Mardi 9/03/2010
Durée : 2 heures
Documents autorisés

Traduction et sémantique
Licence Informatique 3ème année

Attention : donnez autant de détails que vous jugez nécessaire ; des simples copies-collers-miroirs de vos notes de cours ne seront pas jugés satisfaisants. En bref, montrez que vous avez compris.

Examen Partiel

Analyse lexicale

Exercice 1. Un fichier `csv` est une suite de ligne, chaque ligne étant un suite de champs séparées par une virgule. Voici un exemple :

```
CLIENT,DATE,QUANTITE,PRIX UNITAIRE,PRIX TOT

Remi, 12/11/2009,2,20,60
Luigi,21/1/2010,3,12,48
Walid, B.,11,2,22
```

Écrivez, dans le langage lex, un analyseur lexical qui affichera toutes et seules les lignes d'un fichier csv dont le deuxième champ contient une date de la forme JOUR/MOIS/AN. Si MOIS est janvier, alors l'analyseur ajoutera un dernier champ contenant le mot « solde ». Par exemple, l'affichage de l'analyseur sur le fichier ci-dessus sera le suivant :

```
Remi, 12/11/2009,2,20,60
Luigi,21/1/2010,3,12,48,solde
```

Solution.

```
%option main

JOUR 0?[1-9]|[12][0-9]|3[01]
MOIS 0?[1-9]|1[0-2]
AN [0-9]{0,2}[0-9][0-9]

JANVIER 0?1

DATE {JOUR}\/{MOIS}\/{AN}
DATESOLDE {JOUR}\/{JANVIER}\/{AN}

CHAMP [^,\n]*
CHAMPS ({CHAMP},)*{CHAMP}?

BLANCS [ \t]*
%%

{CHAMP},{BLANCS}{DATESOLDE}{BLANCS}(,{CHAMPS})? {printf("%s,solde\n",yytext);}
{CHAMP},{BLANCS}{DATE}{BLANCS}(,{CHAMPS})? {puts(yytext);}

.|\n {}

%%
```

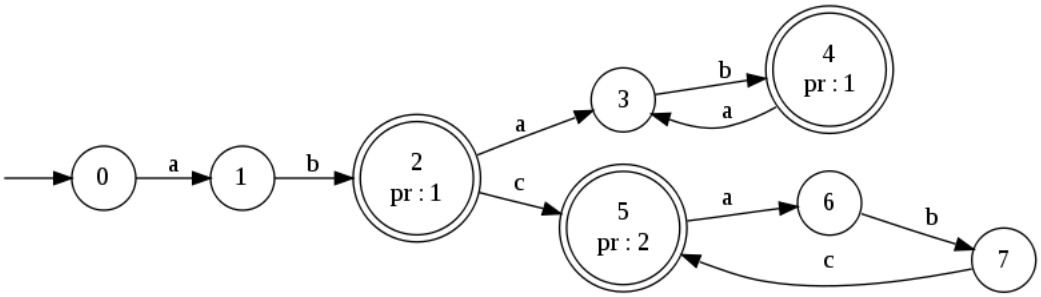
□

Exercice 2. Considerez le fichier lex suivant :

```
%option main
%%
```

```
(ab)+ {printf("Action 1");}
(abc)+ {printf("Action 2");}
%%
```

Voici un AFD qui reconnait le langage denoté par l'expression reguliere `(ab)+|(abc)+` :



Remarquez que chaque état final de ce AFD possède une priorité choisie parmi 1 et 2, selon l'action à déclencher.

En utilisant les pointeurs `yytext`, `fc`, `cc`, simulez le comportement de l'analyseur lexical sur la chaîne de caractères

`abcdababc`

Solution.

Pointeurs	État	Priorité
$\uparrow\uparrow\uparrow$ *abcdababc*	0	1
$\uparrow\uparrow$ *a* $\uparrow$ *bcdababc*	1	1
$\uparrow$ *ab* $\uparrow\uparrow$ *cdababc*	2	1
$\uparrow$ *abc* $\uparrow\uparrow$ *dababc*	3	2
$\uparrow$ *abc* $\uparrow$ *d* $\uparrow$ *ababc*	*err*	2
déclencher action 2		
abc $\uparrow\uparrow\uparrow$ *dababc*	0	1
abc $\uparrow\uparrow$ *d* $\uparrow$ *ababc*	*err*	1
abcd $\uparrow\uparrow\uparrow$ *ababc*	0	1
abcd $\uparrow\uparrow$ *a* $\uparrow$ *babc*	1	1
abcd $\uparrow$ *ab* $\uparrow\uparrow$ *abc*	2	1
abcd $\uparrow$ *ab* $\uparrow$ *a* $\uparrow$ *bc*	3	1
abcd $\uparrow$ *abab* $\uparrow\uparrow$ *c*	4	1
abcd $\uparrow$ *abab* $\uparrow$ *c* $\uparrow$	*err*	1
déclencher action 1		
abcdabab $\uparrow\uparrow\uparrow$ *c*	0	1
abcdabab $\uparrow\uparrow$ *c* $\uparrow$	*err*	1
abcdababc $\uparrow\uparrow\uparrow$	*err*	1
fin de l'entrée		

□

Analyse syntaxique

Exercice 3. Est ce que la grammaire

$$S \rightarrow Sa \mid b$$

est LL(1) ? Justifiez votre réponse.

Solution. Cette grammaire n'est pas LL(1). Nous avons $NULL = \emptyset$ et $First(S) = \{ b \}$. Par conséquent

$$First(Sa) \cap First(b) = \{ b \} \cap \{ b \} \neq \emptyset .$$

Car Sa et b sont la partie droite de productions ayant le même non-terminal à gauche (ici S), nous remarquons qu'une des conditions définissant les grammaires LL(1) n'est pas satisfaite. □

Exercice 4. Considérez cette grammaire :

$$\begin{aligned} S &\rightarrow TT \\ T &\rightarrow aTb \mid \epsilon \end{aligned}$$

1. Calculez NULL. Listez les contraintes permettant de calculer FIRST et FOLLOW, et ensuite calculez FIRST et FOLLOW.

   Solution. Évidemment, $T \in NULL$, car $T \rightarrow \epsilon$. Aussi $S \in NULL$, car $S \rightarrow TT$ et $T \in NULL$. On a donc

   $$NULL = \{ S, T \}.$$

   Les contraintes pour FIRST :

   $$FIRST(S) \supseteq FIRST(T) \qquad FIRST(T) \supseteq \{ a \}$$

   Les contraintes pour FOLLOW :

   $$FOLLOW(S) \supseteq \emptyset \qquad FOLLOW(T) \supseteq \{ b \} \cup FIRST(T) \cup FOLLOW(S) = \{ a, b \} \cup FOLLOW(S).$$

   La plus petite solution des contraintes ci-dessus donne :

   $$\begin{aligned} FIRST(S) &= \{ a \} & FIRST(T) &= \{ a \} \\ FOLLOW(S) &= \emptyset & FOLLOW(T) &= \{ a, b \}. \end{aligned}$$

   □

2. Est ce que la grammaire est LL(1) ? Justifiez votre réponse.

   Solution. Non. Nous avons $FIRST(aTb) = \{ a \}$ et $FIRST(\epsilon) = \emptyset$. On a donc, pour les productions $T \rightarrow aTb \mid \epsilon$,

   $$FIRST(aTb) \cap FIRST(\epsilon) = \emptyset.$$

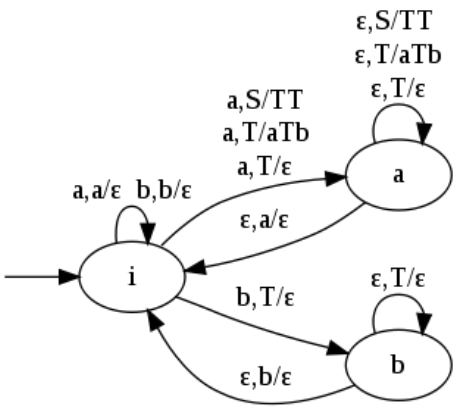
   D'ailleurs, nous avons

   $$\begin{aligned} FIRST(TT) \cap FOLLOW(S) &= \emptyset, \\ FIRST(aTb) \cap FOLLOW(T) &= \{ a \} \cap \{ a, b \} = \{ a \}, \\ FIRST(\epsilon) \cap FOLLOW(T) &= \emptyset. \end{aligned} \tag{X}$$

   La relation (X) montre que cette grammaire n'est pas $LL(1)$. □

3. Esquissez l'APD qui accepte le langage de la grammaire.

   Solution. Il ne s'agit pas donc d'un APD, mais plutôt d'un APN.



   □

4. A l'aide de cet APD, montrez que le mot *abaabb* appartient au langage de la grammaire.

Solution.

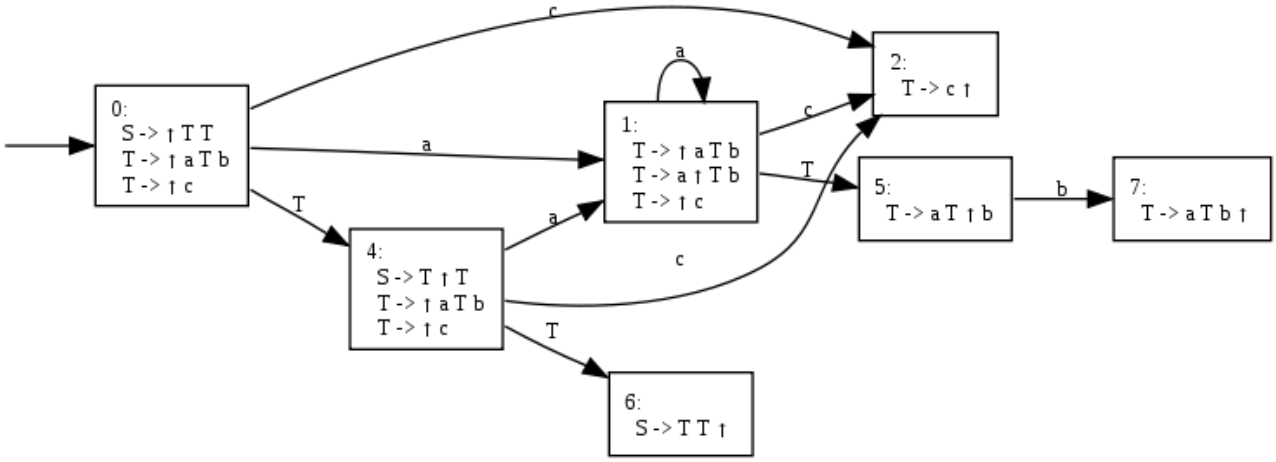
$$(i, abaabb, S) \vdash (a, baabb, TT) \vdash (a, baabb, aTbT) \vdash (i, baabb, TbT) \vdash (b, aabb, bT)$$
$$\vdash (i, aabb, T) \vdash (a, abb, aTb) \vdash (i, abb, Tb) \vdash (a, bb, aTbb)$$
$$\vdash (i, bb, Tbb) \vdash (b, b, bb) \vdash (i, b, b) \vdash (i, ,)$$

□

Exercice 5. Considérez la grammaire

$$S \rightarrow TT$$
$$T \rightarrow aTb \mid c$$

Voici l'AFD des items :



1. Est ce que la grammaire est $LR(0)$? Justifiez votre réponse.

   Solution. La grammaire est bien sur $LR(0)$ car
   – l'axiome n'apparaît pas dans la partie droite d'une production,
   – tout macro état contenant un item complet est un singleton.

   □

2. Simulez le calcul d'un analyseur ascendant $LR(0)$ sur les entrées suivantes :

   $$caacbb, \quad acbacb.$$

3. Pour chaque mot $caacbb, acbacb$, écrivez une dérivation droite du mot témoin de l'appartenance du mot au langage de grammaire. Annotez cette dérivation et le calcul de l'analyseur $LR(0)$, de façon à mettre en correspondance les réductions de l'analyseur ascendant avec les productions utilisés dans la dérivation.

Solution. Pour le mot $caacbb$:

Pile	Entrée	Action	
0	$caacbb$	$shift$	
$0c2$	$aacbb$	$reduce$	$T \rightarrow c$ (1)
$0T4$	$aacbb$	$shift$	
$0T4a1$	$acbb$	$shift$	
$0T4a1a1$	cbb	$shift$	
$0T4a1a1c2$	bb	$reduce$	$T \rightarrow c$ (2)
$0T4a1a1T5$	bb	$shift$	
$0T4a1a1T5b7$	b	$reduce$	$T \rightarrow aTb$ (3)
$0T4a1T5$	b	$shift$	
$0T4a1T5b7$		$reduce$	$T \rightarrow aTb$ (4)
$0T4T6$		$reduce$	$S \rightarrow TT$ (5)
$0S$		$accept$	

La dérivation droite est :

$$S \Rightarrow^{(5)} TT \Rightarrow^{(4)} TaTb \Rightarrow^{(3)} TaaTbb \Rightarrow^{(2)} Taacbb \Rightarrow^{(1)} caacbb$$

Pile	Entrée	Action	
0	$acbacb$	$shift$	
$0a1$	$cbacb$	$shift$	
$0a1c2$	$bacb$	$reduce$	$T \rightarrow c$ (1)
$0a1T5$	$bacb$	$shift$	
$0a1T5b7$	acb	$reduce$	$T \rightarrow aTb$ (2)
$0T4$	acb	$shift$	
$0T4a1$	cb	$shift$	
$0T4a1c2$	b	$reduce$	$T \rightarrow c$ (3)
$0T4a1T5$	b	$shift$	
$0T4a1T5b7$		$reduce$	$T \rightarrow aTb$ (4)
$0T4T6$		$reduce$	$S \rightarrow TT$ (5)
$0S$		$accept$	

La dérivation droite est :

$$S \Rightarrow^{(5)} TT \Rightarrow^{(4)} TaTb \Rightarrow^{(3)} Tacb \Rightarrow^{(2)} aTbacb \Rightarrow^{(1)} acbacb$$

□