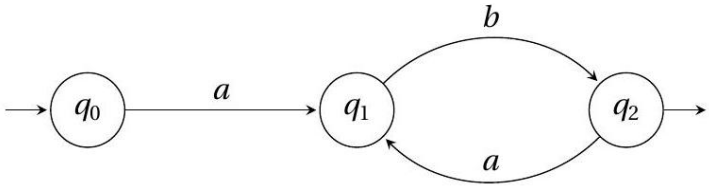
Il ne reste plus qu'à tester l'appartenance à la liste des états finaux.

4.3 Propriétés spécifiques auxquelles on peut se ramener

1. Complétion

   Définition 15. *Un blocage de l'automate (Q, q_0, F, δ) sur l'alphabet Σ est un couple $(q, a) \in Q \times \Sigma$ pour lequel la fonction de transition n'est pas définie. Un automate sans blocage est dit complet.*

   Exemple 17. *L'automate suivant n'est pas complet.*

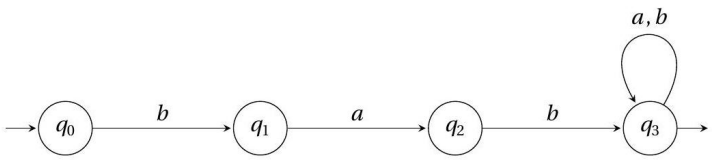
   En effet, il y a trois blocages : $(q_0, b), (q_1, a)$ *et* (q_2, b).

   Théorème 5. *Tout langage reconnaissable est reconnu par un automate complet.*

   idée -> On ajoute un état puit et tous les blocages deviennent des transitions vers cet état.

Formellement :

Exemple 18. *Avec ce résultat, on passe de l'automate incomplet*



à l'automate complet

2. Emondage

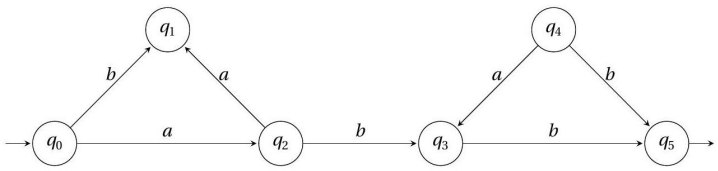
Définition 16. *Soit un automate (Q, q_0, F, δ) sur l'alphabet Σ et δ^* la fonction de transition étendue.*
— *Un état $q \in Q$ est dit accessible s'il existe un mot $u \in \Sigma^*$ tel que $\delta^*(q_0, u) = q$.*
— *Un état $q \in Q$ est dit co-accessible s'il existe un mot $u \in \Sigma^*$ tel que $\delta^*(q, u) \in F$.*
— *Un état $q \in Q$ est dit utile s'il est accessible et co-accessible.*
— *L'automate est dit émondé si tous ses états sont utiles.*

Remarque : Si un automate reconnaît un langage non vide, alors il existe au moins un état accessible et co-accessible ; et on peut trouver un état final accessible.

Théorème 6. *Tout langage reconnaissable non vide est reconnu par un automate émondé.*

preuve : idée -> On efface les états qui ne sont pas à la fois accessibles et co-accessibles et toutes les transitions impliquant ces états.

Exemple 19. *Pour l'automate*



on obtient

5 Automates finis non déterministes

Nous allons maintenant considérer des automates dans lesquels à la lecture d'un mot, il y a plusieurs manières possibles de lire un tel mot. Un mot sera accepté ssi l'une des manières de le lire permet de passer de l'état initial à un état final. Prenons un exemple simple : considérons le langage des mots terminant par a :

5.1 Non-déterminisme

On va donc s'autoriser à avoir une fonction de transition qui, à la lecture d'une lettre sur un état donné, permet d'accéder à plusieurs états différents. On relâche donc l'hypothèse de lecture unique d'un mot dans le graphe de l'automate. Ainsi, on relâchera aussi le fait d'avoir un seul état initial : à la lecture du mot vide ϵ, on peut aussi se retrouver dans plusieurs états différents.

On donne donc la définition d'un automate fini non déterministe (afdn) de la manière suivante :

Définition 17. *Un automate fini non déterministe (ou AFND) sur l'alphabet Σ est un quadruplet (Q, I, F, δ) composé de*

- *Q, un ensemble fini dont les éléments sont appelés états de l'automate*
- *$I \subset Q$, appelé* ***ensemble des états initiaux***
- *$F \subset Q$, appelé ensemble des états finaux (ou acceptants)*
- *δ une application de $Q \times \Sigma$ dans $\mathcal{P}(Q)$, appelée fonction de transition.*

Il convient de définir la notion de fonction de transition étendue dans ce contexte : plutôt que d'avoir comme image un état, la fonction aura pour image un ensemble d'états.

Définition 18. *Soit un automate non déterministe (Q, I, F, δ) sur l'alphabet Σ. La fonction de transition δ^* étendue aux mots est la fonction $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$ définie récursivement par :*

- *pour tout $q \in Q, \delta^*(q, \epsilon) = \{q\}$.*
- *pour tout $q \in Q, x = ua \in \Sigma^*, \delta^*(q, x) = \bigcup_{q' \in \delta^*(q,u)} \delta(q', a)$.*

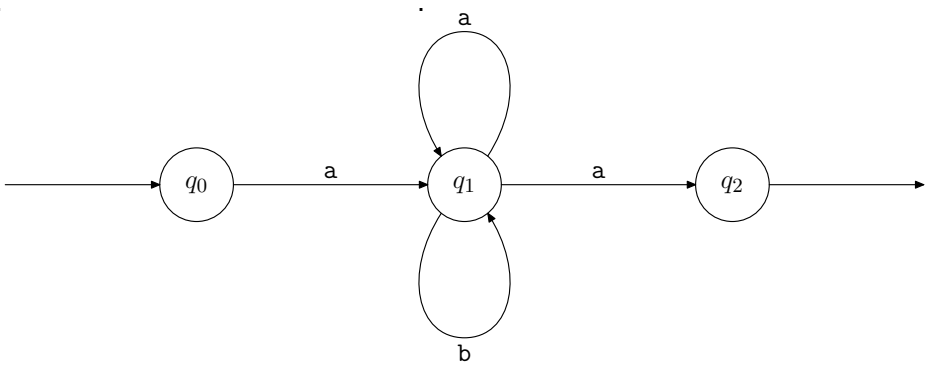
Définition 19. *Soit un AFND $A = (Q, I, F, \delta)$ sur l'alphabet Σ et δ^* la fonction de transition étendue.*

- *Un mot $u \in \Sigma^*$ est dit reconnu par A s'il existe $q_0 \in I$ tel que $\delta^*(q_0, u) \cap F \neq \emptyset$.*
- *Le langage reconnu par A, noté $\mathcal{L}(A)$, est l'ensemble des mots reconnus par A.*

Moins formellement, u est accepté par l'automate si parmi tous les chemins possibles lors de la lecture de u, il en existe un d'un état initial vers un état final.

Remarque 1. *On peut aussi utiliser la notion de relation plutôt que la notion de fonction dans l'ensemble des parties des états : un AFND peut donc être la donnée de (Q, I, F, R) où R est une relation sur $Q \times \Sigma \times Q$. Ainsi, un mot $u = u_1 \dots u_n$ est accepté ssi il existe une suite d'états de l'automates $q_0, \dots, q_n$ telle que $q_0 \in I, q_n \in F$ et $\forall i \in \{1, \dots, n\}, R(q_{i-1}, u_i, q_i)$.*

Exemple 20. *Quel est le langage reconnu par l'automate suivant :*



Exemple 21. *Donner un automate non déterministe qui reconnait le langage des mots terminant par* $abab$.

5.2 Pourquoi deux modèles ?

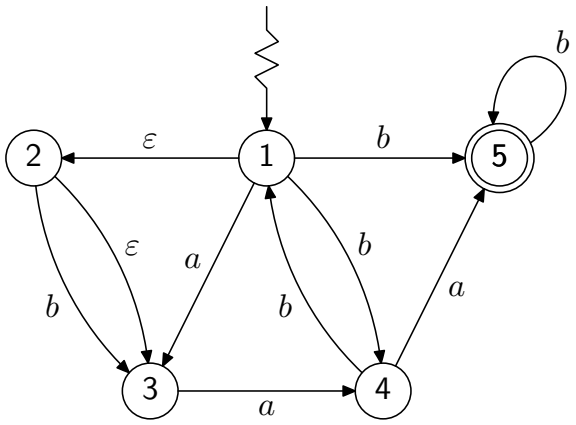
Un avantage des automates finis non déterministes sur les automates finis déterministes est leur simplicité de manipulation : il est beaucoup plus facile de concevoir un automate non déterministe reconnaissant un langage donné mais il est aussi plus facile de les manipuler dans des démonstrations formelles :

1. Soit $u = u_1 \dots u_n$ un mot donné. Donner un AFND qui reconnait le langage des mots dont un facteur est u. Même question avec u un sous-mot.

2. Soit A_1 un AFND qui reconnait un langage L_1 et soit A_2 un AFND qui reconnait un langage L_2. Trouver un automate A qui reconnait $L_1 L_2$.

5.3 ϵ-transitions

Pour rajouter au non déterministe, on peut même étendre la définition d'automate non déterministe en autorisant l'utilisation d'ϵ-transitions, aussi appelées transitions instantannées qui permettent de passer d'un état à un autre sans lire de lettre.



Exemple 22.
baa et aa sont par exemple reconnus par cet automate.

Définition 20. *Soit Σ un alphabet fini. Un automate fini non déterministe est un quintuplet $(Q, I, F, \delta, \varphi)$ où*
- *Q, un ensemble fini dont les éléments sont appelés états de l'automate*
- *$I \subset Q$, appelé* ***ensemble des états initiaux***
- *$F \subset Q$, appelé ensemble des états finaux (ou acceptants)*
- *δ une application de $Q \times \Sigma$ dans $\mathcal{P}(Q)$, appelée fonction de transition.*
- *φ est une application de Q dans $\mathcal{P}(Q)$ qui associe à chaque état l'ensemble des états auxquels il peut accéder par une ϵ-transition.*

Définition 21. *Soit Σ un alphabet fini. Soit A un automate fini non déterministe défini par le quintuplet $(Q, I, F, \delta, \varphi)$ et soit $Q' \subset Q$. On définit la fermeture transitive de Q' par ϵ-transition, notée $\kappa(Q)$ par la plus petite partie de Q contenant Q' et stable par ϵ-transition c'est-à dire une partie P vérifiant :*

$$\bigcup_{q' \in P} \varphi(q') \subset P$$

Théorème 7. *Soit Σ un alphabet fini. Soit A un automate fini non déterministe défini par le quintuplet $(Q, I, F, \delta, \varphi)$ et soit $Q' \subset Q$. La suite d'ensembles définie par $K_0 = Q'$ et $\forall n \in \mathbb{N}, K_{n+1} = K_n \cup (\bigcup_{q' \in K_n} \varphi(q'))$ est stationnaire de limite $\kappa(Q')$.*

preuve :

Intuitivement, $\kappa(Q')$ désigne l'ensemble de tous les états (dont ceux de Q') qui peuvent être atteints à partir de Q' par un nombre fini quelconque d ϵ-transitions.

Les définitions de calculs et de langage reconnus s'adaptent alors facilement :

Définition 22. *Soit un automate non déterministe $(Q, I, F, \delta, \varphi)$ sur l'alphabet Σ. La fonction de transition δ^* étendue aux mots est la fonction $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$ définie récursivement par :*
- *pour tout $q \in Q, \delta^*(q, \epsilon) = \kappa(q)$.*
- *pour tout $q \in Q, x = ua \in \Sigma^*, \delta^*(q, x) = \kappa(\bigcup_{q' \in \delta^*(q,u)} \delta(q', a))$.*

Définition 23. *Soit un AFND $A = (Q, I, F, \delta, \varphi)$ sur l'alphabet Σ et δ^* la fonction de transition étendue.*
- *Un mot $u \in \Sigma^*$ est dit reconnu par A s'il existe $q_0 \in I$ tel que $\delta^*(q_0, u) \cap F \neq \emptyset$.*
- *Le langage reconnu par A, noté $\mathcal{L}(A)$, est l'ensemble des mots reconnus par A.*

Il est évident que l'ensemble des langages reconnus par AFD est inclus dans celui reconnus par AFND. Qu'en est-il de la réciproque ? Existe-t-il des langages reconnus par un AFND pour lesquels on ne peut pas construire d'AFD ?

5.4 Déterminisation

Théorème 8. *Soit A un automate non déterministe. Il existe A' un automate déterministe qui reconnait le même langage.*

C'est la construction de l'automate des parties.

Prenons un exemple : déterminisons l'exemple 20.

Formellement :

Déterminisons les exemples 21 et 22 :

Complexité de la déterminisation :

Théorème 9. *Soit $n \geq 1$ un entier naturel. Il existe un afnd à $n+1$ états, tel que tout afd qui reconnaît le même langage ait au moins 2^n états. La déterminisation est donc un algorithme de coût exponentiel.*

On dira de deux automates qu'ils sont équivalents s'ils reconnaissent le même langage. On a vu que tout automate non déterministe (sans ϵ-transitions) est équivalent à

* un automate déterministe ;
* un automate déterministe complet (sans blocage) ;
* un automate déterministe émondé.
* un automate déterministe standardisé.

Notons $\mathrm{Rec}(\Sigma)$ l'ensemble des langages reconnaissables.

6 Opérations sur les automates

A partir de deux automates $\mathcal{A}_1$ et $\mathcal{A}_2$ reconnaissant respectivement des langages L_1 et $L2$, on peut construire un automate :

1. $\mathcal{A}$ reconnaissant $L_1 \cup L_2$.
2. $\mathcal{A}$ reconnaissant $L_1 \cap L_2$.
3. $\mathcal{A}$ reconnaissant $L_1 L_2$.
4. $\mathcal{A}$ reconnaissant L^*.