Université de Versailles Saint-Quentin-en-Yvelines — première session, janvier 2015
Master 1 Info
Compilation
L. PILARD, L. RODIER et D. SOHIER
Durée : 2h30

Examen terminal

*Aucun document ni dispositif électronique **(y compris téléphone portable)** n'est autorisé. Toute réponse doit être soigneusement justifiée.*

Exercice

On considère la grammaire ci-dessous, qui décrit des listes :

$$\begin{array}{rl} S \rightarrow & \text{(L)} \\ L \rightarrow & \text{L, L} \\ \mid & \texttt{id} \end{array}$$

1. Levez l'ambiguïté de cette grammaire (la virgule est associative).
2. Supprimez sa récursivité gauche, et effectuez des factorisations si nécessaire.
3. Calculez sa table LL(1) ou son analyseur syntaxique récursif descendant.
4. Analysez le texte `(id, id, id)`.

Problème

Les questions de ce problème sont indépendantes.

On considère la grammaire décorée suivante :

$$\begin{array}{rll}
I \rightarrow & \texttt{essai}\ L'\ \texttt{recuperation}\ N\ L\ \texttt{fin} & \{\ \texttt{backtrack(}L'\texttt{.liste, }N\texttt{.quad);} \\
 & & \quad \texttt{backtrack(}N\texttt{.quad-1, nextquad())}\} \\
\mid & \texttt{id :=}\ E & \{\ \texttt{gen(:=, }E\texttt{.place, , @id.lex)}\} \\
 & & \\
L' \rightarrow & I\ L' & \\
\mid & \texttt{probleme(}B\texttt{)}\ M\ L' & \{\ \texttt{backtrack(}B\texttt{.faux, M.quad);} \\
 & & \quad L'\texttt{.liste:=}L'^{(1)}\texttt{.liste} \cup B\texttt{.vrai}\} \\
\mid & \varepsilon & \{\ L'\texttt{.liste=()}\} \\
 & & \\
L \rightarrow & I\ L & \\
\mid & \varepsilon & \\
 & & \\
N \rightarrow & \varepsilon & \{\ \texttt{gen(goto, , , ?);}N\texttt{.quad:=nextquad()}\} \\
M \rightarrow & \varepsilon & \{\ M\texttt{.quad:=nextquad()}\}
\end{array}$$

Le comportement de cette structure algorithmique est le suivant : les instructions du bloc `essai` sont exécutées séquentiellement. Lorsqu'une instruction `probleme` est rencontrée, le booléen passé en argument est évalué. S'il est faux, on poursuit l'exécution du bloc. Sinon, on interrompt l'exécution du bloc et on exécute le bloc `recuperation`. A la fin du bloc `essai`, on passe à l'instruction située après le bloc `recuperation`.

Dans la table ci-dessous, `probleme` est abrégé en `pb` et `recuperation` en `recup`.

	essai	id	:=	E	pb(	B	)	recup	fin	$\dashv$	I	L	L'	N	M
0	d2	d3									1				
1										acc					
2	d2	d3			d6			r5			5		4		
3			d7												
4								d8							
5	d2	d3			d6			r5			5		9		
6						d10									
7				d11											
8	r8	r8							r8					12	
9								r3							
10							d13								
11	r2	r2			r2			r2	r2	r2					
12	d2	d3							r7		15	14			
13	r9	r9			r9			r9							16
14									d17						
15	d2	d3							r7		15	18			
16	d2	d3						r5			5		19		
17	r1	r1			r1			r1	r1	r1					
18									r6						
19								r4							

1. Faites une représentation graphique des différents blocs, et ajoutez-y les instructions générées par la grammaire décorée ci-dessus. Le bloc L' a un attribut `liste` qui désigne une liste d'adresses à remplir, du même type que les attributs `vrai` et `faux` des booléens : on pourra adopter le même type de représentation.
2. Calculez les 10 premiers items LR(0) de cette grammaire.
3. Résolvez les conflits présents dans la table d'analyse des expressions booléennes (fournie en annexe ; on donne la priorité au $\wedge$ sur le $\vee$; le $\neg$ est le plus prioritaire des opérateurs).
4. Complétez les lignes 5, 7 et 10 de la table SLR des expressions arithmétiques (fournie en annexe avec les états LR(0) de cette grammaire).
5. Exécutez l'analyseur LR des expressions arithmétiques sur les séquences `0`, `-1`, `1/x`, et `z/(x+1)` (vous donnerez la pile et le flot d'entrée à chaque étape, et préciserez les quadruplets générés au fur et à mesure ; on supposera que -1 est analysé comme un entier par l'analyseur lexical).
6. Exécutez l'analyseur LR des expressions booléennes sur les séquences `x=0` et `x=-1` (vous donnerez la pile et le flot d'entrée à chaque étape, et préciserez les quadruplets générés au fur et à mesure).
7. Exécutez l'analyseur SLR sur le programme ci-dessous (vous donnerez la pile et le flot d'entrée à chaque étape, et préciserez les quadruplets générés et/ou complétés au fur et à mesure).

```
essai
    probleme(x=0)
    z:=1/x
```

```
    probleme(x=-1)
    z:=z/(x+1)
recuperation
    z:=0
fin
```

`z` sera analysé par l'analyseur lexical comme des `id` (`z` comme valeur associée). `1/x`, `z/(x+1)`, `0` seront reconnus comme des E : lorsque vous atteindrez ces séquences lors de l'analyse, vous décalerez directement sur E et générerez le code calculé à la question précédente. `x=0` et `x=-1` seront reconnus comme des B : lorsque vous atteindrez ces séquences lors de l'analyse, vous décalerez directement sur B et générerez le code calculé à la question précédente.

8. Décrivez l'exécution du code généré (quadruplets exécutés, valeurs des variables) dans le cas où `x` vaut initialement 1, puis dans le cas où il vaut -1.

Annexe — grammaire et table SLR pour les expressions arithmétiques

$$
\begin{array}{rlll}
E \to & E + E & \{E.place\texttt{=newtmp();gen(+, } E^{(1)}.place\texttt{, } E^{(2)}.place\texttt{, } E.place\texttt{)}\} \\
\mid & E/E & \{E.place\texttt{=newtmp();gen(/, } E^{(1)}.place\texttt{, } E^{(2)}.place\texttt{, } E.place\texttt{)}\} \\
\mid & (E) & \{E.place = E^{(1)}.place\} \\
\mid & \texttt{id} & \{E.place\texttt{=@id}.lex\} \\
\mid & \texttt{cst} & \{E.place\texttt{=newtmp();gen(cst, cst.}val\texttt{, , } E.place\texttt{)}\}
\end{array}
$$

	+	/	(	)	id	cst	⊣	E
0			d2		d3	d4		1
1	d5	d6					acc	
2			d2		d3	d4		7
3	r4	r4		r4			r4	
4	r5	r5		r5			r5	
5								
6			d2		d3	d4		9
7								
8	r1	d6		r1			r1	
9	r2	r2		r2			r2	
10								

$I_0 = \{E' \to .E \dashv, E \to .E + E, E \to .E/E, E \to .(E), E \to .\texttt{id}, E \to .\texttt{cst}\}$
$I_1 = \{E' \to E. \dashv, E \to E. + E, E \to E./E\}$
$I_2 = \{E \to (.E), E \to .E + E, E \to .E/E, E \to .(E), E \to .\texttt{id}, E \to .\texttt{cst}\}$
$I_3 = \{E \to \texttt{id}.\}$
$I_4 = \{E \to \texttt{cst}.\}$
$I_5 = \{E \to E + .E, E \to .E + E, E \to .E/E, E \to .(E), E \to .\texttt{id}, E \to .\texttt{cst}\}$
$I_6 = \{E \to E/.E, E \to .E + E, E \to .E/E, E \to .(E), E \to .\texttt{id}, E \to .\texttt{cst}\}$
$I_7 = \{E \to (E.), E \to E. + E, E \to E./E\}$
$I_8 = \{E \to E + E., E \to E. + E, E \to E./E\}$
$I_9 = \{E \to E/E., E \to E. + E, E \to E./E\}$
$I_{10} = \{E \to (E).\}$

Annexe — grammaire et table SLR pour les expressions booléennes

$B \to$ $B \vee N\ B$ $\{B.vrai$=$B^{(1)}.vrai \cup B^{(2)}.vrai\,;\ B.faux$=$B^{(2)}.faux\,;$ `backtrack`$(B^{(1)}.faux, N.quad)\}$

| $B \wedge N\ B$ $\{B.faux$=$B^{(1)}.faux \cup B^{(2)}.faux\,;\ B.vrai = B^{(2)}.vrai\,;$ `backtrack`$(B^{(1)}.vrai, N.quad)\}$

| $\neg B$ $\{B.vrai$=$B^{(1)}.faux\,;\ B.faux = B^{(1)}.vrai\}$

| (B) $\{B.vrai$=$B^{(1)}.vrai\,;\ B.faux = B^{(1)}.faux\}$

| E `op` E $\{B.vrai$=(`nextquad()`)`;` `gen(op`$.lex$`,`$E^{(1)}.place$`,`$E^{(2)}.place$`,?)` `;` $B.faux$=(`nextquad()`)`;gen(goto,,?)`$\}$

$N \to$ ε $\{N.quad$:=`nextquad()`$\}$

`op` désigne un opérateur de comparaison (=, $<$, $>$, $\leq$, $\geq$), et `op`$.lex$ est la chaîne correspondant à cet opérateur.

	$\vee$	$\wedge$	$\neg$	(	)	`op`	E	$\dashv$	B	N
0			d2	d3			d4		1	
1	d5	d6						acc		
2			d2	d3			d4		7	
3			d2	d3			d4		8	
4						d9				
5			r6	r6			r6			10
6			r6	r6			r6			11
7	d5/r3	d6/r3			r3			r3		
8	d5	d6			d12					
9							d13			
10			d2	d3			d4		14	
11			d2	d3			d4		15	
12	r4	r4			r4			r4		
13	r5	r5			r5			r5		
14	d5/r1	d6/r1			r1			r1		
15	d5/r2	d6/r2			r2			r2		

$I_5 = \{B \to B \vee .NB, N \to \varepsilon.\}$
$I_6 = \{B \to B \wedge .NB, N \to \varepsilon.\}$
$I_7 = \{B \to \neg B., B \to B. \vee NB, B \to B. \wedge NB\}$
$I_{14} = \{B \to B \vee NB., B \to B. \vee NB, B \to B. \wedge NB\}$
$I_{15} = \{B \to B \wedge NB., B \to B. \vee NB, B \to B. \wedge NB\}$