Automates sur les mots transfinis

Xun GONG

en retard

Résumé

Ce document présente une généralisation des automates finis sur les mots indexés par des ordinaux. Cette généralisation, donnée notamment par Büchi et Wojciechowski, permet d'établir l'équivalence entre ce type d'automate et des expressions rationnelles définies en ajoutant deux opérateurs, ω et #.

Table des matières

1 Rappels et notations

1.1 Ordinaux

Pour les discussion de base des ordinaux, on pourra consulter les livres classiques sur la théorie des ensembles. Pour la commodité, les notations utilisées ici sont proches de celles de [Deh06].

On appelle un *ordinal* un bon ordre dont les éléments sont tous les bons ordres plus petits. Les lettres minuscules grecques, $\alpha, \beta, \gamma, \ldots$, indiquent en général des ordinaux. On note le successeur de α comme $\alpha + 1$. La classe des ordinaux $\mathtt{Ord} = \{0\} \cup \mathtt{Succ} \cup \mathtt{Lim}$, où $\mathtt{Succ}$ est la classe des successeurs, et $\mathtt{Lim}$ celle des ordinaux limites. sss Les opérations arithmétiques sur les ordinaux ont leurs significations canoniques. Puisque $\alpha+\beta = \alpha+\beta'$ entraîne que $\beta = \beta'$, on définit, pour $\beta < \alpha$, $\alpha - \beta$ comme l'unique γ, s'il existe, tel que $\alpha = \beta + \gamma$.

On présente ensuite un concept assez important, la cofinalité. Intuitivement, la cofinalité fonctionne pour un relation d'ordre quelconque, de la même façon de l'apparition infinie pour l'ordre ω ; une partie B est dit cofinal dans un order total A si B n'est pas borné dans A.

Définition 1.1 (cofinal)**.** Soit A un ensemble totalement ordonné par l'ordre $<$, on dit que une partie B de A est *cofinale* dans A si pour tout $a \in A$, il existe un $b \in B$ tel que $b \geq a$.

Exemples.

- $\mathbb{N}$ est cofinal dans $\mathbb{R}$, pour l'ordre usuel.
- $\{\omega n + 1, n \in \mathbb{N}\}$ est cofinal dans ω^2

1.2 Opérations rationelles sur les mots transfinis

Sur un ensemble A fini, une α-suite est une fonction $f : \alpha \to A$. On la note parfois avec la notation de suite, $f_\beta := f(\beta)$, ou bien ${}_\beta f := f(\beta)$. On appelle α la longeur d'une α-suite. La classe des suites sur I est notée comme $I^\#$, et la restriction d'une α-suite en β comme $I|_\beta$.

On utilise le mot *mot* pour une suite sur un alphabet.

Rappelons qu'un langage rationnel des mots finis est défini par récurrence des opérateurs dites rationnels (l'union, le produit et l'étoile). Maintenent, on cherche à définir des langage rationnels transfinis en ajoutant des autres opérations. L'opération de réunion est bien défini, étant donné une théorie avec des classes propres. Cependant, les opérations multiplicatives ne se prolongent pas *a priori*. Il faut donc décrire une façon de faire le produit des mots transfinis.

Définition 1.2 (sous-suite)**.** Soit h une α-suite, on note par $h\,[\beta, \gamma[\,(\delta)$ la $(\gamma - \beta)$-suite telle que $h\,[\beta, \gamma[\,(\delta) = h(\beta + \delta)$.

Remarque. Il est facile à voir qu'une suite transfinie est bien définie si une famille des sous-suites la couvrant le sont.

Définition 1.3 (produit)**.** Soient B et C deux classes des mots transfinis (deux *langages transfinis*) sur l'alphabet A. Le langage produit $B \cdot C$ est une sous-classe de $A^\#$,

$$B \cdot C := \{\nu / \exists \alpha, \nu\,[0, \alpha[\in A \text{ et } \nu\,[\alpha, \text{longueur de } \nu[\in B\}.$$

Intuitivement, un mot ν appartient à B^α, s'il existe une coupure de B à α parties, dont chacune est dans le langage de base.

Définition 1.4 (puissance)**.** On dit une suite h est *propre* si elle est croissante, et de plus si tout son élément limite ζ est la borne supérieure de $\{h_\beta / \beta < \zeta\}$.

Soient B un langage transfini sur A, et α un ordinal quelconque. B^α est la classe des mots ν tel qu'il existe une $(\alpha + 1)$-suite propre β, où $\beta_0 = 0$ et $\beta_\alpha =$ la longueur de ν, et pour tout $\gamma \in \mathrm{Succ} \leq \alpha$, $\nu\,[\beta_\gamma, \beta_{\gamma+1}[\in A$.

Exemple.

Il est désormais naturel de définir le reste des opérations.

Définition 1.5 (fermetures)**.** Soit B un langage transfini sur A, on définit les opérations $\star$, ω et #

$$B^\star := \bigcup_{n \in \mathbb{N}} B^{\underline{n}},$$
$$B^\omega := B^\omega,$$
$$B^\# := \bigcup_{\alpha \in Ord} B^\alpha.$$

L'usage de $A^\#$ s'est justifié.

Définition 1.6 (expression rationnelle)**.** La famille d'*expressions rationnelles* $\mathcal{E}$ sur les mot transfinis est la plus petite classe qui vérifie

- $\emptyset \in \mathcal{E}$;
- Pour tout lettre a, $a \in \mathcal{E}$;
- Pour toute expression $E, F \in \mathcal{E}$, $(E \cup F)$, $(E \cdot F)$, $(E^\star)$, (E^ω), $(E^\#)$.

On peut alors définir, par la manière canonique, un langage rationnelle.

1.3 Suites continues

On note $\mathfrak{P}\mathfrak{z}(S)$, où S est un ensemble fini, $\mathcal{P}(S) \cup S$. Les éléments de $\mathfrak{P}\mathfrak{z}(S)$ se divise bien en les éléments de S et les parties de S. En discutant $\mathfrak{P}\mathfrak{z}(S)$, on appelle ses éléments qui sont dans S les *éléments reguliers*, et ceux qui sont dans $\mathcal{P}(S)$ les *éléments limites*. La raison pour les noms sera claire bientôt.

Définition 1.7 (suites continues)**.** Une *suite continue* h sur I est une λ-suite sur $\mathfrak{P}\mathfrak{z}(I)$ vérifiant les conditions suivantes :

a) $h_0 \in I$;
b) $h_\alpha \in I$ si $\alpha \in Succ$;
c) $h_\alpha = \{s \in I, \{\beta < \alpha, h_\beta = s\}$ est cofinal dans $\alpha\}$.

Evidemment, une suite continue est déterminée uniquement par ses valeurs non limites.

Exemple. Pour une suite infinie a_i, il existe une extension unique de la suite de longeur $\omega + 1$. Par exemple, la suite $101010\ldots$ a sa valeur dans ω bien définie comme $\{0, 1\}$.

Remarque. Plus généralement, on peut éliminer les suites continues de longueur limite, en les prolongeant uniquement à elles-mêmes.

Notations. On prend les notations suivantes, pour un certain suite continue h.

2 $\mathcal{W}$-automates

Rappelons que un automate *accepte* un mot s'il existe un chemin de l'état initial à un état final. Étant donné une façon naturelle de prolonger les suites finies, il est facile à considérer un automate analogique, ce qu'on appelle un $\mathcal{W}$-automate, en prenent la suite de [Bed96].

Définition 2.1 ($\mathcal{W}$-automate)**.** Un $\mathcal{W}$-*automate* sur un alphabet fini A est un quintuplet (Q, A, E, I, F), avec Q l'ensemble des états, $I \in Q$ l'état initial, $E \in \mathfrak{P}_{\mathfrak{z}}(Q) \times A \times Q$ les transitions d'une situation (c'est-à-dire, un élément de $\mathfrak{P}_{\mathfrak{z}}(Q)$) vers un état, et F l'ensemble des situations finales. On appelle le triplet (Q, A, E) un *quasi-automate*, vu comme la restriction d'un automate.

Définition 2.2 (chemin)**.** Soit (Q, A, E) un quasi-automate. Un *chemin* $h_0 \xrightarrow{\nu} h_\alpha$ dans cet automate sur le mot ν dont la longueur est α, est une $(\alpha + 1)$-suite continue h où la relation $(h_\beta, \nu_\beta, h_{\beta+1})$ se verifie pour tout $\beta \leq \alpha$.

Définition 2.3 (acceptation)**.** Soit (Q, A, E, I, F) un $\mathcal{W}$-automate. On dit qu'il *accepte* un mot ν de longueur α s'il en existe un chemin continu $I \xrightarrow{\nu} q$ où q est une situation finale.

Exemples. On considère d'abord les exemples simples suivants :
- Les automates finis sur les mots finis sont des $\mathcal{W}$-automates.
- Intuitivement, un automate fini sur les mot infinis est bien un $\mathcal{W}$-automate, par exemple, l'automate suivant accepte le langage $(ab^\star)^\omega$.

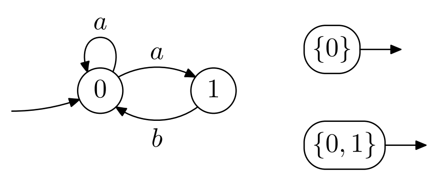

FIG. 1 – L'automate acceptant un ω-mot sur $\{a, b\}$ avec le nombre de a infini.

2.1 Un $\mathcal{W}$-automate à partir d'une expression rationnelle

On se concentre, pour le moment, sur la constuction non déterministe. On peut donc utiliser la construction naïve (chez Kleene) pour la réunion, le produit et le fermeture de Kleene.

Lemme 2.4 (construction ω)**.** *Soit* (Q_B, A, E_B, I_B, F_B) *un automate dont le langage est* $\mathcal{B}$*, on peut construire un automate* (Q_C, A, E_C, I_C, F_C) *reconnaissant le langage* $\mathcal{C} = \mathcal{B}^\omega$*.*

Explication. Il est facile de le faire par analogie avec les $\mathcal{W}$-automates reconnaissant les simples ω-langages. On remarque que pour reconnaître l'opérateur ω, tout ce qu'il faut est dans un sens «compter» un nombre ω d'un état certain. Cependant, ce n'est pas directement réalisable, car tout situation limite est composée par des états simple, plutôt que d'autres situations limites.

La solution, donnée par [Bed96], est de colorier les états qu'on veut apparaître infiniment. Formellement, l'automate recherché est

$$(Q_B \times \{0, 1\}, A, E_C, (I_B, 0), F_C)\,,$$

où E_C contient seulement les relations indiquées suivant

- Pour tout $(q_1, a, q_2) \in E_B$ où $q_1 \in Q_B$, on a
$$((q_1, c), a, (q_2, 0)) \in E_C,$$
quel que soit c.
- Pour tout $(q_1, a, q_2) \in E_B$ où $q_1 \in \mathcal{P}(Q_B)$, on a
$$((q_1, 0), a, (q_2, 0)) \in E_C.$$
- Pour tout $(I_B, a, q_2) \in E_B$, on a
$$(q_1', a, (q_2, 1)) \in E_C,$$
où
$$q_1' \in (F_B \cap Q_B) \times \{0, 1\} \cup (F_B \cap \mathcal{P}(Q_B)) \times \{0\}.$$

L'ensemble des situations finales
$$F_C = \begin{cases} \{s \in \mathcal{P}(E_C) / \exists q, (q, 1) \in s\} & \text{si } I_B \notin F_B, \\ \\ \{s \in \mathcal{P}(E_C) / \exists q, (q, 1) \in s\} \cup & \\ \quad \{s \in \mathcal{P}(E_C) / \{q / (q, c) \in s, \forall c\} \in F_B\} \cup & \\ \quad (F_B \times \{0, 1\} \cap E_C) & \text{si } I_B \in F_B. \end{cases}$$

$\square$

Exemple. Considérons ce simple automate,

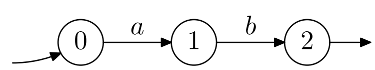

FIG. 2 – L'automate acceptant le langage ab.

On applique la procédure décrite ci-dessus, en obtenent :

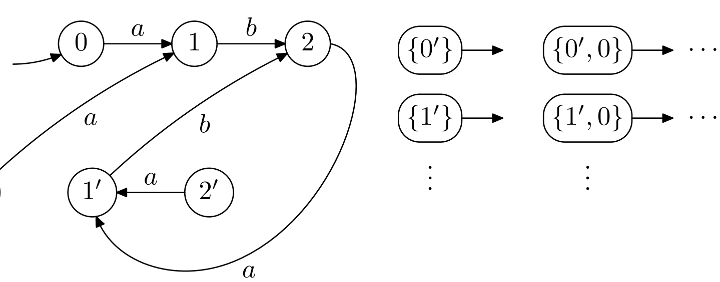

FIG. 3 – L'automate acceptant le langage $(ab)^\omega$.

ou bien :
L'identité avec le langage voulu est évident.

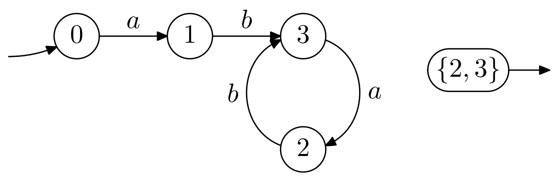

FIG. 4 – L'automate acceptant le langage $(ab)^\omega$, avec les situations superflues enlevées.

Lemme 2.5 (construction #). *Soit* (Q_B, A, E_B, I_B, F_B) *un automate dont le langage est* $\mathcal{B}$*, on peut construire un automate* (Q_C, A, E_C, I_C, F_C) *reconnaissant le langage* $\mathcal{C} = \mathcal{B}^\#$.

Explication. On prend une solution similaire, en coloriant en deux les états, et en ajoutant un nouvel état initial pour y comprendre le mot vide.

On se donne l'automate (Q_C, A, E_C, s, F_C) où $Q_C = \{s\} \cup Q_C \times \{0, 1\}$ et E_C ne contient que les transitions énumérées suivant :

- Pour tout $(I_B, a, q) \in E_B$, $(s, a, (q, 0)) \in E_C$.
- Pour tout $(q_1, a, q_2) \in E_B$ où $q_1 \in Q_B$,

$$((q_1, c), a, (q_2, 0)) \in E_C,$$

  pour c égal à 0 ou 1.
- Pour tout $(q_1, a, q_2) \in E_B$ où $q_1 \in \mathcal{P}(Q_B)$,

$$((q_1, 0), a, (q_2, 0)) \in E_C.$$

- Pour tout $(I_B, a, q_2) \in E_B$, on a

$$(q_1', a, (q_2, 1)) \in E_C,$$

  où

$$q_1' \in (F_B \cap Q_B) \times \{0, 1\} \cup (F_B \cap \mathcal{P}(Q_B \times \{0\})).$$

- Pour tout $(I_B, a, q) \in E_B$,

$$(t, a, (q, 1)) \in E_C,$$

  où $t \in \mathcal{P}(E_C)$ et $\exists q, (q, 1) \in t$.

Les situations finales sont

$$\{s\} \cup (F_B \cap Q_B) \times \{0, 1\} \cup (F_B \cap \mathcal{P}(Q_B)) \times \{0\} \cup \{t \in \mathcal{P}(E_B \times \{0, 1\}) / \exists q, (q, 1) \in t\}.$$

□

Exemple. Prenons encore l'automate reconnaissant ab, comme la Figure 2.

On l'applique la procédure décrite ci-dessus :

Comparaison avec la Figure 3 révèle l'ensemble final agrandi pour les cas vide et non-limite, et plus les transitions pour continuer la course en temps transfini.

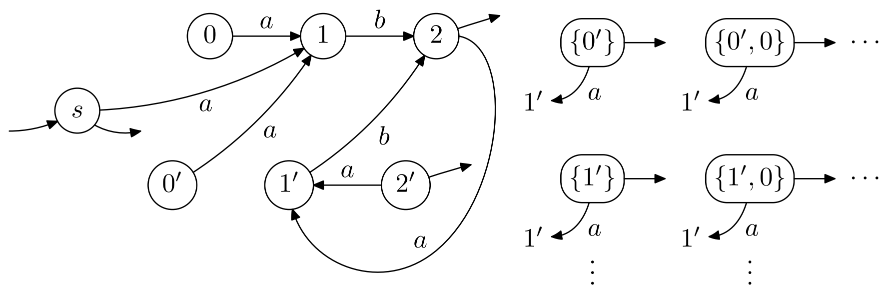

FIG. 5 – L'automate acceptant le langage $(ab)^{\#}$.

Théorème 2.6 (théorème de Kleene, 1ère partie)**.** *Pour toute expression rationelle transfinie, il existe un automate équivalent.*

2.2 Une expression rationnelle à partir d'un $\mathcal{W}$-automate

Il est possible d'avoir une expression rationelle dont le langage décrit est pareil qu'un $\mathcal{W}$-automatedonné. La construction est trop technical pour une discussion sensée ici. En gros, il s'agit de «transporter» le preuve classique de la théoreme de Kleene. Dans le preuve classique, on crée, d' une façon ne pas trop différente que dans l'algorithme Floyd-Warshall, des ensembles sous la forme $\mathcal{E}_k(q_i, q_j)$, qui représente un ensemble des mots sur lesquels l'automate prend un chemin de q_i à q_j, en ne jamais entrant un état intermédiaire qui majore k.

On peut prolonge cet idée à l'ensemble des $\mathcal{W}$-situations, où il exist un ordre bien fondé naturel. Donc on peut numéroter les formes $\mathcal{E}_Z(s_1, s_2)$, où s_1 et s_2 sont des situations, et $Z \in \mathcal{P}(\mathfrak{P}_\mathfrak{z}(Q))$, avec quelques contraints nécessaires de l'ensemble Z. L'automate vient du s_1 au s_2 en traversant eventuellement tous les situations dans Z, quand l'ensemble maximum des états traversés est Z. On note $\mathcal{E}\mathcal{E}_Z(s_1, s_2)$ pour les ensembles dont il n'est pas possible de traversée totale.

On consultera [Woj85] pour les technicalités de cette proposition :

Théorème 2.7 (théorème de Kleene, 2nde partie)**.** *Pour tout $\mathcal{W}$-automate dans un certain langage, il existe une expression rationelle transfinie équivalente.*

Pour conclure, le théorème de Kleene des $\mathcal{W}$-automates découle facilement la décidabilité du leur problème vide.

Références

[Bed96] Nicolas Bedon. Finite automata and ordinals. *Theoretical Computer Science*, 156 :119–144, March 1996.

[Deh06] Patrick Dehornoy. Logique et théorie des ensembles ; *Notes de cours*. http://www.math.unicaen.fr/~dehornoy/surveys.html, 2006.

[Woj85] Jerzy Wojciechowski. Finite automata on transfinite sequences and regular expressions. *Annales Societatis Mathematicae Polonae Series IV : Fundamenta Informaticae*, VIII.3.4 :379–396, 1985.