Exemple d'analyse LR(0)

Mathias Millet

2014

Introduction

Analyse syntaxique : à partir d'un mot, on souhaite :

- savoir s'il appartien au language d'une grammaire donnée.
- reconstruire l'arbre syntaxique qui l'a engendré.

Définitions préliminaires et conventions : Soit $G = (V_N, V_T, P, S)$ une grammaire non-contextuelle, c'est à dire que l'on a :

- V_N est l'ensemble des symboles non-terminaux.
- V_T est l'ensemble des symboles terminaux.
- P est l'ensemble des productions de la grammaire, avec $P \subset V_N \times (V_N + V_T)^*$.
- $S \in V_N$ est l'axiome de la grammaire.

On utilisera les conventions suivantes :

- Les éléments de V_N seront représentés par les lettres $A, B, C, D, ...$ et S
- Les éléments de V_T seront représentés par les lettres $\mathrm{a}, \mathrm{b}, \mathrm{c}, ...$; les éléments de V_T^* par les lettres $\mathrm{u}, \mathrm{v}, \mathrm{w}, ...$
- Les éléments de $V_N \cup V_T$ seront représentés par les lettres $X, Y, ...$; les éléments de $(V_N \cup V_T)^*$ par les lettres $\alpha, \beta, ...$ Le mot vide sera représenté par ϵ.
- Un élément (A, α) de P sera noté $A \to \alpha$.

Analyse syntaxique ascendante

L'analyse ascendante (avec lecture de gauche à droite) fonctionne de la manière suivante :

- On garde en mémoire un mot α sur $V_N \cup V_T$, qui correspondra à la partie déjà reconnue.
- On peut alors
  - Soit lire un caractère supplémentaire du mot d'entrée, auquel cas on ajoute ce caractère à α
  - Soit identifier le suffixe de α à la partie droite δ d'une production de la grammaire $D \to \delta$, auquel cas on remplace δ par la partie gauche D associée.

Cela correspond à l'algorithme non-déterministe ci-dessous :

Algorithme 1 : Analyse ascendante()

Entrées : Grammaire G, mot w
$\alpha \longleftarrow$ ""
$i \longleftarrow 0$
Tant que $\alpha \neq S$ *et* $i \leq |w|$ **:**
  Choisir :
    soit *Lecture*
      $\alpha \longleftarrow \alpha \mathrm{w}_i$
      $i \longleftarrow i + 1$
    soit *Réduction*
      Choisir $(\delta_0, D_0, \beta_0) \in \{(\delta, D, \beta)$ tels que $\alpha = \beta\delta$ et $(D \to \delta) \in P\}$
      $\alpha \longleftarrow \beta_0 D_0$

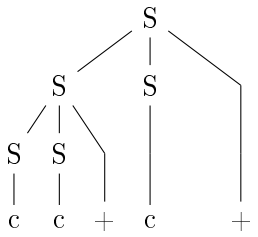
Exemple

Nous illustrerons ce développement à l'aide de la grammaire suivante :

$$G_0 = (\{S\}, \{c, +\}, \{S \to SS+, S \to c\}, S)$$

qui représente les expressions arithmétiques de l'opérateur + en notation post-fixée.

Figure 1: Arbre syntaxique correspondant au mot $cc+c+$



Automate des items

Définition (Item) Les items de la grammaire G sont les triplets (A, α, β) tels que $A \to \alpha\beta$ est une production de P. On notera $[A \to \alpha \bullet \beta]$ un tel item.
Un item $[A \to \alpha \bullet]$ sera dit complet.

À partir des items d'une grammaire, on considèrera l'automate non déterministe $AFN_G = (Q, V_N \cup V_T, \Delta, Q_0, F)$ suivant, appellé automate des items de la grammaire :

- Chaque état de Q correspond à un item de la grammaire.
- L'alphabet d'entrée est constitué des terminaux et des non-terminaux.
- La fonction de transition Δ est telle que
  - $\Delta([A \to \alpha \bullet B\beta], \epsilon) = \{[B \to \bullet\gamma]$ où $B \to \gamma$ est une production$\}$
  - $\Delta([A \to \alpha \bullet X\beta], X) = \{[A \to \alpha X \bullet \beta]\}$

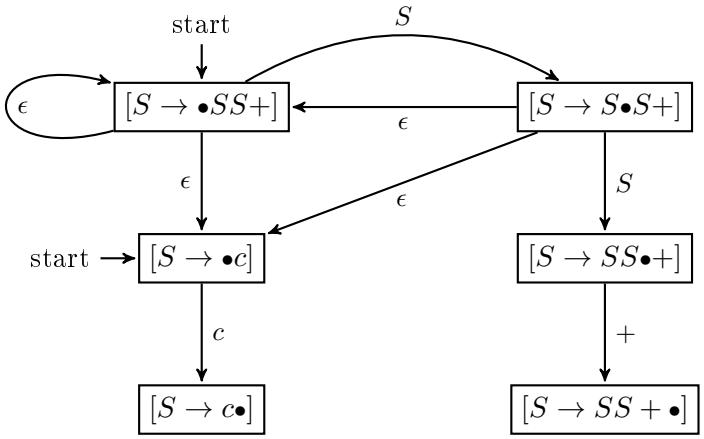


Figure 2: Automate des items de la grammaire G_0

- L'ensemble des états initiaux est $Q_0 = \{[S \to \bullet\alpha], S \to \alpha \in P\}$.
- Tous les états sont finaux : $F = Q$.

Théorème Soit $\gamma \in (V_N \cup V_T)^*$, on a, pour tout item $[A \to \alpha\bullet\beta]$ tel que α est un suffixe de γ (avec δ tel que $\gamma = \delta\alpha$):

$$[A \to \alpha\bullet\beta] \in \Delta(q_0, \delta\alpha) \quad \text{ssi} \quad \exists \text{w tels que } S \overset{*}{\Rightarrow}_d \delta A\text{w} \Rightarrow \delta\alpha\beta\text{w}$$

Autrement dit, la lecture d'un mot γ par l'automate nous amène dans les états correspondants aux items $[A \to \alpha\bullet\beta]$ où α est un suffixe de γ. Cela correspond au fait que l'on est en train d'essayer d'associer un suffixe de γ avec la partie droite de $(A \to \alpha\beta)$, et que α a déjà été reconnu. Si tout $\beta = \epsilon$, toute la partie droite de la règle est reconnue, et on pourra alors réduire en A, comme on le verra dans la partie suivante.

Idée de la démonstration

$\boxed{\Rightarrow}$ Induction sur la longueur du chemin.

$\boxed{\Leftarrow}$ On montre que $[A \to \bullet\alpha\beta] \in \Delta(q_0, \delta)$ par induction sur la longueur de la dérivation $S \overset{*}{\Rightarrow}_d \delta A\text{w} \Rightarrow_d \delta\alpha\beta\text{w}$

Automate de la grammaire

On a construit dans la section précédente un automate fini (non déterministe) reconnaissant les préfixes viables des proto-phrases de la grammaire (ce qui montre au passage que le language des préfixes viables est régulier). On va ici construire un automate à pile qui reconnaîtra exactement le langage de la grammaire.

Première étape On commence par déterminiser l'automate précédent. On obtient ainsi $AFD_G = (\bar{Q}, V_N \cup V_T, \bar{\Delta}, q_0)$. On notera $q_0, ..., q_n$ les éléments de $\bar{Q}$.

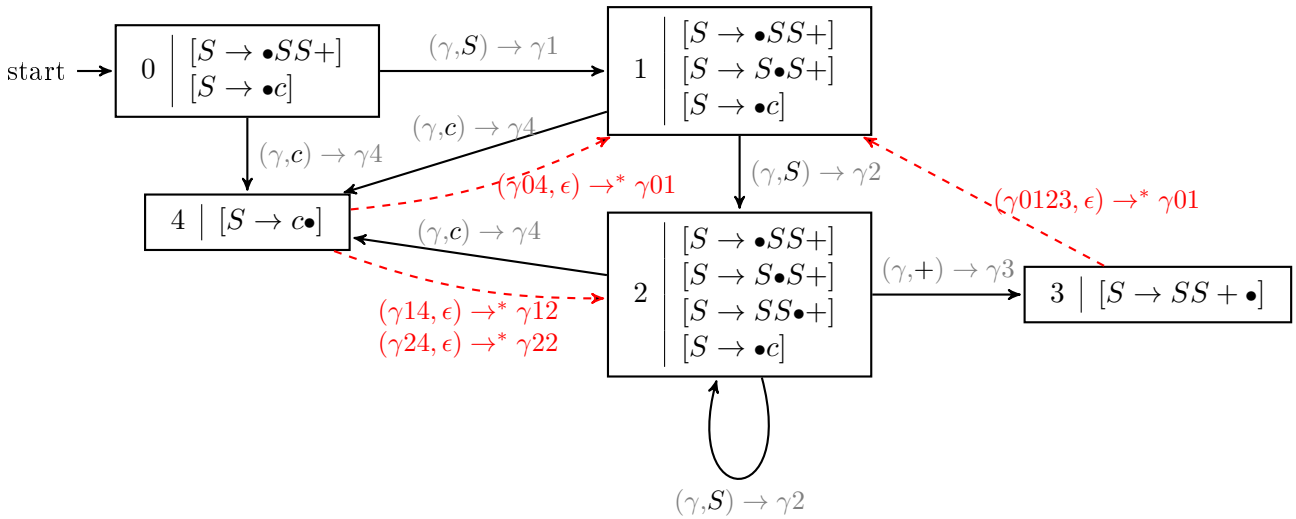


Figure 3: Automate à pile $\mathcal{A}_{G_0}$ correspondant à la grammaire G_0

Deuxième étape On ajoute les opérations de pile Λ suivantes:

- Pour toute transition $q_i \xrightarrow{a} q_j \in \bar{\Delta}$, $\quad (\gamma, q_i) \xrightarrow{a} (\gamma q_i, q_j) \in \Lambda$.

  Ces transitions, dites de lecture, servent à garder la trace des lectures effectuées, de manière à pouvoir associer au mot de pile un mot de $(V_N \cup V_T)^*$.

- Aux états contenant des items complets, on ajoute les transitions de pile suivantes :

  Si $[A \to \alpha\bullet] \in q_i$, alors il existe un préfixe viable γ tel que $S \overset{*}{\Rightarrow}_d \gamma A\mathrm{w} \Rightarrow_d \gamma\alpha\mathrm{w}$. Soit alors $q_j = \bar{\Delta}(\gamma A, q_0)$, on ajoute une série de transitions telles que$(\gamma\alpha, q_i) \xrightarrow{\epsilon}{}^* (\gamma, q_j)$ à Λ (on ne peut pas le faire en une seule transition car un automate à pile ne peut dépiler qu'un symbole par transition).

  Ces transitions, dites de réduction, correspondent à la reconnaissance de la phrase α comme étant la dérivation du symbole A.

Ainsi, A_G va lire son mot d'entrée jusqu'à se trouver dans un état contenant un item complet, auquel cas il pourra réduire. Si $\mathcal{A}_G$ a reconnu xy, sa pile contient exactement xy; s'il se trouve dans un état contenant $[A \to \mathrm{y}\bullet]$, on pourra réduire y en A, auquel cas la pile contiendra maintenant xA. On pourra alors continuer les lecture, ou réduire, jusqu'à épuiser l'entrée. Si alors le mot correspondant à l'état de la pile est exactement l'axiome, c'est à dire si l'on a réussi à réduire le mot d'entrée en l'axiome de la grammaire, alors il sera reconnu comme un élément de $L(G)$.

LR(0) Enfin, la grammaire G sera dite LR(O) si l'automate à pile obtenu est déterministe, c'est à dire qu'il n'y aura aucun conflit entre deux transitions de l'automate.

Les conflits peuvent être de deux ordres :

- *Lecture-réduction* Lorsque l'automate se trouve dans un état contenant un item complet $[A \to \alpha\bullet]$ et une transition par lecture d'un terminal $[B \to \beta\bullet b\beta'] \xrightarrow{b} [B \to \beta b\bullet\beta']$, si le caractère suivant du mot d'entrée est un b, alors l'automate devra faire un choix non-déterministe.

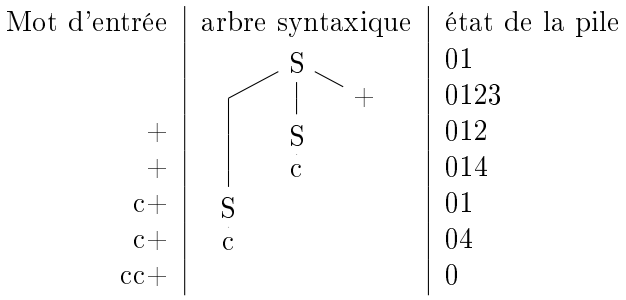


Figure 4: Reconnaissance du mot $cc+$ par l'automate $\mathcal{A}_{G_0}$.

- *Réduction-réduction* Lorsque l'automate se trouve dans un état contenant deux items complets, l'automate devra choisir de manière non déterministe entre les deux réductions.

Si aucun de ces cas ne se produit, ce qui peut se vérifier facilement, alors l'automate aura un fonctionnement déterministe. Dans ce cas, la construction de $\mathcal{A}_G$ nous donne une procédure effective pour savoir si G est $LR(0)$, et, le cas échéant, nous permet de reconnaître les mots de $L(G)$ de manière efficace.

Références

- J. E. Hopcroft, J. D. Ullman *Introduction to automata theory, languages and computation*

  Seule source (presque) sérieuse et (presque) claire trouvée pour traiter LR.

- Willhelm, Maurer *Compilers*

  Très formel, manque d'explications; traite LR de manière tordue; preuve au gros coup de bluff.

- *Dragon*

  Pas assez formel, mais fournit une partie de l'intuition.