ANALYSE LR(0)

Schwarzentruber / Legendre
Chap 5.

L'analyse LR(0) est une méthode d'analyse syntaxique, c-à-d dont le but est de déterminer si un texte dérive d'une grammaire, et dans ce cas de fournir un arbre de dérivation de frontière ce mot/texte.

Plus précisément il s'agit ici d'une analyse ascendante, c-à-d qu'on essaye de retrouver l'axiome à partir du mot en suivant des règles de réécritures inverses des dérivations, ce qu'on appelle des réductions.

Ce faisant on réalise une "right-most derivation", une dériva° la plus à droite. En effet on réécrit le plus à gauche possible d'abord, mais une fois remis dans le sens de la dérivation, cela fait qu'on dérive d'abord le plus à droite possible.
C'est ce que signifie le R

Le premier L signifie qu'on lit le texte une seule fois sans retour arrière et de gauche à droite "left to right"
Enfin le 0 indique qu'on n'a pas à lire de lettre pour résoudre des conflits, en fait il n'y a pas de conflits à résoudre ici

Cette analyse est une approche gloutonne que l'on réalise grâce à l'automate LR(0)

Bien sûr, de même qu'il y a des conditions sur la grammaire pour qu'on puisse y appliquer l'analyse LL(1), il y en a pour l'analyse LR(0), et elles s'expriment sur l'automate LR(0).

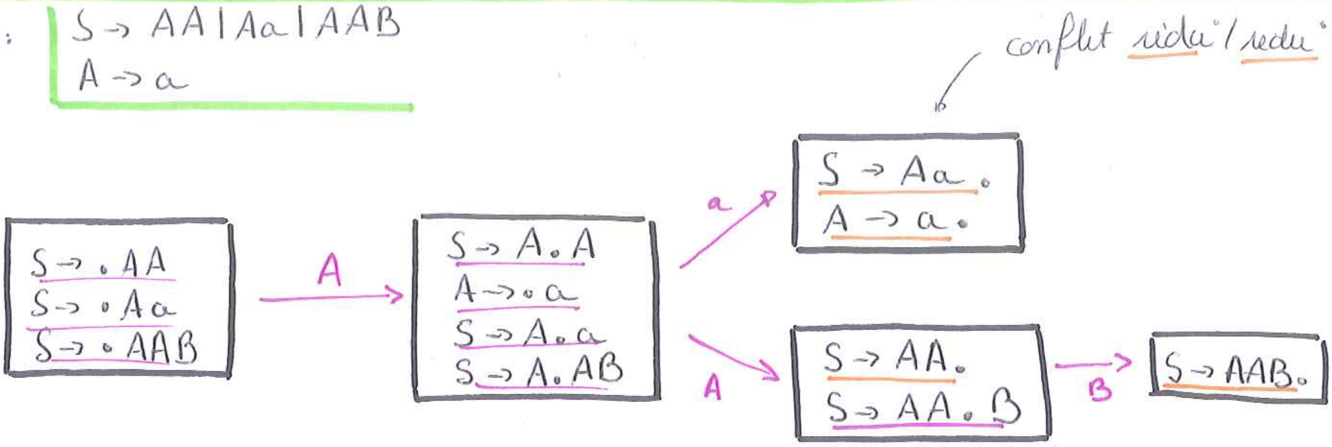
Déf p101.

G est une grammaire LR(0) si chaque état de son automate LR(0) qui possède un item de réduction est en fait réduit à cet item.

Cela assure qu'on saura sans aucun doute quand et comment réduire, pas de doute avec une autre réduc° possible (ie pas de conflit réduction/réduction) ni d'hésitations avec la lecture (ie pas de conflit lecture/réduction).

exc

G : S → AA | Aa | AAB
A → a

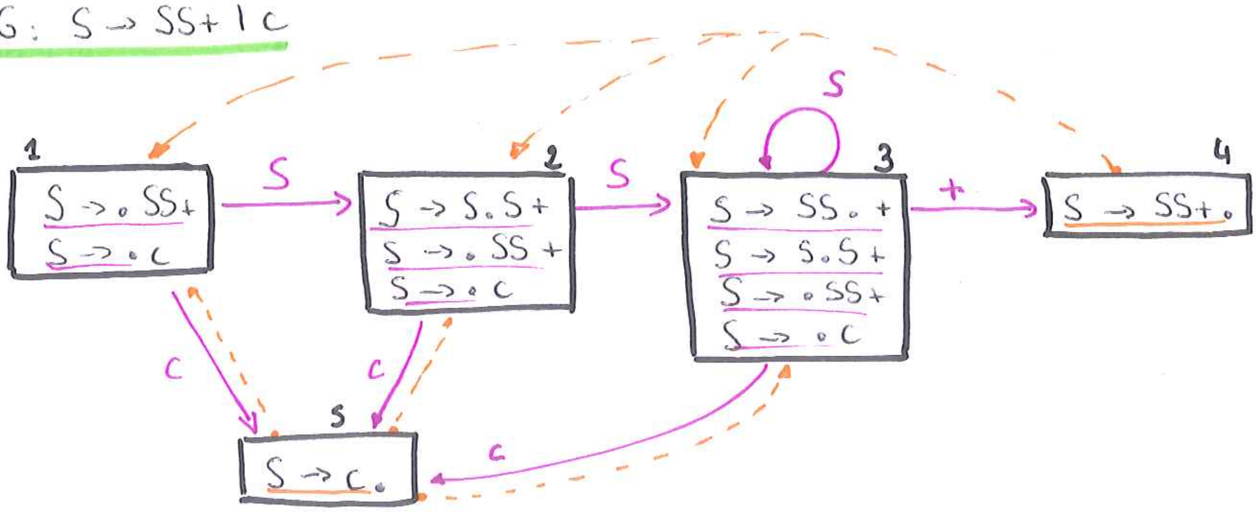


G n'est pas LR(0)

conflit lecture/réduit

exc

G : S → SS+ | c



Pas de conflit. G est LR(0)

Algo

Considérons G une grammaire LR(0) fixée, et son automate des items $\mathcal{A}$ d'état initial q_0, et de fonction transition δ.

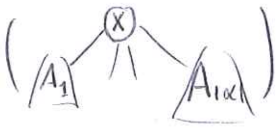
LR_0 (t)

```
Pa ← pile_vide        // pile d'arbres
Pe ← pile_vide        // pile d'états

Pe.empiler(q0) ; q ← q0
i ← 1
l ← |t|

tant que i ≤ l + 1
    si i = 1 et l = 1 et t1 = S
    alors break

    si q est un état de réduction
    alors soient X et α tels que q = [X → α•]
        Pour j allant de |α| à 1
            Aj ← Pa.dépiler()
            Pe.depiler()
        Pa.empiler( ... )
```



```
        q ← Pe.sommet_de_pile
        substituer α en t_{i-|α|} — t_i par X dans t
        i ← i-|α|
        l ← l -(|α|-1)

    si q est un état de lecture et i ≤ |t|
    alors q ← δ(q, t_i) ; Pe.empiler(q), i ← i+1
        Si t_i ∈ Σ
        alors Pa.empiler( [t_i] )

    si q est un état de lecture et i = l+1
    alors erreur

Si Pa = pile_vide
alors erreur
sinon A = Pa.depiler ;
    si Pa = pile vide et rac(A) = S
    alors retourner A
    sinon erreur
```

⚠ le i et le l sont des variables

⚠ il faut aller jusqu'à l+1 pour le cas d'une réduction finale où le "curseur" c-à-d i se trouve après la dernière lettre

Si $t_i \in \Sigma$
alors Pa . empiler (q)

ex. analyse de $t = 12+3+$ avec $G = S \to SS+ \mid c$ ↑ type nommé de l'analyse syntaxique

t : $\boxed{c \atop 1}\ \boxed{c \atop 2}\ \boxed{+}\ \boxed{c \atop 3}\ \boxed{+}$ après analyse syntaxique.

États

$\cdot\, 12+3+$ ① $\xrightarrow{c}$ $1\cdot 2+3+$ ⑤ a réduit $1 \to S$

$\cdot\, S\,2+3+$ ① $\xrightarrow{S}$ $S\cdot 2+3+$ ② $\xrightarrow{c}$ $S\,2\cdot +3+$ ⑤ b réduit $2 \to S$

$SS+3+$ ② $\xrightarrow{S}$ $SS\cdot +3+$ ③ $\xrightarrow{+}$ $SS+\cdot 3+$ ④ c réduit $SS+ \to S$

$\cdot\, S\,3+$ ① $\xrightarrow{S}$ $S\cdot 3+$ ② $\xrightarrow{c}$ $S\,3\cdot +$ ⑤ d réduit $3 \to S$

$S\cdot S+$ ② $\xrightarrow{S}$ $SS\cdot +$ ③ $\xrightarrow{+}$ $SS+\cdot$ ④ e réduit $SS+ \to S$

$\cdot\, S$ ①

— = réduction
— lecture

dériva° associée

$S \overset{e}{\Leftarrow} SS+ \overset{d}{\Leftarrow} S3+ \overset{c}{\Leftarrow} SS+3+ \overset{b}{\Leftarrow} S2+3+ \overset{a}{\Leftarrow} 12+3+$

pile d'arbre