Linguaggi di Programmazione e Compilatori

III° Appello del 9/9/2004

Scrivere **in stampatello** COGNOME e NOME su ogni foglio. La brutta copia va consegnata (indicare che si tratta della brutta), il testo no.
NOTA: Nelle espressioni regolari si possono usare le usuali convenzioni di precedenza: l'operatore * lega più della concatenazione che, a sua volta, lega più dell'operatore $|$. Inoltre si può usare l'abbreviazione $+$ con il solito significato.

ESERCIZIO 1 (7 punti)

Si scriva una definizione regolare che definisca una categoria lessicale **commands** che denoti il linguaggio così definito:

- Ogni comando inizia con il carattere \ o con il carattere !
- Dopo il carattere di inizio deve essere inserito, senza spazi intermedi, il nome del comando che deve iniziare per lettera e può proseguire con lettere o cifre
- Il nome del comando può essere seguito da un parametro opzionale che deve essere inserito fra parentesi graffe dopo il nome del comando senza spazi intermedi. Il parametro può essere un qualunque numero intero che non comincia per zero (e quindi non può essere nemmeno zero).

Esempi di comandi validi: `\com1, !c, \fun{67}`. Suggerimento: si ricordi che nelle definizioni regolari possono essere definiti dei nomi per denotare espressioni regolari di comodo (es. **letter**).

SOLUZIONE

$$
\begin{array}{lcl}
\textbf{letter} & \rightarrow & a \mid b \mid \cdots \mid z \mid A \mid \cdots \mid Z \\
\textbf{digit} & \rightarrow & 0 \mid 1 \mid \cdots \mid 9 \\
\textbf{nonzerodigit} & \rightarrow & 1 \mid 2 \mid \cdots \mid 9 \\
\textbf{opt\_parameter} & \rightarrow & \{\textbf{nonzerodigit}\,\textbf{digit}^*\} \\
\textbf{commands} & \rightarrow & \textbf{letter}(\textbf{letter} \mid \textbf{digit})^*\,(\textbf{opt\_parameter} \mid \epsilon)
\end{array}
$$

ESERCIZIO 2 (12 punti)

Si consideri il seguente linguaggio:

$$\{a^n\,b\,c^{n+1} \mid n > 0\} \cup \{b^n\,c\,d^k \mid n \geq 0, k > 0\} \cup \{d^n\,a\,c^k\,b^m \mid m = n + k, n > 0, k > 0\}$$

1. Si scriva una grammatica libera dal contesto per il linguaggio
2. Il linguaggio è LL(1)? Se sì, si scriva la tabella per un parser predittivo top-down che analizzi il linguaggio.
3. Utilizzando la tabella si faccia il parsing delle stringhe $aabccc$ e $cddd$

SOLUZIONE

Scrivendo la grammatica in maniera naturale evitando la ricorsione a sinistra e usando la ϵ per i casi base delle ricorsioni semplici a destra si vede che tutti i casi sono mutualmente esclusivi e pertanto la grammatica è LL(1).

$$\begin{array}{lcl} S & \rightarrow & aAcc \mid B \mid dEb \\ A & \rightarrow & aAc \mid b \\ B & \rightarrow & bB \mid cdD \\ D & \rightarrow & dD \mid \epsilon \\ E & \rightarrow & dEb \mid acFb \\ F & \rightarrow & cFb \mid \epsilon \end{array}$$

Si ha FOLLOW(S) = FOLLOW(B) = FOLLOW(D) = $\{\$\}$. FOLLOW$(A)$ = $\{c\}$. FOLLOW(F) = $\{b\}$. La tabella è la seguente (nelle caselle è indicato il numero della produzione da applicare. Le produzioni sono numerate a partire da 1 che è la prima $S \rightarrow aAcc$):

	a	b	c	d	$\$$
S	1	2	2	3	
A	4	5			
B		6	7		
D				8	9
E	11			10	
F		14	13		

Di seguito il parsing di $aabccc$. Quello dell'altra stringa è omesso.

STACK	INPUT	AZIONE
$\$S$	$aabccc\$$	$S \rightarrow aAcc$
$\$ccAa$	$aabcc\$$	match
$\$ccA$	$abcc\$$	$A \rightarrow aAc$
$\$cccAa$	$abccc\$$	match
$\$cccA$	$bccc\$$	$A \rightarrow b$
$\$cccb$	$bccc\$$	con quattro match consecutivi si accetta

ESERCIZIO 3 (12 punti)

Si consideri la seguente grammatica

$$\begin{array}{lcl} S & \rightarrow & Ad \mid Bbc \\ A & \rightarrow & aB \\ B & \rightarrow & aAb \mid c \mid d \end{array}$$

1. Si scriva il linguaggio generato dalla grammatica utilizzando espressioni su insiemi
2. La grammatica è LR(1)? Se sì, si scriva la tabella per un parser bottom-up shift-reduce che analizzi il linguaggio.
3. Si dica quali sono tutti gli item validi per il viable prefix $aaaaa$

SOLUZIONE

$$L(S) = \{a^{2n+1}\,(b|c)\,b^n\,d \mid n \geq 0\} \cup \{a^{2n}\,(c|d)\,b^{n+1}\,c \mid n \geq 0\}$$

Per determinare se la grammatica è LR(1) proviamo prima a verificare se è SLR. Calcoliamo gli insiemi di item LR(0):

$I_0 = \begin{array}{l} S' \to \cdot S \\ S \to \cdot Ad \\ S \to \cdot Bbc \\ A \to \cdot aB \\ B \to \cdot aAb \\ B \to \cdot c \\ B \to \cdot d \end{array}$	$I_1 = \texttt{goto}(I_0, S) = S' \to S\cdot$
$I_2 = \texttt{goto}(I_0, A) = S \to A \cdot d$	$I_3 = \texttt{goto}(I_0, B) = S \to B \cdot bc$
$I_4 = \texttt{goto}(I_0, a) = \begin{array}{l} A \to a \cdot B \\ B \to a \cdot Ab \\ B \to \cdot aAb \\ B \to \cdot c \\ B \to \cdot d \\ A \to \cdot aB \end{array}$	$I_5 = \texttt{goto}(I_0, c) = B \to c\cdot$
$I_6 = \texttt{goto}(I_0, d) = B \to d\cdot$	$I_7 = \texttt{goto}(I_2, d) = S \to Ad\cdot$
$I_8 = \texttt{goto}(I_3, b) = S \to Bb \cdot c$	$I_9 = \texttt{goto}(I_4, B) = A \to aB\cdot$
$I_{10} = \texttt{goto}(I_4, A) = B \to aA \cdot b$	$\texttt{goto}(I_4, a) = I_4$
$\texttt{goto}(I_4, c) = I_5$	$\texttt{goto}(I_4, d) = I_6$
$I_{11} = \texttt{goto}(I_8, c) = S \to Bbc\cdot$	$I_{12} = \texttt{goto}(I_{10}, b) = B \to aAb\cdot$

Si ha che FOLLOW$(S) = \{\$\}$ e FOLLOW$(A) =$ FOLLOW$(B) = \{b, d\}$. Negli stati non ci sono conflitti e quindi la grammatica è SLR, da cui segue che è anche LR(1) per le inclusioni che conosciamo. La tabella di parsing è la seguente (le produzioni sono numerate come al solito a partire dalla 0 che è quella aggiunta $S' \to \cdot S$):

	a	b	c	d	\$	S	A	B
0	$s4$		$s5$	$s6$		1	2	3
1					acc			
2				$s7$				
3		$s8$						
4	$s4$		$s5$	$s6$			10	9
5		$r5$		$r5$				
6		$r6$		$r6$				
7					$r1$			
8			$s11$					
9		$r3$		$r3$				
10		$s12$						
11					$r2$			
12		$r4$		$r4$				

Sappiamo dal teorema centrale del parsing LR che tutti gli item validi di un certo viable prefix sono quelli dello stato della collezione di item LR(0) a cui si giunge partendo da

I_0 ed etichettando con il viable prefix un cammino sull'automa deterministico definito dalla funzione goto. Nel nostro caso il cammino parte da I_0 e va in I_4 con la prima a di $aaaaa$. In seguito, visto che $\texttt{goto}(I_4, a) = I_4$, si eseguono sempre transizioni verso lo stesso stato I_4. Pertanto tutti gli item validi del viable prefix $aaaaa$ sono gli item dell'insieme I_4.