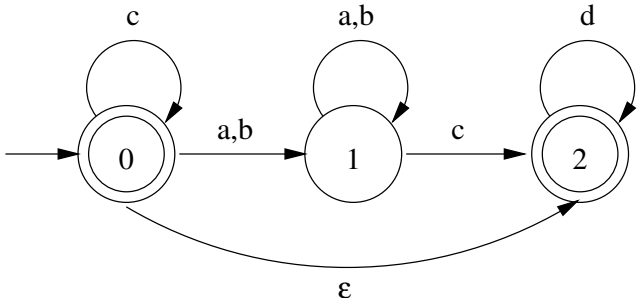
Linguaggi di Programmazione e Compilatori

IV° Appello del 23/9/2004

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 consideri il seguente automa.



1. Si esprima il linguaggio accettato dall'automa mediante una espressione regolare.
2. L'automa è deterministico? Se no, si dia un automa deterministico equivalente.
3. L'automa deterministico è minimo? Si giustifichi la risposta.

SOLUZIONE

Analizzando i cammini vediamo che ci sono tre possibilità: c^*, $c^*(a|b)^+cd^*$ e c^*d^*. Mettendo tutto insieme in una sola espressione regolare otteniamo:

$$c^*(\epsilon \mid (a|b)^+c)d^*$$

L'automa non è deterministico perché contiene una ϵ-transizione. La seguente è la tabella dell'automa deterministico che si ottiene con la costruzione dei sottoinsiemi (A è lo stato iniziale, B e A sono gli stati finali):

	a	b	c	d
$A = \{0, 2\}$	B	B	A	C
$B = \{1\}$	B	B	C	
$C = \{2\}$				C

L'automa risultante ha tre stati. Si può vedere, aggiungendo un *dead state* e procedendo con l'algoritmo di minimizzazione, che il numero di stati non si può ulteriormente diminuire. Quindi l'automa ottenuto con la costruzione dei sottoinsiemi è minimo.

ESERCIZIO 2 (12 punti)

Si consideri il seguente linguaggio:

$$L = \{a^n\, b^m\, c^n \mid n \geq 0, m > 0\} \cup \{c^{2n}\, ba\, c^n \mid n \geq 0\}$$

1. Scrivere una grammatica che generi il linguaggio
2. Il linguaggio è LL(1)? Se sì, si dia la tabella per un parser predittivo che analizzi il linguaggio e si mostrino i passi di costruzione dell'albero di derivazione per la stringa $ccbac$ durante il parsing.

SOLUZIONE

Proviamo a scrivere direttamente una grammatica che sia analizzabile LL(1). Notiamo come prima cosa che i due casi iniziano con simboli terminali diversi per quasi tutte le stringhe. Ci sono delle intersezioni solo per le stringhe $\{b^m \mid m > 0\}$ appartenenti al primo caso e la stringa ba appartenente al secondo caso. Questa situazione può essere gestita facilmente osservando che, dopo aver letto una b, se si legge un'altra b o la fine dell'input allora la stringa appartiene al primo caso. Se invece si legge una a allora la stringa appartiene al secondo caso. Esprimiamo queste considerazioni formalmente scrivendo la grammatica:

$$\begin{array}{lcl} S & \rightarrow & aAc \mid bB \mid ccCc \\ A & \rightarrow & aAc \mid bD \\ D & \rightarrow & bD \mid \epsilon \\ B & \rightarrow & bD \mid a \\ C & \rightarrow & ccCc \mid ba \end{array}$$

Si ha FOLLOW(S) = FOLLOW$(B) = \{\$\}$. FOLLOW$(A)$ = FOLLOW$(C) = \{c\}$. FOLLOW$(D) = \{c, \$\}$.

La tabella di parsing è la seguente:

	a	b	c	$
S	$S \rightarrow aAc$	$S \rightarrow bB$	$S \rightarrow ccCc$	
A	$A \rightarrow aAc$	$A \rightarrow bD$		
D		$D \rightarrow bD$	$D \rightarrow \epsilon$	$D \rightarrow \epsilon$
B	$B \rightarrow a$	$B \rightarrow bD$		
C		$C \rightarrow ba$	$C \rightarrow ccCc$	

Per la costruzione dell'albero sappiamo che all'inizio del parsing esso è costituito solo dalla radice etichettata con S. All'input $ccbac$ la prima azione è quella di espandere S con la produzione $S \rightarrow ccCc$ come indicato nella tabella di parsing. A questo punto l'albero è costituito dalla radice etichettata con S e da quattro figli etichettati, in ordine da sinistra a destra, con i simboli $c\, c\, C\, c$.

I due passi successivi del parsing sono due match con i simboli $c\, c$, seguiti dall'espansione di C con $C \rightarrow ba$ (come indicato dalla tabella). Nell'albero, quindi, vengono aggiunti due figli $b\, a$ al nodo etichettato con C. A questo punto tre match consecutivi portano all'accettazione della stringa da parte del parser e l'albero che abbiamo ottenuto è l'albero di derivazione della stringa.

ESERCIZIO 3 (12 punti)

Si consideri la seguente grammatica:

$$\begin{array}{lcl} S & \rightarrow & bSb \mid aAbB \\ A & \rightarrow & cA \mid cb \\ B & \rightarrow & aBc \mid ca \end{array}$$

1. Si specifichi il linguaggio generato dalla grammatica mediante espressioni su insiemi.

2. La grammatica è LR(1)? Se sì, si dia la tabella per un analizzatore shift-reduce.

SOLUZIONE

$$L = \{b^n\, a\, c^m\, c\, b\, b\, a^k\, c\, a\, c^k\, b^n \mid n, m, k \geq 0\}$$

Come al solito cerchiamo di provare che la grammatica è SLR(1). Se questo è vero sappiamo che è anche LR(1). Calcoliamo la collezione canonica degli item LR(0).

$I_0 = \begin{array}{lcl} S' & \rightarrow & \cdot S \\ S & \rightarrow & \cdot bSb \\ S & \rightarrow & \cdot aAbB \end{array}$	$I_1 = \texttt{goto}(I_0, S) = S' \rightarrow S\cdot$
$I_2 = \texttt{goto}(I_0, b) = \begin{array}{lcl} S & \rightarrow & b \cdot Sb \\ S & \rightarrow & \cdot bSb \\ S & \rightarrow & \cdot aAbB \end{array}$	$I_3 = \texttt{goto}(I_0, a) = \begin{array}{lcl} S & \rightarrow & a \cdot AbB \\ A & \rightarrow & \cdot cA \\ A & \rightarrow & \cdot cb \end{array}$
$I_4 = \texttt{goto}(I_2, S) = S \rightarrow bS \cdot b$ $\texttt{goto}(I_2, b) = I_2$	$\texttt{goto}(I_2, a) = I_3$ $I_5 = \texttt{goto}(I_3, A) = S \rightarrow aA \cdot bB$
$I_6 = \texttt{goto}(I_3, c) = \begin{array}{lcl} A & \rightarrow & c \cdot A \\ A & \rightarrow & c \cdot b \\ A & \rightarrow & cA\cdot \\ A & \rightarrow & \cdot cb \end{array}$	$I_7 = \texttt{goto}(I_4, b) = S \rightarrow bSb\cdot$
$I_8 = \texttt{goto}(I_5, b) = \begin{array}{lcl} S & \rightarrow & aAb \cdot B \\ B & \rightarrow & \cdot aBc \\ B & \rightarrow & \cdot ca \end{array}$	$I_9 = \texttt{goto}(I_6, A) = A \rightarrow cA\cdot$ $I_{10} = \texttt{goto}(I_6, b) = A \rightarrow cb\cdot$ $\texttt{goto}(I_6, c) = I_6$ $I_{11} = \texttt{goto}(I_8, B) = S \rightarrow aAbB\cdot$
$I_{12} = \texttt{goto}(I_8, a) = \begin{array}{lcl} B & \rightarrow & a \cdot Bc \\ B & \rightarrow & \cdot aBc \\ B & \rightarrow & \cdot ca \end{array}$	$I_{13} = \texttt{goto}(I_8, c) = B \rightarrow c \cdot a$ $I_{14} = \texttt{goto}(I_{12}, b) = B \rightarrow aB \cdot c$
$\texttt{goto}(I_{12}, a) = I_{12}$ $\texttt{goto}(I_{12}, c) = I_{13}$	$I_{15} = \texttt{goto}(I_{13}, a) = B \rightarrow ca\cdot$ $I_{16} = \texttt{goto}(I_{14}, c) = B \rightarrow aBc\cdot$

Non ci sono conflitti negli stati e quindi la grammatica è SLR(1). Abbiamo FOLLOW$(S) = \{\$, b\}$, FOLLOW$(A) = \{b\}$ e FOLLOW$(B) = \{c, b, \$\}$. La tabella del parser è, quindi, la seguente:

	a	b	c	\$	S	A	B
0	s3	s2			1		
1				acc			
2	s3	s2			4		
3			s6			5	
4		s7					
5		s8					
6		s10	s6			9	
7		r1		r1			
8	s12		s13				11
9		r3					
10		r4					
11		r2		r2			
12	s12		s13				14
13	s15						
14			s16				
15		r6	r6	r6			
16		r5	r5	r5			