IFIPS – Cycle Ingénieur 2[e] année Informatique
Compilation 2006–2007

Travaux Dirigés n^{o}3
Analyse ascendante (corrigé)

1 Séquence avec séparateur

Fait en TD.

2 Différence entre LL(1) et SLR(1)

Soit la grammaire :

S	:=	AaAb | BbBa
A	:=	ϵ
B	:=	ϵ

Montrer que cette grammaire est LL(1) mais pas SLR(1).

Correction : *On a* $\textsc{prem}(AaAb) = \{a\}, \textsc{prem}(BbBa) = \{b\}, \textsc{suiv}(A) = \textsc{suiv}(B) = \{a, b\}$ *La grammaire est donc LL(1). La table de transitions est la suivante :*

	a	b	$\sharp$
S	$AaAb$	$BbBa$	
A	ϵ	ϵ	
B	ϵ	ϵ	

Pour l'analyse ascendante, on construit l'automate LR(0).

L'état initial est : état1 : $[S' \to .S\sharp]$ *Il y a conflit entre les deux réductions :* $A{:=}\ \epsilon$ *et* $B{:=}\ \epsilon$

$[S \to .AaAb]$
$[S \to .BbBa]$
$[A \to .]$
$[B \to .]$

or les deux non-terminaux A et B ont les mêmes caractères suivants. Donc la grammaire n'est pas SLR(1).

3 Notation post-fixée

On se donne la grammaire suivante pour reconnaître des expressions arithmétiques écrites de manière post-fixée. L'ensemble des terminaux est {`num`,`op`} et la grammaire possède un seul non-terminal E et les règles suivantes :

E ::= `num`
E ::= E E `op`

1. Les expressions suivantes appartiennent-elles au langage engendré par cette grammaire ?
   (a) `num num num op op`
   (b) `num num op num op op`

   Correction : *On a* $E \Rightarrow E\ E\ \mathtt{op} \Rightarrow \mathtt{num}\ E\ \mathtt{op} \Rightarrow \mathtt{num}\ E\ E\ \mathtt{op}\ \mathtt{op} \Rightarrow \mathtt{num}\ \mathtt{num}\ E\ \mathtt{op}\ \mathtt{op} \Rightarrow \mathtt{num}\ \mathtt{num}\ \mathtt{num}\ \mathtt{op}\ \mathtt{op}$ *donc* `num num num op op` *est reconnaissable.*
   On peut montrer par récurrence sur la longueur de la dérivation que dans toute expression reconnue, le nombre de `num` *est égale au nombre de* `op`*+1. Donc le mot* `num op num num op op` *qui a le même nombre de* `op` *et de* `num` *n'appartient pas au langage engendré. Une alternative est de montrer que tous les mots reconnus ont un nombre impair de symboles terminaux.*

2. La grammaire est-elle LL(1) ? LR(0) ? SLR(1) ? Dans le cas d'une réponse positive, construire la table de transitions correspondante.

   Correction : *La grammaire est récursive gauche donc n'est pas LL(1). L'automate d'analyse LR(0) a les états suivants :*

$$s_0 : \boxed{\begin{array}{l}[S \rightarrow .E\sharp] \\ [E \rightarrow .\mathtt{num}] \\ [E \rightarrow .E\ E\ \mathtt{op}]\end{array}} \quad s_1 : \boxed{\begin{array}{l}[S \rightarrow E.\sharp] \\ [E \rightarrow E.E\ \mathtt{op}] \\ [E \rightarrow .\mathtt{num}] \\ [E \rightarrow .E\ E\ \mathtt{op}]\end{array}} \quad s_2 : \boxed{[E \rightarrow \mathtt{num}.]} \quad s_3 : \boxed{\begin{array}{l}[E \rightarrow E\ E.\ \mathtt{op}] \\ [E \rightarrow E.E\ \mathtt{op}] \\ [E \rightarrow .\mathtt{num}] \\ [E \rightarrow .E\ E\ \mathtt{op}]\end{array}} \quad s_4 : \boxed{[E \rightarrow E\ E\ \mathtt{op}.]}$$

   Les états où les réductions sont possibles sont l'état s_2 *et l'état* s_4*. Ces deux états ne présentent pas de conflit LR(0). La grammaire est donc LR(0) et par conséquent SLR(1). Tous les symboles terminaux sont des suivants de E. La table de transition est la suivante :*

	num	op	♯	E
s_0	s_2			s_1
s_1	s_2		succès	s_3
s_2	red E:= num			
s_3	s_2	s_4		s_3
s_4	red E:= E E op			

3. On suppose que l'on veut traiter également des opérateurs unaires, pour cela on ajoute la règle E:= E `op`. La nouvelle grammaire ainsi obtenue est-elle LL(1) ? SLR(1) ? LR(1) ?

   Correction : *Si on ajoute la règle E:= E* `op`*, alors la grammaire devient ambiguë, en effet* `num num op op` *admet deux dérivations gauches :* $E \Rightarrow E\ E\ \mathtt{op} \Rightarrow \mathtt{num}\ E\ \mathtt{op} \Rightarrow \mathtt{num}\ E\ \mathtt{op}\ \mathtt{op} \Rightarrow \mathtt{num}\ \mathtt{num}\ \mathtt{op}\ \mathtt{op}$ *et* $E \Rightarrow E\ \mathtt{op} \Rightarrow E\ E\ \mathtt{op}\ \mathtt{op} \Rightarrow \mathtt{num}\ E\ \mathtt{op}\ \mathtt{op} \Rightarrow \mathtt{num}\ \mathtt{num}\ \mathtt{op}\ \mathtt{op}$
   Une grammaire ambiguë n'est ni LL(1), ni SLR(1) ni LR(1).