Automates à pile déterministes

Mathieu Barbin

décembre 2007

Table des matières

1 Introduction

De manière intuitive, un automate à pile déterministe est un automate à pile tel que pour chaque entrée, un seul calcul est possible. A l'inverse des automates finis qui sont déterminisables, on montre que les automates à pile déterministes reconnaissent une sous classe-stricte des langages algébriques. On perd également l'équivalence entre les différents modes d'acceptation pour ces automates. Cette classe montre l'intérêt d'être stable par complémentation, et possède un grand terrain d'application en analyse syntaxique des langages de programmation, le determinisme offrant la possibilité de générer automatiquement des analyseurs syntaxiques efficaces (analyseurs LR). On montre également que tout langage déterministe est non ambigu (la réciproque étant fausse) et on termine en évoquant un joli résultat assez récent, $L(A) = L(B)$ est décidable pour A et B automates à pile déterministes .

2 Formalisation

On rappelle la définition du cours, ou Q désigne l'ensemble des états de l'automates, A l'alphabet d'entrée, et Z l'alphabet de pile :

Définition 1. *Un automate à pile (Q, A, Z, q_0, z_0) est déterministe si pour toute paire (p, z) de $Q \times Z$,*

1. *soit il existe une unique ϵ-transition $(p, \epsilon, z \to q, h)$ et dans ce cas aucune autre transition $(p, a, z \to q, h)$ pour tout a dans l'alphabet d'entrée,*
2. *soit il n'existe pas d'ϵ-transition à partir de (p, z) et pour chaque lettre $a \in A$, il existe au plus une transition $(p, a, z \to q, h)$*

Avant de définir les *langages algébriques déterministes*, il est nécessaire de préciser le mode d'acceptation de l'automate puisque ceux-ci ne sont pas équivalents. On montre $PV \subset EF$ (inclusion stricte) et $EF = SSP$ (reconnaissance par Pile Vide, Etats Finaux ou Symbole de Sommet de Pile). Dans la suite, on parle aussi *d'états finaux* pour désigner les états d'acceptation de l'automate, dans le cas de ce mode de reconnaissance.
La proposition suivante donne une caractérisation des langages reconnu par un automate à pile déterministe , par PV.

Proposition 1. *Un langage reconnu par un automate à pile déterministe par pile vide est un langage préfixe (deux mots du langage ne sont pas préfixe l'un de l'autre.)*

Preuve. Soit L un langage reconnu par automate à pile déterministe $P = (Q, A, Z, q_0, z_0)$ par pile vide. Supposons L non préfixe. Cela signifie qu'on peut trouver 2 mots a et ab dans L avec $b \neq \epsilon$. On a donc nécessairement la transition $(q, a, z) \to (q1, \epsilon)$ puisque a est reconnu par PV. Comme le calcul est deterministe, en prenant ab comme mot d'entrée, on passe par les configurations $(q, ab, z) \hookrightarrow (q1, b, \epsilon)$ et le calcul s'arrete puisque la pile est vide. Comme tout le mot n'a pas été lu, ab n'est pas dans L. Contradiction.

Proposition 2. *Un langage est reconnu par un automate à pile déterministe par états finaux si et seulement si il est reconnu par un automate à pile déterministe par symbole de sommet de pile.*

Preuve

1. si L est reconnu par un automate à pile déterministe $P = (Q, A, Z, q_0, z_0)$ par symbole de sommet de pile $z \in Z$, alors on construit un nouvel automate P' dans lequel on ajoute pour chaque état $q \in Q$ un nouvel état conjugué q' et dans lequel on remplace toutes les transitions $(q1, a, z) \to (q2, h)$ par 2 nouvelles transitions

$$\begin{cases} (q1, \epsilon, z) \to (q1', z) \\ (q1', a, z) \to (q2, h) \end{cases}$$

   On vérifie alors que $P' = (Q \cup Q', A, Z, q_0, z_0)$ reconnait L par états finaux Q'.

2. reciproquement, si L est reconnu par un automate à pile déterministe $P = (Q, A, Z, q_0, z_0)$ par états finaux F. On peut supposer P à fond de pile testable. On commence par ajouter pour chaque couple (état final f, lettre de pile z) un nouvel état $q_f z$, et on ajoute un nouveau symbole de pile t qui servira de symbole de sommet de pile. On remplace les transitions $(f, a, z) \to (qh)$ pour tout ètat final f par les 2 transitions

$$\begin{cases} (f, \epsilon, z) \to (q_{fz}, t) \\ (q_{fz}, a, t) \to (q, h) \end{cases}$$

On doit de plus remplacer toute transition $(q, a, d) \to (q', \epsilon)$ où d est un symbole de fond de pile par la transition $(q, a, z) \to (q', t)$
L'automate ainsi obtenu est un automate à pile déterministe qui reconnait L par symbole de sommet de pile t.

Proposition 3. *Si L est un langage reconnu par un automate à pile déterministe par pile vide, alors il est reconnu par un automate à pile déterministe par états finaux. La réciproque est fausse.*

Preuve On peut par exemple ajouter un nouveau symbole de pile t initial, et on ajoute la règle initiale $(q_0, \epsilon, t) \to (q_0, z_0)$. Si un calcul vidait la pile, dans ce nouvel automate, la pile n'est plus vide, puisqu'elle contient t. On ajoute alors un nouvel état f, et les transitions suivantes : $(q, \epsilon, t) \to (f, t)$ pour tout état $q \in Q$. L'automate ainsi construit reconnait L par EF $\{f\}$. Pour la réciproque, il suffit de considérer un langage algébrique déterministe qui ne soit pas préfixe.

Définition 2. *La reconnaissance par états finaux étant plus expressive, on définit les langages algébriques déterministes comme étant les langages reconnus par des automates à pile déterministes par états finaux.*

3 Cloture par complémentation

Avant d'aller plus loin, on fait quelques remarques sur les ϵ-transitions qui est essentiellement la raison pour laquelle la démonstration de la cloture par complémentation des langages algébriques déterministes est assez technique.

S oit P un automate à pile déterministe . Pour un mot a, q_0 etat initial, et z_0 mot de pile initial, supposons que $(q_0, a, z_0) \hookrightarrow (q', \epsilon, z')$ soit un calcul valide de P. q' et z' ne sont pas déterminés de facon unique, car il est possible que dans le calcul après la lecture de la dernière lettre de a, il y ait encore des ϵ-transitions . On ne peut donc pas parler d'unicité du calcul valide partant de la configuration (q_0, a, z_0) et englobant la lecture de a.

Définition 3 (calcul valide maximal)**.** *Par contre, de la meme facon que precedemment, on montre que s'il y a plusieurs tels calculs, ceux-ci sont préfixes les uns des autres, et il y a unicité du calcul valide maximal. On le notera $(q_0, a, z_0) \hookrightarrow^* (q', \epsilon, z')$ pour indiquer que c'est un cacul valide maximal, c'est à dire qu'aucune ϵ-transition n'est applicable dans l'état q' avec z' dans la pile.*

Dans un premier temps pour simplifier le probleme, on peut s'intérresser aux automates à pile déterministes ne comportant pas d'ϵ-transition .

3.1 Cas simplifié d'un automate à pile déterministe real-time

Définition 4. *On dit d'un automate à pile déterministe qu'il est real-time s'il ne comporte pas d'ϵ-transition.*

L'idée étant que si à chaque transition de calcul, on lit effectivement un caractère, le calcul se termine en un nombre de transitions qui est au plus la longueur du mot. A l'inverse, si on permet les ϵ-transitions, le temps de calcul n'est pas borné.

Proposition 4. *Les automates à pile déterministes real-time sont strictement moins expressifs que les automates à pile déterministes .*

L'exemple suivant donne l'intuition d'un langage algébrique déterministe, non reconnu par un automate à pile déterministe real-time.

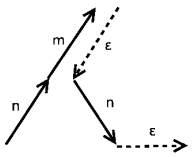

FIG. 1 – Etat de pile pour $L = a^n b^m c^n d^p$

3.2 Passage au complémentaire des états finaux

Si P est un automate à pile déterministe real-time reconnaissant un langage L, on peut voir qu'il suffit presque de considérer l'automate P', dans lequel on inverse les états acceptants des états non acceptants, reconnait L^c. Un seul détail cependant, si un mot w de A^* conduit à vider la pile avant d'être lu totalement, alors $w \notin L$. Le probleme, est que dans P', w videra également la pile, et ne sera pas non plus accepté. Pour résoudre ce probléme, on introduit le lemme suivant :

Lemme 1. *Si L est un langage déterministe, il est reconnu par états d'acceptation par un automate à pile déterministe qui ne se bloque jamais, i.e. tel que pour toute configuration (q, a, h) avec h non vide, le calcul valide maximal englobe la lecture complète du mot a.*

Remarque 1. *Dans un tel automate P_{nb} (non bloquant), plus aucun mot n'est rejeté par PV, seule la nature de l'état d'arrivée en fin de calcul est déterminante, il suffit donc de passer au complémentaire des états finaux dans P_{nb} et non pas dans P.*

Preuve du lemme Supposons L reconnu par un automate à pile déterministe à fond de pile testable $P = (Q, A, Z, q_0, z_0)$ par états d'acceptation F. Soit $Z_1 \subset Z$ l'ensemble des symboles de fond de pile. Soit q_p un nouvel état et t un nouveau symbole de pile. On pose $Q' = Q \cup \{q_p\}$ et $Z' = Z \cup \{t\}$ et on construit $P' = (Q', A, Z', q_0, z_0)$ à partir de P, en conservant les mêmes transitions que dans P, et en ajoutant les transitions suivantes :

$$\forall a \in A (q_p, a, t) \rightarrow (q_p, t)$$

Il reste 2 cas de bloquage à régler :

1. P pouvait se bloquer si pour $q \in Q$ et $z \in Z$, aucune ϵ-transition n'était applicable, et simultanément pour un $a \in A$, aucune transition de P n'avait pour membre gauche (q, a, z). Pour ne pas que P' se bloque dans ce cas, on ajoute alors à P' la transition $(q, a, z) \rightarrow (q_p, t)$.
2. P pouvait également se bloquer si la pile devenait vide avant la fin de la lecture du mot, c'est à dire si on appliquait une transition $(q, y, z) \rightarrow (q', \epsilon)$ avec $z \in Z$ un symbole de fond de pile. Pour remédier à ce cas, on remplace cette transition dans P' par $(q, y, z) \rightarrow (q', t)$ et on ajoute les règles

$$\forall a \in A, \forall q \in Q, (q, a, t) \rightarrow (q_p, t)$$

L'automate P' ainsi obtenu est un automate à pile déterministe qui reconnait le même langage que P et tel que pour tout mot m de A^*, le calcul de P' sur (q_0, m, z_0) termine sans bloquer.
On peut alors passer au complémentaire dans les etats, et considérer le même automate, en fixant $F' = Q' \setminus F$ comme nouveau état acceptant. Un tel automate reconnait le complémentaire de L.

3.3 Cas général d'un automate à pile déterministe

Nous avons donc vu que si l'automate ne comporte pas d'ϵ-transition , il est possible de construire assez simplement le complémentaire. Malheuresement, comme on l'a dit, le fait d'enlever les ϵ-transitions réduit la classe des langages considérés. On est donc obligé de traiter explicitement les ϵ-transitions lors de la complémentation. La difficulté des ϵ-transitions provient du fait que l'on peut passer par une ϵ-transition à partir d'un état acceptant vers un état qui ne l'est pas (ou l'inverse) et que si l'on se contente de changer Q en $Q \setminus F$, certains mots seront reconnus dans les deux automates. Pour résoudre ce probléme, on va utiliser le lemme suivant, qui nous permet d'enlever de telles ϵ-transitions , (en conservant les autres, qui ne posent à priori pas de problème).

Lemme 2. *Si L est un langage déterministe, il est reconnu par un automate à pile déterministe qui ne bloque jamais par états finaux F, et tel que dans tout état de F, aucune ϵ-transition n'est applicable.*

Preuve Supposons L reconnu par un automate à pile déterministe qui ne se bloque jamais $P = (Q, A, Z, q_0, z_0)$ par états d'acceptation F. (un tel automate existe d'après le lemme précédent). On fait une construction en deux temps :

1. Dans un premier temps, on construit un automate P' ayant deux fois plus d'états, les nouveaux états permettant de savoir si on a rencontré un état d'acceptation depuis la dernière lettre lue. A partir de $Q = \{q_0, q_1, \ldots, q_n\}$, on pose $Q' = \{q'_0, q'_1, \ldots, q'_n\}$ un nouvel ensemble d'états en bijection avec Q. On définit $P' = (Q \cup Q', A, Z, q, z_0)$ en construisant les transitions à partir de celles de P' en ajoutant à toutes les ϵ-transitions de P les transitions suivantes :
   si $(q_i, \epsilon, z) \to (q_j, h)$ est une transition de P alors $(q'_i, \epsilon, z) \to (q'_j, h)$ est une transition de P'. De plus, si q_j est un état final, alors la règle $(q_i, \epsilon, z) \to (q_j, h)$ est remplacée par $(q_i, \epsilon, z) \to (q'_j, h)$.
   Ensuite, pour toute règle $(q_i, a, z) \to (q_j, h)$ avec $a \in A$ on ajoute dans P' la règle correspondante avec q_i, q'_i, q_j, q'_j selon qu'ils appartiennent ou non aux états finaux. L'automate ainsi obtenu est un automate à pile déterministe qui reconnait L par états finaux Q' et qui est tel que m un mot de A^* est reconnu par P' si et seulement si le calcul valide maximal englobant m, amène à un état final : si $(q, m, z_0) \hookrightarrow^* (q', \epsilon, h)$, $m \in L$ si et seulement si $q' \in Q'$.
2. Dans un second temps, on construit un automate P'' vérifiant la condition du lemme. Si au cours d'un calcul valide de P' on applique un règle ayant (q'_i, a, z) avec $a \in A$ pour membre gauche, c'est que le mot déjà lu est un mot reconnu. Dans ce cas, nous allons insérer dans ce calcul valide une ϵ-transition en introduisant des nouveaux états Q'' en bijection avec Q, et en remplacant la transition $(q'_i, a, z) \to (q_j, h)$ ou, selon le cas resp. $(q'_i, a, z) \to (q'_j, h)$ par les deux règles $(q'_i, \epsilon, z) \to (q''_i, z)$ et $(q''_i, a, z) \to (q_j, h)$ resp $(q''_i, a, z) \to (q'_j, h)$
   Dans ces conditions, si $m \in L$, le calcul valide maximal est prolongé et amène dans un état de Q'', mais clairement un état de Q'' ne peut être atteint que par un calcul valide maximal. (par construction des nouvelles règles). L'automate P'' ainsi obtenu reconnait bien L, par états finaux Q'', et d'un état d'acceptation aucune ϵ-transition ne peut être appliquée.

fin de la construction du complémentaire : Ce lemme étant aquis, on verifie que l'automate obtenu après la première construction est tel que l'on ne peut plus passer d'un état d'acceptation à un état qui ne l'est pas par une ϵ-transition . La deuxième construction fait que les mots de L sont reconnus grace à des calculs valides maximaux, seuls amenant à des états d'acceptation. Etant donné L algébrique déterministe reconnu par P, on fait une construction analogue au lemme pour construire P_c reconnaissant L_c. Si $Q'' = \{q''_0, \ldots, q''_n\}$ est comme dans le lemme, nous allons nous servir de ces états pour indiquer que depuis la dernière lettre lue, on n'est jamais passé dans un état d'acceptation de l'automate de départ.
Cette fois, ce sont les règles $(q_i, a, z) \to (q_j, h)$ ou $(q_i, a, z) \to (q'_j, h)$ qui sont remplacées par $(q_i, \epsilon, z) \to (q''_j, z)$ et selon le cas $(q''_i, a, z) \to (q_j, h)$ ou $(q''_i, a, z) \to (q'_j, h)$. On obtient ainsi un automate à pile déterministe qui reconnait L_c par états d'acceptation Q''.

4 Tout langage algébrique déterministe est non ambigu

Proposition 5. *Tout langage algébrique déterministe est non ambigu.*

Preuve La preuve détaillée se trouve dans [Aut87]. On n'en expose ici que l'idée générale. On se donne $L = L(\mathcal{P}, F)$ un langage algébrique déterministe, reconnu par un automate à pile déterministe $\mathcal{P} = (Q, A, Z, q_0, z_0)$ par état d'acceptation F. La première idée est de diviser le langage en une partition de même cardinal que F :

$$L = \bigcup_{f \in F} L(\mathcal{P}, \{f\})$$

On peut comme dans les lemmes construire $\mathcal{P}$ tel que les états de F ne sont atteints que par des calculs valides maximaux, et que pour tout mot donné le calcul valide maximal est unique, les langages L_f sont disjoints. On est donc ramené a prouver que chacun des langages L_f est non-ambigu. A partir de $\mathcal{P}$ on construit un automate P' qui va reconnaitre L_f par pile vide. (plus nécessairement déterministe). On est amené à considérer la grammaire associée à l'automate à pile P', et on montre par l'absurde en considérant deux dérivations distinctes de longueur n donnant lieu à un même mot de L_f, que nécessairement l'ambiguité provient d'une dérivation de longueur inférieure à n. On montre donc par récurrence que cette grammaire est non-ambigue. Ceci étant fait pour f quelconque dans F, et une union distincte de langages non-ambigu étant non ambigue, cela montre bien L non ambigu.

Proposition 6. *Il existe des langages non ambigus qui ne sont pas déterministes.*

En effet, le langage $L = \{a^n b^n \mid n > 0\} \cup \{a^n b^{2n} \mid n > 0\}$ est un langage non ambigu puisqu'il est engendré par la grammaire non ambigue suivante :

$$\begin{cases} S \to D \mid U \\ D \to aDbb \mid abb \\ U \to aUb \mid ab \end{cases}$$

L'intuition pour voir que L n'est pas déterministe est qu'une fois avoir lu les n a, et à la lecture du premier b, on ne sait pas s'il faut prévoir d'en lire n ou $2n$. Comme le calcul est déterministe, on ne peut laisser les différentes transitions comme on pourrait le faire avec un automate à pile classique.
Une preuve formelle du déterminisme de L de cet exemple peut etre obtenu à partir d'un lemme d'itération adapté pour les langages algébriques déterministes. (cf compléments)

Remarque 2. *En recapitulant, on a donc montré les inclusions strictes suivantes :*

$$Det_{PV} \subset Det_{EF} \subset AlgNonAmbigus \subset Alg$$

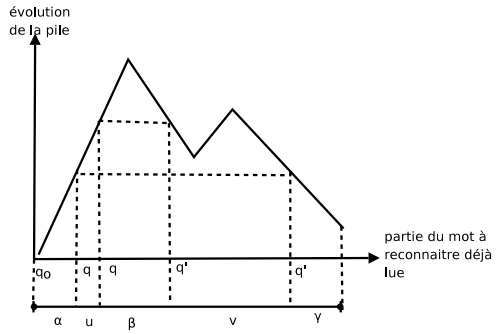

FIG. 2 – lemme d'itération

5 Compléments

5.1 Lemme d'itération pour L algébrique déterministe

On trouve une version du lemme d'itération pour les langages algébriques analogue au lemme d'Ogden pour les langages algébriques.

Proposition 7. *Pour tout langage déterministe L, il existe un entier k tel que tout mot f de L dont on a distingué au moins k lettres se factorise en $f = \alpha u \beta v \gamma$ avec :*

1. $\forall n \in \mathbb{N}, \alpha u^n \beta v^n \gamma \in L$
2. *soit α, u et γ soit γ, v et γ contiennent chacun une lettre distinguée au moins*
3. *$u\beta v$ contient moins de k lettres distinguées*
4. *si γ n'est pas vide, alors quel que soit $w \in A^* \exists p$ tel que $\gamma u^p \beta v^p w \in L \Rightarrow \forall p : \alpha u^p \beta v^p w \in L$*

La preuve de cette proposition repose sur l'intuition exprimée par le schéma de l'évolution de la pile lors du calcul valide menant à la reconnaissance du mot f de la figure 2.

5.2 Décidabilité de L(A) = L(B), A et B APD

Depuis le milieu des années 60, on s'interressait au problème de l'équivalence des automates à piles déterministes. En janvier 1997, G. Sénizegues chercher de l'équipe *Langages Formels et Graphes* du LaBRI de Bordeaux a montré la décidabilité de ce problème, en décrivant un algorithme de décision (166 pages) [1] G.Sénizergues a proposé de gén'eraliser le problème initial en examinant l'ensemble des *"séries rationnelles déterministes"* écrites sur un alphabet Z (essentiellement l'alphabet de pile). L'ensemble de ces séries forme une structure algébrique nouvelle, nommée *espace déterministe*, qui présente des analogies

[1] URL, http ://www.labi.u-bordeaux.fr/ ges

avec les espaces vectoriels, et aussi avec les automates : Ces espaces permettent en particulier une formalisation rigoureuse de la notion d'indépendance linéaire de langages, (Meitus 1989). Le probleme est alors ramené à montrer la complétude d'un système axiomatique qui engendre certaines identités entre des séries rationnelles d'éterministes.

5.3 Automates à pile généralisés

L'idée est que la lecture de la pile ne se limite plus à une unique lettre du sommet. On donne un langage rationnel K, et l'automate est capable d'observer le plus long mot $k \in K$ qui soit préfixe du contenu de la pile, et de l'enlever. On ajoute donc de nouvelles transitions de la forme : $(q, a, K) \to (q', \epsilon)$. Notons pour l'exemple les automates à taquets qui sont un exemple d'automates à pile généralisés où $K = Y^*t$ avec Y un alphabet de pile, et t un symbole de taquet. On peut ensuite étendre le nombre de taquets, celui-ci fournissant une hiérarchie dans les langages déterministes.

Références

[Aut87] J.-M. Autebert *Langages algébriques* 1987. Masson.

[Wop91] P. Wolper *Introduction à la calculabilité* 1991. InterÉditions.

[Aut94] J.-M. Autebert *Théorie des langages et des automates* 1994. Masson.