# Linguaggi di Programmazione e Compilatori

II° Appello del 22/7/2004

Scrivere **in stampatello** COGNOME e NOME su ogni foglio. La brutta copia va consegnata (indicare che si tratta della brutta), il testo no.
**NOTA:** Nel testo le espressioni regolari vengono scritte con la usuale convenzione di precedenza: l'operatore $^*$ lega più della concatenazione che, a sua volta, lega più dell'operatore $|$. Inoltre verrà usata l'abbreviazione $+$ con il solito significato.

## ESERCIZIO 1 (5 punti)

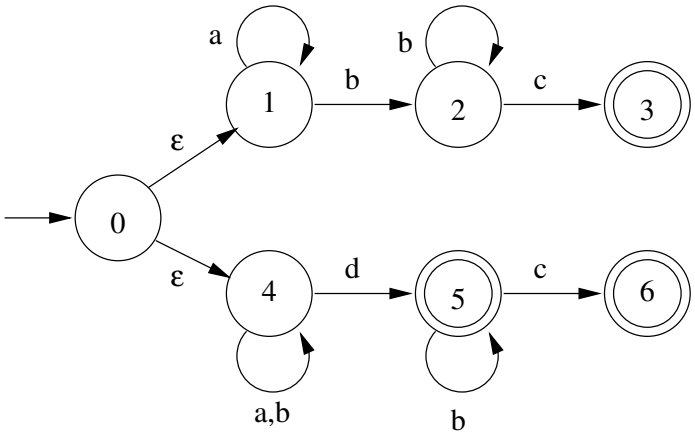
Si scriva un automa **deterministico** che accetti il linguaggio denotato dall'espressione

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

Illustrare il procedimento seguito.

## SOLUZIONE

Possiamo partire da un automa non deterministico ricavato dall'espressione regolare e semplificando un po' quello che verrebbe fuori dalla costruzione di Thompson:




A questo punto applichiamo la costruzione dei sottoinsiemi per ricavare un automa deterministico equivalente. La tabella $move$ risultante è la seguente:

| Stato | $a$ | $b$ | $c$ | $d$ |
|---|---|---|---|---|
| $A = \{0, 1, 4\}$ | $B$ | $C$ | | $D$ |
| $B = \{1, 4\}$ | $B$ | $C$ | | $D$ |
| $C = \{2, 4\}$ | $E$ | $C$ | $F$ | $D$ |
| $D = \{5\}$ | | $D$ | $G$ | |
| $E = \{4\}$ | $E$ | $E$ | | $D$ |
| $F = \{3\}$ | | | | |
| $G = \{6\}$ | | | | |

Gli stati finali sono $F$, $D$ e $G$. Possiamo notare che gli stati $A$ e $B$ sono equivalenti in quanto sono entrambi non finali e si comportano nello stesso identico modo. Volendo, nel disegnare l'automa finale, potremmo accorparli.

## ESERCIZIO 2 (6 punti)

Si consideri un linguaggio di espressioni formate da identificatori (token), parentesi tonde e due operatori **binari** $\oplus$ e $\otimes$. Si scriva, illustrando il procedimento seguito, una grammatica che generi le espressioni di questo linguaggio e che esprima le seguenti regole strutturali tra gli operatori:

- L'operatore $\oplus$ ha precedenza maggiore di $\otimes$
- L'operatore $\oplus$ associa a destra
- L'operatore $\otimes$ associa a sinistra

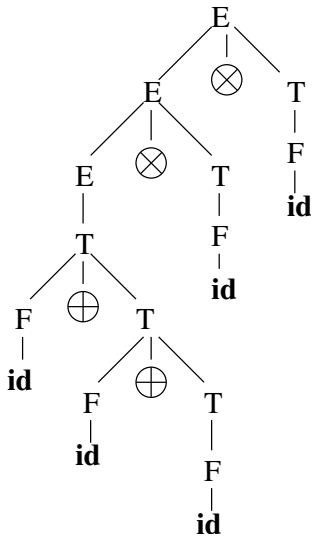
Si costruisca poi un albero di derivazione per la stringa $\mathbf{id} \oplus \mathbf{id} \oplus \mathbf{id} \otimes \mathbf{id} \otimes \mathbf{id}$ secondo la grammatica che si è scritta e si parentesizzi questa espressione secondo la struttura indicata dalle regole di precedenza e associatività.

## SOLUZIONE

Assegnamo ad ogni livello di precedenza un simbolo di categoria sintattica: $F$ per il livello più alto che comprende gli operandi e le espressioni tra parentesi tonde, $T$ per il livello del $\oplus$ ed $E$, il simbolo iniziale, per il livello a precedenza più bassa, cioè quello dell'operatore $\otimes$. Per l'associatività a destra basta scrivere le produzioni con ricorsione a destra e per quella sinistra, analogamente, basta usare la ricorsione a sinistra. La grammatica è la seguente:

$$\begin{aligned} E &\rightarrow E \otimes T \mid T \\ T &\rightarrow F \oplus T \mid F \\ F &\rightarrow \mathbf{id} \mid (E) \end{aligned}$$

L'albero per la stringa è il seguente:




La stringa parentesizzata è la seguente (usiamo le parentesi quadre per enfatizzare che si tratta di una parentesizzazione sottointesa, in quanto la struttura è già individuata dalle regole date):

$$[[\mathbf{id} \oplus [\mathbf{id} \oplus \mathbf{id}]] \otimes \mathbf{id}] \otimes \mathbf{id}$$

## ESERCIZIO 3 (11 punti)

Si consideri il seguente linguaggio:

$$\{a^n\, u\, b^{k-1}\, v\, c^m \mid n, k, m > 0 \text{ e } m = n + k\}$$

1. Scrivere una grammatica che generi il linguaggio.
2. Il linguaggio è LR? Illustrare il procedimento seguito per rispondere alla domanda.
3. Se il linguaggio è LR allora si dia la relativa tabella per il parser bottom-up shift-reduce.

## SOLUZIONE

Il linguaggio può essere equivalentemente espresso come $\{a^n\, u\, b^k\, v\, c\, c^k\, c^n \mid n > 0, k \geq 0\}$. Da questo modo di vedere il linguaggio si ottiene facilmente una grammatica utilizzando due diverse ricorsioni a destra e a sinistra:

$$\begin{aligned} S &\rightarrow aSc \mid auBc \\ B &\rightarrow bBc \mid vc \end{aligned}$$

Proviamo a vedere se questa grammatica è SLR. In questo caso, infatti, sapremmo anche che è LR(1). Se la grammatica è LR(1) allora possiamo affermare che anche il linguaggio lo è, poiché esiste una grammatica LR(1) che lo genera.
Calcoliamo gli insiemi di item LR(0):

| | | | |
|---|---|---|---|
| $I_0 =$ | $S' \to \cdot S$<br>$S \to \cdot aSc$<br>$S \to \cdot auBc$ | $I_1 = goto(I_0, S) =$ | $S' \to S\cdot$ |
| $I_2 = goto(I_0, a) =$ | $S \to a \cdot Sc$<br>$S \to a \cdot uBc$<br>$S \to \cdot aSc$<br>$S \to \cdot auBc$ | $I_3 = goto(I_2, S) =$ | $S \to aS \cdot c$ |
| $I_4 = goto(I_2, u) =$ | $S \to au \cdot Bc$<br>$B \to \cdot bBc$<br>$B \to \cdot vc$ | $goto(I_2, S) = I_2$ | |
| $I_5 = goto(I_3, a) =$ | $S \to aSc\cdot$ | $I_6 = goto(I_4, B) =$ | $S \to auB \cdot c$ |
| $I_7 = goto(I_4, b) =$ | $B \to b \cdot Bc$<br>$B \to \cdot bBc$<br>$B \to \cdot vc$ | $I_8 = goto(I_4, v) =$ | $B \to v \cdot c$ |
| $I_9 = goto(I_6, c) =$ | $S \to auBc\cdot$ | $I_{10} = goto(I_7, B) =$ | $B \to bB \cdot c$ |
| $goto(I_7, b) = I_7$ | | $goto(I_7, v) = I_8$ | |
| $I_{11} = goto(I_8, c) =$ | $B \to vc\cdot$ | $I_{12} = goto(I_{10}, c) =$ | $B \to bBc\cdot$ |

Negli stati non ci sono conflitti e quindi la grammatica è SLR. Per quello che abbiamo detto sopra possiamo concludere che il linguaggio è LR.
Si ha FOLLOW$(S') = \{\$\}$, FOLLOW$(S) = \{c, \$\}$ e FOLLOW$(B) = \{c\}$. La tabella è la seguente (le produzioni sono state numerate nella maniera usuale partendo da zero):

| | $c$ | $a$ | $b$ | $u$ | $v$ | \$ | $S$ | $B$ |
|---|---|---|---|---|---|---|---|---|
| 0 | | s2 | | | | | 1 | |
| 1 | | | | | | acc | | |
| 2 | | s2 | | s4 | | | 3 | |
| 3 | s5 | | | | | | | |
| 4 | | | s7 | | s8 | | | 6 |
| 5 | r1 | | | | | r1 | | |
| 6 | s9 | | | | | | | |
| 7 | | | s7 | | s8 | | | 10 |
| 8 | s11 | | | | | | | |
| 9 | r2 | | | | | r2 | | |
| 10 | s12 | | | | | | | |
| 11 | r4 | | | | | | | |
| 12 | r3 | | | | | | | |

## ESERCIZIO 4 (10 punti)

Si consideri il seguente linguaggio:

$$\{a^n\, b\, c \mid n > 0\} \cup \{b^n\, c\, b \mid n \geq 0\} \cup \{c\, a^n \mid n > 0\}$$

1. Scrivere una grammatica che generi il linguaggio.

2. Il linguaggio è LL(1)? Illustrare il procedimento seguito per rispondere alla domanda.

3. Se il linguaggio è LL(1) si dia una tabella per un parser predittivo top-down che lo analizzi e si esegua il parsing delle stringhe $cb$ e $caa$.

## SOLUZIONE

Cerchiamo subito di scrivere una grammatica che sia analizzabile LL(1). Innanzitutto seguiamo la struttura del linguaggio che si presenta diviso in tre casi differenti (i tre insiemi in unione). Un primo approccio potrebbe essere quello di utilizzare una categoria sintattica diversa per ogni caso. Notiamo che, seguendo questa strada, l'unico problema da risolvere si presenta per le stringhe che iniziano con $c$, in quanto possono essere sia del secondo caso che del terzo. Si vede subito però che una volta letta la prima $c$ è possibile determinare in quale caso ci troviamo guardando il carattere successivo: se il carattere successivo è una $b$ allora ci troviamo nel caso due, mentre se è una $a$ allora ci troviamo nel caso tre. La grammatica è la seguente:

$$\begin{array}{lcl} S & \rightarrow & aA \mid bB \mid cC \\ A & \rightarrow & aA \mid bc \\ B & \rightarrow & bB \mid cb \\ C & \rightarrow & b \mid aD \\ D & \rightarrow & aD \mid \epsilon \end{array}$$

Guardando semplicemente le produzioni per $S, A, B, C$ si vede subito che le guide sono tutte disgiunte. Calcoliamo FOLLOW$(D) = \{\$\}$ e vediamo che anche per il simbolo terminale $D$ le guide sono disgiunte. Possiamo quindi concludere che questa grammatica è LL(1) e pertanto il linguaggio è LL(1).
La tabella del parser predittivo è la seguente:

| | $a$ | $b$ | $c$ | \$ |
|---|---|---|---|---|
| $S$ | $S \rightarrow aA$ | $S \rightarrow bB$ | $S \rightarrow cC$ | |
| $A$ | $A \rightarrow aA$ | $A \rightarrow bc$ | | |
| $B$ | | $B \rightarrow bB$ | $B \rightarrow cb$ | |
| $C$ | $C \rightarrow aD$ | $C \rightarrow b$ | | |
| $D$ | $D \rightarrow aD$ | | | $D \rightarrow \epsilon$ |

Facciamo il parsing di $cb$ e di $caa$:

| STACK | INPUT | AZIONE |
|---|---|---|
| $\$S$ | $cb\$$ | $S \rightarrow cC$ |
| $\$Cc$ | $cb\$$ | match |
| $\$C$ | $b\$$ | $C \rightarrow b$ |
| $\$b$ | $b\$$ | match |
| $\$$ | $\$$ | accept |

| STACK | INPUT | AZIONE |
|---|---|---|
| $\$S$ | $caa\$$ | $S \rightarrow cC$ |
| $\$Cc$ | $caa\$$ | match |
| $\$C$ | $aa\$$ | $C \rightarrow aD$ |
| $\$Da$ | $aa\$$ | match |
| $\$D$ | $a\$$ | $D \rightarrow aD$ |
| $\$Da$ | $a\$$ | match |
| $\$D$ | $\$$ | $D \rightarrow \epsilon$ |
| $\$$ | $\$$ | accept |