911

Automates à pile. Exemples et Applications.

I_ Automates à pile et langages.

A_ Généralités

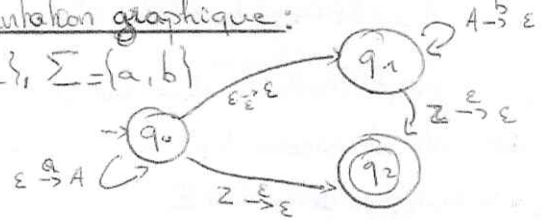
Def: Un automate à pile est un septuplet $(Q, \Sigma, \Gamma, \Delta, q_0, \gamma_0, F)$ où : - Q est l'ens des états
- Σ est l'alphabet d'entrée
- Γ est l'alphabet de pile
- Δ est l'ens des transitions $\subset Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times Q \times \Gamma^*$
- q_0 l'état initial
- γ_0 le symbole de fond de pile $\in \Gamma$
- $F \subset Q$ les états finaux.

Notations: soit $(q, a, X, q', w) \in \Delta$, on note $q.X \xrightarrow{a} q', w$

Ex et représentation graphique:

$Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$

$\Gamma = \{A, Z\}$



Def. Une config est un triplet $(q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$.
On définit une config initiale comme une config de la forme (q_0, w, γ_0).

Def: • Une config $(q', w, B\gamma)$ se dérive de $(q, aw, A\gamma)$ si $(q, w, A, q', B) \in \Delta$. on note $(q, aw, A\gamma) \vdash (q', w, B\gamma)$

• (q', w', γ') est dérivable en plusieurs étapes de (q, w, γ) si il existe $C_1, \ldots, C_k$ des configs tq $(q, w, \gamma) \vdash C_1 \vdash \cdots \vdash C_k \vdash (q', w', \gamma')$. On note $(q, w, \gamma) \vdash^* (q', w', \gamma')$

Def: On dit qu'un mot $w \in \Sigma^*$ est reconnu par pile vide ssi $(q, w, \gamma) \vdash^* (q', \varepsilon, \varepsilon)$.

On dit qu'il est reconnu par état acceptant si $(q, w, \gamma) \vdash^* (q', \varepsilon, \gamma')$ avec $q' \in F$

On note respectivement $N(A)$ et $L(A)$ l'ensemble des mots reconnus par pile vide et état acceptant par un automate à pile A.

Prop: - Soit $L = L(A)$, alors $L = N(A')$ pour un certain A'.
- Soit $L = N(A)$, alors $L = L(A')$ pour un certain A'.

dv I : le langage de pile est rat

B_ Langages algébriques

Def. Une grammaire algébrique est un quadruplet $G = (\Sigma, X, R, S)$ où : - Σ est l'alphabet terminal
- X est l'ensemble des symboles non-terminaux
- $R \subset X \times (\Sigma \cup X)^*$ sont les règles de production.
- $S \in X$ est le symbol initial.

Def. • $u \in (\Sigma \cup X)^*$ se dérive en v si $u = u_1 T u_2$ avec $T \in X$ et $v = u_1 m u_2$ tq $T \to m \in R$. on note $u \to v$.

• u se dérive en v en plusieurs étapes si il existe $v_1, \ldots, v_k$ tq $u \to v_1 \to \cdots \to v_k \to v$. On note $u \to^* v$

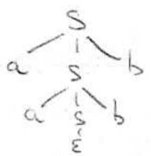
Def: u est engendré par G si $S \to^* u$.
$\in \Sigma^*$

On note $L(G)$ l'ensemble des mots engendrés par G.

Ex: $\left(\Sigma = \{a,b\},\ \{S\},\ \begin{array}{l} S \to aSb \\ S \to \varepsilon \end{array},\ S\right)$ engendre le langage $\{a^n b^n \mid n \in \mathbb{N}\}$.

Def: Un arbre de dérivation est un arbre tq :
- La racine est étiquetée par le symbole initiale.
- chaque noeud interne est étiqueté par un $T \in X$
- si un noeud interne est étiqueté par T, et si ses descendants directs sont étiquetés par $u_1, \dots, u_k$, alors $T \to u_1 \dots u_k \in R$.

Ex: $a^2 b^2$ a pour arbre de dérivation relativement à la grammaire de l'exemple précédent.



Def: Une grammaire est ambigüe si il existe deux arbres de dérivation distincts ayant la même frontière.

- Un langage est inhéremment ambigü si toutes les grammaires qui l'engendrent sont ambigües. On dit qu'il est non-ambigü sinon.

Th: L'ensemble des langages reconnus par un automate à pile est exactement l'ensemble des langages algébriques.

C. Automates à pile déterministes

Def: Un automate à pile est déterministe si :
- pour tout $(q,a,T) \in Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$, il y a au plus une transition de la forme (q, a, T, q', T')
- si il existe une transition (q, a, T, q', T') avec $a \in \Sigma$, alors il n'y a pas de transition $(q, \varepsilon, T, q', T')$.

Def: On dit qu'un langage L est déterministe si il peut être reconnu par un APD.

Def: Un langage L est préfixe si deux mots de L ne sont jamais préfixes l'un de l'autre.

Prop: Soit A un APD et $L = N(A)$, alors L est préfixe.

Prop: Le complémentaire d'un langage algébrique déterministe est encore algébrique déterministe.

Prop: Un langage déterministe est non ambigü.

II. Analyse syntaxique

En compilation, les langages de prog sont donnés par des grammaires algébriques. Pour retrouver la structure d'un prog lors de la compilation, il faut donc retrouver l'arbre de dérivation du programme : c'est le but de l'analyse syntaxique

dr2 (

Il existe deux approches de l'analyse syntaxique :

. L'analyse descendante qui consiste à partir du symbole initial et essayer de créer une dérivation du programme, et l'analyse ascendante, qui consiste à partir du programme et essayer de retrouver le symbole initial de la grammaire.

Nous verrons deux exemples illustrant chacune des deux approches : l'analyse LL(1) et l'analyse LR(0)

On se fixe désormais une grammaire $G = (\Sigma, X, R, S)$

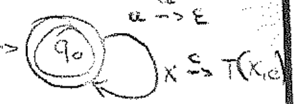
A – Analyse descendante

Idée : On construit une table T qui, étant donné un non-terminal T et le premier caractère de l'entrée, donne la règle $X \to \beta$ à appliquer. On suppose qu'un caractère # marque la fin de l'entrée.

On utilise une pile, qui contient le symbole initial S de la grammaire au début de l'analyse. A chaque étape, on regarde le sommet de la pile et la prochaine lettre c de l'entrée.

- si la pile est vide, le mot est accepté ssi $c = \#$
- si le sommet de la pile est un terminal a, on doit avoir $a = c$. Si c'est le cas, on dépile a et on consomme c. Sinon, le mot n'est pas engendré par la grammaire.
- si le sommet de la pile est un non-terminal T, on remplace X par β où $X \to \beta = T(X, c)$.

Le fonctionnement de cet algo est donc celui d'un automate à pile à un seul état.



Il reste à construire une telle table T lorsque cela est possible.

Lorsque l'on peut construire une telle table T, on dit que la grammaire est LL(1).

B – Analyse ascendante

C'est beau, il y a des automates "presque" à pile, mais je suis à court de temps ^.^

Références : . O. Carton
. Wolper
. Le dragon
. Le Autebert (Un peu ...).

Beaucoup de cours sur internet ; il serait pratiquement impossible de rencontrer un truc pas J. Pensez à utiliser google pendant vos récessions.

dv1 Handbook of formal languages vol 1

dv2 Carton mais c'est vraiment pas beau