Svolgimento appello di Linguaggi di Programmazione e Compilatori (Ascoli Piceno) Traccia 1

Giovedì, 12 luglio 2007

Esercizio 1 - (9 Punti)

Si consideri il linguaggio $\mathcal{L} = \{a^n b^m c^p d^q | n \geq 1, m \geq 0, p \geq 0, q \geq 0, n = m + p - q\}$ e se ne determini la classe di appartenenza in accordo alla classificazione di Chomsky. Si definisca un'automa capace di accettare il linguaggio fornendo la definizione di tutte le sue componenti, commentando altresì le scelte effettuate.

Svolgimento :
Punto 1: La struttura del linguaggio richiede che il numero di occorrenze dei simboli a e d sia pari a quella dei simboli b e c. Vincolo ulteriore che viene posto è che le occorrenze dei simboli non possano essere mischiati, vale a dire prima si dovrà presentare un blocco di simboli a poi di b, successivamente di c ed infine di d. Dalle equazioni risulta che il blocco di simboli a deve essere sempre presente, dunque se la stringa non iniziasse per a l'automa la dovrà immediatamente rifiutare.

Nella derivazione dell'automa si potrebbe pensare di cercare di applicare il seguente algoritmo: si cominci con il contare il numero di occorrenze del simbolo a. Tale valore viene decrementato fino ad essere azzerato quando si incontrano simboli b e successivamente c. Arrivati a zero si riparte con l'incremento del contatore finchè vengono trovati simboli b o c. Chiaramente nel decrementare ed incrementare si dovrà tener conto che comunque le occorrenze di simboli non possono essere mischiati, ovvero dopo una c non può apparire una c. Infine quando il blocco di d viene incontrato si procede descrementando il contatore. Se alla fine della lettura della stringa il contatore contiene il valore zero (la pila è vuota) la stringa viene accettata. Dalla descrizione si possono identificare alcuni stati che possono essere utilizzati nelle fasi di "conteggio" ed allo stesso tempo la necessità di utilizzare un singolo simbolo di pila per implementare il conteggio.

La generazione dell'automa può essere derivata a partire dalle precedenti considerazioni prevedendo i seguenti stati:

- Un primo stato conta le occorrenze di simboli a finché un'occorrenza dei simboli b, c o d non venga individuata.
- Un secondo stato può essere utilizzato per gestire il blocco di simboli b in particolare questo stato può decrementare il "contatore" fino ad azzerarlo. A quel punto se lo stream di input non è ancora vuoto si può transire in uno stato che incrementi il contatore.
- Il terzo stato continua ad incrementare finchè il blocco di b non sia terminato. Se viene incontrato un simbolo c si continua ad incrementare e si passa ad uno stato equivalente a questo ma che considera il simbolo c. Se invece il simbolo incontrato è d si effettua un decremento e si passa in uno stato che gestisca le occorrenze del simbolo d
- Il quarto stato può essere considerato equivalente al secondo dove però si considera il simbolo c.
- Per il quinto stato è equivalente al terzo per il simbolo c
- Infine un'ultimo stato può essere utilizzato per la lettura dell'eventuale blocco di d

Attenzione si rende necessario prevedere stati per ogni simbolo onde evitare che lo stesso stato proceda nel conteggio anche quando i simboli sono mischiati. Dalla descrizione si deriva il seguente automa a pila $A = \langle \Sigma, \Gamma, Z_0, Q, q_0, F, \delta \rangle$ che accetta per pila vuota:

$\Sigma = \{a, b, c, d\}$, $\Gamma = \{C, Z_0\}$, $Q = \{q_1, q_2, q_3, q_4, q_5, q_6\}$, $F = \{\}$ e la tabella 1 per la funzione di transizione δ.

	Z_0					C			
	a	b	c	d	ε	a	b	c	d
q_1	(q_1, C)					(q_1, CC)	(q_2, ε)	(q_4, ε)	(q_6, ε)
q_2		(q_3, CC)	(q_5, CC)				(q_2, ε)	(q_4, ε)	
q_3							(q_3, CC)	(q_5, CC)	(q_6, ε)
q_4			(q_5, CC)					(q_4, ε)	
q_5								(q_5, CC)	(q_6, ε)
q_6									(q_6, ε)

Tabella 1: Automa a pila per il riconoscimento del linguaggio $\mathcal{L}$

Ovviamente altri automi a pila possono essere derivati per accettare lo stesso linguaggio.

Esercizio 2 - (15 Punti)

Si consideri la seguente grammatica G:

$$S \longrightarrow ABC \quad A \longrightarrow Bb \mid a \quad B \longrightarrow Aa \mid b \quad C \longrightarrow abCc \mid abc \tag{1}$$

e si risolvano i seguenti punti, commentando adeguatamente i vari passi attuati:

1. si derivino gli insiemi FIRST, FOLLOW e *nullable* per G. Nella derivazione degli insiemi si annotino i vari simboli con l'indice dell'iterazione e il riferimento alla produzione che hanno richiesto l'aggiunta del simbolo all'insieme;
2. si discuta l'applicabilità del parsing LL(1);
3. si derivi l'automa LR(1) e la corrispondente tabella di parsing discutendo altresì la sua applicabilità.

Svolgimento :
Punto 1: La tabella seguente mostra la derivazione dei differenti insiemi a partire dal seguente ordinamento delle regole:

$$1.A \longrightarrow a \;\; 2.B \longrightarrow b \;\; 3.C \longrightarrow abc \;\; 4.A \longrightarrow Bb \tag{2}$$

$$5.B \longrightarrow Aa; \;\; 6.C \longrightarrow abCc \;\; 7.S \longrightarrow ABC \tag{3}$$

	nullable	FIRST	FOLLOW
S		$a_{1,7}$ $b_{2,7}$	
A		$a_{1,1}$ $b_{2,4}$	$a_{1,5}$ $b_{1,7}$
B		$b_{1,2}$ $a_{1,5}$	$b_{1,4}$ $a_{1,7}$
C		$a_{1,3}$	$c_{1,6}$

Punto 2: La grammatica presenta delle caratteristiche che non permettono di applicare correttamente un'analisi sintattica attraverso un parser LL(1). Si noti a tal fine che dalle produzioni 4 e 5 si derivano delle condizioni di ricorsione sinistra che come noto non permette l'applicazione di tecniche di parsing predittive. In particolare le seguenti due sequenze di produzioni, risultanti in ricorsione sinistra, sono possibili:

$$A \longrightarrow Bb \longrightarrow Aab \quad B \longrightarrow Aa \longrightarrow Bba \tag{4}$$

Si può inoltre notare che le produzioni per il non terminale C iniziano con la stessa sottostringa "ab". Ciò come noto richiederebbe di applicare la tecnica di fattorizzazione sinistra.

In maniera più dispendiosa è possibile arrivare alla stessa conclusione derivando la tabella di parsing LL(1) ed osservando che questa contiene diversi conflitti.

Punto 3: Per derivare l'automa si procede dapprima ad aumentare la grammatica con la produzione $S' \longrightarrow S\$$. Successivamente partendo dall'item $S' \longrightarrow .S, \$$ si procede iterativamente fino a giungere all'automa LR(1) di Figura 1. A partire da questo automa è possibile derivare la tabella di parsing LR(1) mostrata in Tabella 2.

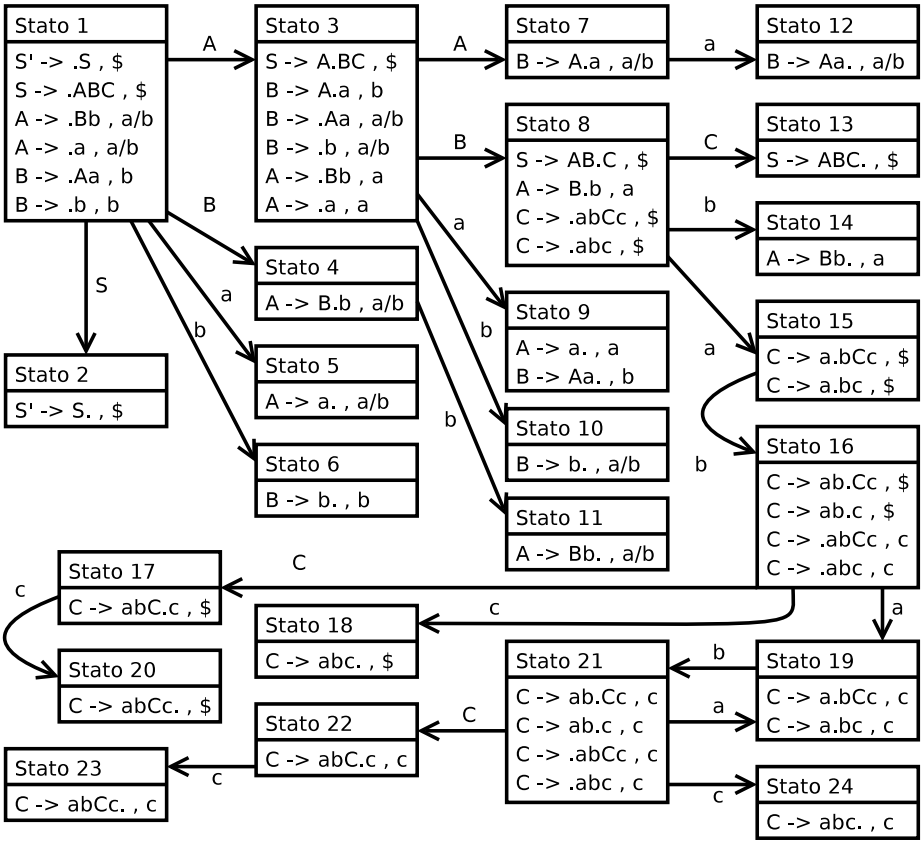

Figura 1: Automa LR(1) per la grammatica G

Osservando la tabella di parsing si può osservare che nessun conflitto viene generato. Si può dunque concludere che la grammatica può essere analizzata utilizzando un parser LR(1).

□

	a	b	c	\$	S	A	B	C
1	S5	S6			G2	G3	G4	
2				acc				
3	S9	S10				G7	G8	
4		S11						
5	$R(A \longrightarrow a)$	$R(A \longrightarrow a)$						
6		$R(B \longrightarrow b)$						
7	S12							
8	S15	S14						G13
9	$R(A \longrightarrow a)$	$R(B \longrightarrow Aa)$						
10	$R(B \longrightarrow b)$	$R(B \longrightarrow b)$						
11	$R(A \longrightarrow Bb)$	$R(A \longrightarrow Bb)$						
12	$R(B \longrightarrow Aa)$	$R(B \longrightarrow Aa)$						
13				$R(S \longrightarrow ABC)$				
14	$R(A \longrightarrow Bb)$							
15		S16						
16	S19		S18					G17
17			S20					
18				$R(C \longrightarrow abc)$				
19		S21						
20				$R(C \longrightarrow abCc)$				
21	S19		S24					G22
22			S23					
23			$R(C \longrightarrow abCc)$					
24			$R(C \longrightarrow abc)$					

Tabella 2: Tabella di parsing LR(1)

Esercizio 3 - (5 Punti)

Si fornisca una formalizzazione per la forma di normale di Chomsky (CNF) e si dimostri che ogni grammatica libera dal contesto in forma ridotta ammette una rappresentazione in CNF.

Svolgimento :
Si consulti il testo Ausiello, D'Amore, Gambosi – Linguaggi, Modelli, Complessità. □

Esercizio 4 - ($\star$ punti)

Si consideri la generazione di una tabella di parsing LR(0). Data una grammatica libera da contesto G in forma ridotta, si dimostri che non è possibile derivare una tabella di parsing in cui per un dato stato esistano conflitti di tipo reduce/goto non associati a conflitti di tipo shift/reduce (vale a dire non è possibile generare un tabella di parsing LR(0) per cui ad uno stato associato ad un'azione di tipo reduce che non presenti conflitti per i simboli terminali, presenti invece azioni di goto per uno dei simboli non terminali).

Svolgimento :
Un possibile modo per dimostrare la preposizione è procedere per assurdo. Assumiamo dunque che sia possibile avere un grammatica di tipo 2 in forma ridotta e corrispondentemente di aver generato una tabella di parsing LR(0) per cui almeno uno stato associato ad un'azione di tipo reduce (e nessun conflitto con azioni di tipo shift) sia associata un'azione goto per almeno uno dei simboli non terminali. Sia allora i il generico indice di un tale stato, $\alpha \longrightarrow a$ la corrispondente azione di riduzione da applicare, e sia X un generico non terminale per cui si determina un'azione di goto. Si consideri dunque lo stato i nell'automa LR(0) ed i corrispondenti item contenuti. In particolare la presenza nella tabella dell'azione di riduzione per lo stato i indica l'inclusione dell'item $\alpha \longrightarrow a.$.
Allo stesso tempo la possibilità di un'azione di tipo Goto per il simbolo non terminale X indica la presenza di un item del tipo $Y \longrightarrow \beta.X\delta$ (dove β e δ sono generiche stringhe di simboli terminali e non).
La presenza di un item di questo tipo richiede, visto che gli stati dell'automa di parsing debbono essere chiusi, anche la presenza di tutte le possibili produzioni per cui X é il simbolo non terminale alla sinistra. Per ipotesi però sappiamo che nessuno degli item contenuti in questo stato può contenere un simbolo terminale alla destra del punto (altrimenti avremmo infatti un conflitto di tipo shift/reduce ed abbiamo assunto che questo non 'e vero). Dunque tutti gli item inclusi a partire dal simbolo X contengono come primo elemento un simbolo non terminale. Ma date le proprietà di chiusura il procedimento dovrà essere ripetuto, includendo gli item per i simboli non terminali alla destra del punto identificati ad ogni passo. Non potendo nessuno degli item inseriti nei passi successivi poter contenere nella prima posizione un simbolo terminale (altrimenti come detto avremmo conflitto shift/reduce) l'unica possibilità per terminare il processo è quello che ad un certo punto tutto i simboli non terminali da dover considerare ad un dato passo siano stati già considerati nei passi precedenti. A questo punto però abbiamo identificato una serie di simboli non terminali sterili ed in particolare ciò sarà vero per il simbolo X. Qualiasi sequenza di produzioni che parta da X conterrà sempre un simbolo non terminale. Vale a dire da X non è possibile generare alcuna stringa del linguaggio e questo vuol dire che la grammatica non è in forma ridotta poiché contiene simboli non prolifici. Siamo giunti a questo punto all'assurdo avendo negato la condizione da cui siamo partiti, ovvero che la grammatica fosse in forma ridotta.

□