Aspects algorithmiques et combinatoires de la théorie des automates

Partie I: fondements de la théorie des automates

Cyril Nicaud, LIGM, Paris-Est

8 juillet 2016

Dans ce chapitre, on redonne les définitions classiques de la théorie des automates finis et des langages rationnels, ainsi que quelques constructions fondamentales.

1 Définitions et propriétés fondamentales

1.1 Mots

Pour tout ce cours, A est un ensemble fini non vide appelé *alphabet*, dont les éléments sont appelés des *lettres*. On prendra souvent $A = \{a, b\}$, ou $A = \{0, 1\}$, ou encore $A = \{a_1, a_2, \ldots, a_k\}$. D'une façon générale, on utilisera la lettre k pour le cardinal de A, c'est-à-dire le nombre de lettres considérées.

Un *mot de longueur* ℓ est un élément de A^ℓ. On peut avoir $\ell = 0$, avec la convention que $A^0 = \{\varepsilon\}$, où ε est le *mot vide*, l'unique mot de longueur 0. L'ensemble de tous les mots [1] est noté A^*, et est défini par $A^* = \cup_{\ell \geq 0} A^\ell$. Comme on veut voir ces objets comme des mots plutôt que de ℓ-uplets, on écrira successivement tous les éléments sans utiliser les parenthèses et les virgules : (a, b, b, a, a) sera simplement écrit *abbaa*.

Si $u \in A^*$, on notera $|u|$ sa longueur. Si $u = u_1 \cdots u_m$ et $v = v_1 \cdots v_n$ sont deux mots de longueurs respectives m et n (les u_i et les v_i sont des lettres de A), on notera uv (ou encore $u \cdot v$) la *concaténation de u et de v*, qui est le mot de longueur $m + n$ défini par

$$uv = u_1 \cdots u_m v_1 \cdots v_m.$$

La concaténation est l'opération essentielle de cette théorie. Elle est associative et possède l'élément neutre ε. Elle n'est pas commutative. Nous introduirons les autres opérations au fur et à mesure, en fonction des besoins.

Un mot u est *préfixe* (resp. *suffixe*) d'un mot v quand il existe un mot w tel que $v = uw$ (resp. $v = wu$). Un mot u est *facteur* d'un mot v quand il existe deux mots w et z tels que $v = wuz$. On évitera d'utiliser le terme de sous-mot qui n'a pas toujours la même signification dans la littérature.

On suppose l'alphabet totalement ordonné. On munit alors l'ensemble des mots de l'*ordre lexicographique* $\leq_{\text{lex}}$ défini par

$$u \leq_{\text{lex}} v \Leftrightarrow \begin{cases} u \text{ préfixe de } v, \\ \text{ou} \\ w\alpha \text{ préfixe de } u \text{ et } w\beta \text{ préfixe de } v, \text{ avec } \alpha, \beta \in A \text{ et } \alpha < \beta. \end{cases}$$

1. Attention, l'étoile ici n'a pas le même sens que quand on écrit $\mathbb{R}^*$ pour l'ensemble de réels non nuls. Elle correspond à l'étoile de Kleene définie un peu plus loin.

Comme l'ordre n'est pas bien fondé (la suite $a^n b$ est strictement décroissante), il est parfois utile de considérer l'*ordre militaire* $\leq_{\text{mil}}$ défini par

$$u \leq_{\text{mil}} v \Leftrightarrow |u| < |v| \text{ ou } (|u| = |v| \text{ et } u \leq_{\text{lex}} v)\,.$$

1.2 Langages

Un *langage* est un ensemble de mots. L'ensemble des langages est donc naturellement muni des opérations ensemblistes (union, intersection, complémentaire, ...). On définit la *concaténation de deux langages* L et K comme une extension naturelle de la concaténation des mots :

$$L \cdot K = \{u \cdot v \mid u \in L,\ v \in K\}.$$

On définit également la puissance d'un langage L par $L^0 = \{\varepsilon\}$ et $L^{n+1} = L \cdot L^n$, pour tout $n \in \mathbb{N}$.

L'*étoile de Kleene* d'un langage L, ou plus simplement l'*étoile* de L, est le langage L^* défini par

$$L^* = \bigcup_{n \geq 0} L^n.$$

Cette opération est particulièrement importante car c'est elle va nous servir, dans le cadre des langages rationnels, à construire des langages infinis.

1.3 Langages rationnels, expressions rationnelles

L'ensemble des langages rationnels sur l'alphabet A est l'ensemble de langages $\text{Rat}(A^*)$ défini inductivement par :

— $\emptyset$, $\{\varepsilon\}$ et $\{a\}$ sont dans $\text{Rat}(A^*)$, pour toute lettre $a \in A$;

— $\text{Rat}(A^*)$ est stable par union, concaténation des langage et étoile de Kleene.

Ainsi $(\{a\} \cup \{b\} \cdot \{a\})^*$ est un langage rationnel. Pour simplifier les notations, on s'autorise à écrire ce genre de formule en ne mettant pas les accolades, en supprimant les symboles de concaténation, et en général en remplaçant le symbole $\cup$ par le symbole $+$: sur notre exemple on écrira plutôt $(a+ba)^*$. Formellement, on peut définir ce genre de formules, qui sont appelées les *expressions rationnelles*, qu'on évalue en un langage rationnel en interprétrant la "formule" dans le monde des ensembles de mots. C'est la même histoire que de différencier une formule logique de la fonction booléenne qu'elle définit : $a \Rightarrow b$ et $\neg a \vee b$ sont deux formules différentes pour la même fonction booléene. De même $(a + b)^*$ et $(a^*b)^*a^*$ sont deux expressions rationnelles qui définissent le même langage rationnel (la vérification de cette affirmation est laissée en exercice).

1.4 Automates finis : automates déterministes

Avec un peu d'habitude, les expressions rationnelles sont un moyen relativement agréable pour définir des langages. On se convainc facilement par exemple que $A^*baabbA^*$ est l'ensemble des mots qui contiennent *baabb* comme facteur (on utilise ici le raccourci A^* pour $(a_1+\ldots+a_k)^*$, où les a_i sont les lettres de a).

En revanche, le traitement algorithmique des expressions rationnelles n'est en général pas facile : tester si un mot est dans le langage, calculer une représentation du complémentaire (on verra plus loin que le complémentaire d'un langage rationnel est encore rationnel), ...

Les automates finis, introduits dans cette partie, vont nous permettre de représenter les langages rationnelles sous une forme plus adaptée aux algorithmes. Ce sont des machine (théoriques) très simples, qui caractérisent des langages.

Formellement, un *automate déterministe et complet* est un quintuplet (A, Q, δ, q_0, F) où

— A est l'alphabet sur lequel on travaille ;
— Q est un ensemble fini dont les éléments sont appelés des *états* ;
— δ est la *fonction de transition*. C'est une application [2] de $Q \times A$ dans Q. On peut la voir aussi comme une collection d'applications $\delta(\cdot, a)$ de $Q \to Q$, une pour chaque lettre $a \in A$. On notera $\delta_a := \delta(\cdot, a)$;
— $q_0 \in Q$ est un état distingué, qu'on appelle l'*état initial* ;
— $F \subseteq Q$ est l'ensemble des *états terminaux*.

La bonne façon de voir ces objets c'est leur représentation graphique : ce sont des graphes orientés, dont les sommets sont les états, et dont les arcs sont étiquetés par des lettres de a. On a un arc $p \xrightarrow{a} q$ si et seulement si $\delta(p, a) = q$. En théorie des automates, ces arcs sont appelés des *transitions*. On distingue l'état initial avec une flêche rentrante et les états terminaux par des doubles-cercles.

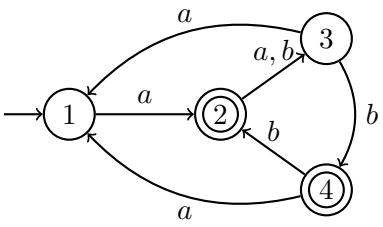
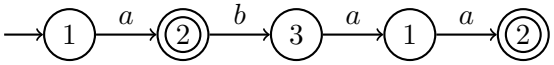


FIGURE 1 – Exemple d'automate déterministe et complet. Cet automate possède 4 états. Son état initial est l'état 1. Ses états terminaux sont les états 2 et 4. On a, par exemple $\delta(2, b) = 3$, car la transition $2 \xrightarrow{a,b} 3$ sert à indiquer qu'on a à la fois $2 \xrightarrow{a} 3$ et $2 \xrightarrow{b} 3$.

Un *chemin* dans un automate se définit comme dans un graphe : c'est une séquence de transitions $(p_i \xrightarrow{a_i} q_i)_{i \in \{1,\dots,\ell\}}$ telle que pour tout $i \in \{1, \dots, \ell - 1\}$, $q_i = p_{i+1}$: on repart toujours de l'état où on est arrivé le coup précédent. L'*étiquette* du chemin est le mot $a_1 \cdots a_\ell$.

Un chemin $(p_i \xrightarrow{a_i} q_i)_{i \in \{1,\dots,\ell\}}$ est dit *acceptant* quand p_1 est l'état initial et q_ℓ est un état terminal. Pour un automate déterministe et complet $\mathcal{A}$ on note $\mathcal{L}(\mathcal{A})$ l'ensemble de tous les mots qui étiquettent un chemin acceptant dans $\mathcal{A}$: c'est le *langage reconnu par* $\mathcal{A}$.

Si on reprend l'exemple de la figure 1, le chemin suivant est acceptant :



Et donc le mot $u = abaa$ étiquette un chemin acceptant, il est reconnu par l'automate : $u \in \mathcal{L}(\mathcal{A})$.

Un *automate déterministe* $\mathcal{A} = (A, Q, \delta, q_0, F)$ se définit comme un automate déterministe et complet, sauf que la fonction de transition δ n'est pas nécessairement une application : il se peut que $\delta(q, a)$ ne soit pas défini. La notion d'acceptation est la même : un mot est reconnu quand il étiquette un chemin de l'état initial à un état terminal.

1.5 Automates finis : automates non-déterministes

Il existe une forme plus générale d'automates, appelés les *automates non-déterministes*. De tels automates peuvent avoir plusieurs états initiaux et il peut y avoir plusieurs transitions étiquetées par la même lettre qui partent d'un même état.

On note $\mathcal{P}(E)$ l'ensemble de tous les sous-ensembles de l'ensemble E (c'est parfois noté 2^E dans la littérature).

2. On rappelle que ce qui distingue une fonction d'une application, c'est que l'application est définie sur tout l'ensemble de départ.

Formellement, un *automate non-déterministe* est un quintuplet (A, Q, δ, I, F) où
— A est l'alphabet ;
— Q est un ensemble d'états ;
— δ est une application de $Q \times A$ dans $\mathcal{P}(Q)$;
— $I \subseteq Q$ est l'ensemble des état initaux ;
— $F \subseteq Q$ est l'ensemble des états terminaux.

On représente également les automates non-déterministes par des graphes, mais cette fois il y a une transition $p \xrightarrow{a} q$ pour chaque état $q \in \delta(p, a)$.

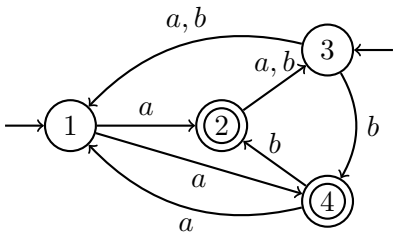


FIGURE 2 – Un automate non-déterministe avec deux états initiaux (1 et 3). On a, par exemple, $\delta(3, b) = \{1, 4\}$ et $\delta(1, b) = \emptyset$.

On remarquera qu'un automate non-déterministe est en fait déterministe quand pour tout état p et toute lettre a, $|\delta(p, a)| \leq 1$ et qu'il est déterministe et complet quand $|\delta(p, a)| = 1$, pour tout $q \in Q$ et pour tout $a \in A$. On dira également qu'un automate non-déterministe est *complet* quand $|\delta(p, a)| \geq 1$, pour tout $q \in Q$ et pour tout $a \in A$.

La condition d'acceptation devient : un mot u est reconnu par un automate non-déterministe s'il existe un chemin d'étiquette u qui relie un état initial à un état terminal. Attention à cette définition : il y a un "il existe", on ne demande pas à ce que tous les chemins d'étiquette u soient acceptés, il suffit d'un seul. Par exemple, aa est reconnu par l'automate de l'exemple 2, en partant 3, même si aucun chemin ne partant de 1 ne le reconnaît.

1.6 Langages reconnaissables

Un langage est *reconnaissable* s'il existe un automate non-déterministe qui le reconnaît. On note $\mathrm{Rec}(A^*)$ l'ensemble de tous les langages reconnaissables sur l'alphabet A^*.

2 Algorithmique élémentaire des automates

2.1 Compléter un automate

On remarque tout d'abord que si un automate (déterministe ou non) n'est pas complet, alors il suffit de :

1. Ajouter un nouvel état π, non-terminal, avec $\delta(\pi, a) = \pi$ (ou $\delta(\pi, a) = \{\pi\}$ si on est dans le cas non-déterministe) pour tout $a \in A$;
2. Pour tout état p et pour toute lettre a, si $\delta(p, a)$ n'est pas définie (ou vaut $\emptyset$), alors on pose $\delta(p, a) = a$ ($\delta(p, a) = \{a\}$).

Cet algorithme simple transforme un automate en un automate complet qui reconnaît le même langage. Cette opération s'appelle la *complétion*. Pour un alphabet de taille fixée, la complexité de cet algorithme est linéaire en le nombre d'état de l'automate.

2.2 Complémentation

Soit $\mathcal{A} = (A, Q, \delta, q_0, F)$ un automate déterministe et complet. Le complémentaire de $\mathcal{A}$ est l'atomate $\mathcal{B} = (A, Q, \delta, q_0, Q \backslash F)$, où les états terminaux deviennent terminaux et réciproquement.

Proposition 1 *Si $\mathcal{A}$ est un automate déterministe et complet reconnaissant le langage $\mathcal{L}$, alors l'automate complémentaire de $\mathcal{A}$ reconnaît le complémentaire $A^* \setminus \mathcal{L}$ de $\mathcal{L}$.*

Attention, cette proposition est fausse si l'automate n'est pas déterministe ou pas complet.

2.3 Déterminisation

Soit $\mathcal{A} = (A, Q, \delta, I, F)$ un automate non-déterministe. L'*automate des sous-ensembles* de $\mathcal{A}$ est l'automate déterministe et complet $\mathcal{B} = (A, \mathcal{P}(Q), \lambda, I, F')$ défini par

— pour tout $X \in \mathcal{P}(Q)$ et pour tout $a \in A$, $\lambda(Q, a) = \cup_{q \in X} \delta(q, a)$;

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

On a la proposition suivante :

Proposition 2 *Un automate non-déterministe et son automate des sous-ensembles reconnaissent le même langage.*

Corollaire 3 *Un langage est reconnu par un automate non-déterministe, si et seulement si il est reconnu par un automate déterministe et complet.*

Corollaire 4 *L'ensemble $Rec(A^*)$ des langages reconnaissables est stable par complément.*

2.4 Automate produit

Soient $\mathcal{A} = (A, Q, \delta, I, F)$ et $\mathcal{A}' = (A, Q', \delta', I', F')$ deux automates définis sur le même alphabet. L'*automate produit* $\mathcal{B} = \mathcal{A} \times \mathcal{A}'$ de $\mathcal{A}$ et $\mathcal{A}'$ est l'automate $(A, Q \times Q', \tilde{\delta}, \tilde{I}, \tilde{F})$ défini par :

— $\tilde{I} = \{(i, i') \mid i \in I \text{ et } i' \in I'\}$;

— pour tout $q \in Q$, pour tout $q' \in Q'$ et pour tout $a \in A$, $\tilde{\delta}((q, q'), a) = \delta(q, a) \times \delta(q', a)$.

— pour $\tilde{F}$ cela dépend de ce que l'on veut faire, mais sauf indication contraire, nous prendrons $\tilde{F} = \{(f, f') \mid f \in F \text{ et } f' \in F'\}$.

On remarquera que si $\mathcal{A}$ et $\mathcal{A}'$ sont tous les deux déterministes, alors $\mathcal{B}$ est également déterministe.

Proposition 5 *L'automate produit de $\mathcal{A}$ et $\mathcal{A}'$ reconnaît l'intersection des langages reconnus par $\mathcal{A}$ et par $\mathcal{A}'$: l'ensemble $Rec(A^*)$ est stable par intersection.*

3 Théorème de Kleene

Le théorème de Kleene est le résultat fondateur de la théorie des automates. Il énonce l'équivalence entre les deux façons de décrire un langage que l'on vient de voir, par expressions rationnelles ou par automates.

Théorème 6 (Kleene) *$Rat(A^*) = Rec(A^*)$: les ensembles des langages rationnels et des langages reconnaissables sont identiques.*

La suite de cette partie est consacrée à la preuve de ce théorème.

3.1 Des expressions aux automates

Il existe plusieurs façon de transformer une expression rationnel en un automate qui reconnaît le même langage. Nous allons voir dans cet partie un de ces algorithmes, du à Glushkov et qui porte son nom.

La première étape consiste à *linéariser* l'expression. Il s'agit de numéroter les lettres pour que même s'il y a plusieurs occurrences de la même lettre, on puisse les distinguer. S'il y a n symboles de lettres dans l'expression, on les numérote de 1 à n de gauche à droite sur le mot de l'expression (ou par un parcours en profondeur si on adopte la représentation par un arbre). Un exemple est plus parlant : l'expression $E = ac(a+b)^*$ devient $\tilde{E} = a_1c_2(a_3+b_4)^*$. On notera ainsi $\tilde{E}$ l'expression linéarisée obtenue, et on indique les numéros des lettres en indice.

Si $\tilde{E}$ est une expression linéarisée sur l'alphabet $\tilde{A}$, *l'automate de Glushkov* de $\tilde{E}$ est l'automate non-déterministe $(\tilde{A}, Q, \{q_0\}, F)$ avec :

- Q est composé des lettres de $\tilde{E}$ ainsi que du symbole q_0 pour l'état initial ;
- pour tout $\alpha \in Q \setminus \{q_0\}$ on a une transition $q_0 \xrightarrow{\alpha} \alpha$ si et seulement si α commence au moins un mot du langage $\mathcal{L}(\tilde{E})$ reconnu par $\tilde{E}$;
- pour tout $\alpha, \beta \in Q \setminus \{q_0\}$ on a une transition $\alpha \xrightarrow{\beta} \beta$ si et seulement si le mot de deux lettres $\alpha\beta$ est facteur d'au moins un mot de $\mathcal{L}(\tilde{E})$.
- F est l'ensemble des lettres qui terminent au moins un mot de $\mathcal{L}(\tilde{E})$. On ajoute q_0 dans F si et seulement si $\varepsilon \in \mathcal{L}(\tilde{E})$.

Théorème 7 (Glushkov) *Si $\tilde{E}$ est une expression linéarisée, alors elle reconnaît le même langage que son automate de Glushkov.*

Corollaire 8 *Si un langage est dans $Rat(A^*)$, alors il est dans $Rec(A^*)$.*

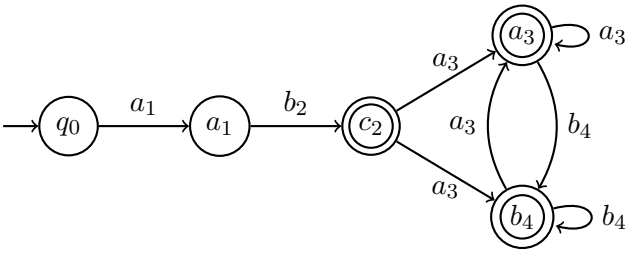


FIGURE 3 – L'automate de Glushkov de $a_1c_2(a_3 + b_4)^*$.

3.2 Des automates aux expressions

Il existe également plusieurs algorithmes pour transformer un automate en une expression qui reconnaît le même langage. Nous allons en voir un qui s'appuie sur le lemme classique suivant :

Lemme 9 (d'Arden) *Soit l'équation $X = E \cdot X \cup F$, où E, F et X sont des langages sur l'alphabet A et où X est l'indeterminée. On suppose de plus que $\varepsilon \notin E$. Alors $X = E^* \cdot F$ est l'unique solution.*

Soit $\mathcal{A} = (A, Q, \delta, q_0, F)$ un automate déterministe et complet. Le système d'équation associé à $\mathcal{A}$ est un système d'équations de langages, où les inconnues sont les L_q pour $q \in Q$, et tel que

pour chaque état $q \in Q$ on a l'équation

$$L_q = \bigcup_{q \xrightarrow{a} q'} a \cdot L_{q'} \ \cup \ \mathbb{1}_{q \in F},$$

où $\mathbb{1}_{q \in F} = \{\varepsilon\}$ si $q \in F$ et $\mathbb{1}_{q \in F} = \emptyset$ sinon.

Le système d'équation de $\mathcal{A}$ caractérise le langage reconnu par $\mathcal{A}$ au sens suivant :

Lemme 10 *Soit $\mathcal{A}$ un automate déterministe et complet d'état initial q_0. Le système d'équation associé à $\mathcal{A}$ admet une unique solution et $\mathcal{L}(\mathcal{A}) = L_{q_0}$.*

Calculer L_{q_0} se fait avec le Lemme d'Arden, de la même façon que le pivot de Gauss. En on déduit :

Proposition 11 *Tout langage reconnu par un automate déterministe et complet est également reconnu par une expression rationnelle.*

Corollaire 12 *Tout langage reconnu par un automate est également reconnu par une expression rationnelle.*