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

Lunedì, 17 settembre 2007

Esercizio 1 - (9 Punti)

Si consideri il linguaggio $\mathcal{L} = \{(abc)^n d^n | \exists m \in \mathbb{N}^+ . n = 2m\} \cup \{(abc)^n d^{(n-1)} | \exists m \in \mathbb{N} . n = 2m + 1\}$ 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.
- Si fornisca una grammatica capace di generare il linguaggio.

Svolgimento :
Punto 1: Il linguaggio è formato da due differenti tipi di stringhe corrispondenti ai due insiemi definiti. In particolare il primo insieme ci dice che appartengono al linguaggio le stringhe che hanno una prima sottostringa costituita da un numero pari di ripetizioni della sottostringa *abc* seguita da una sottostringa avente un ugual numero di occorrenze del simbolo *d*. Il secondo insieme definisce stringhe simili alle precedenti dove però il numero di ripetizioni della sottostringa *abc* è dispari ed il numero di ripetizione del simbolo *d* è inferiore di uno.

Un possibile automa potrebbe essere costruito contando il numero di occorrenze della sottostringa *abc* facendo attenzione se il numero fin qui osservato è pari o dispari. Alla prima occorrenza del simbolo *d* si avrà dunque conoscenza del numero di ripetizioni corrette di questo simbolo che si dovranno osservare.

Nella definizione dell'automa si possono fare le seguenti considerazioni:

- Si prevede l'uso di due caratteri di pila A e B. Il primo viene utilizzato per memorizzare nella pila il fatto che l'ultima occorrenza dispari mentre il secondo serve a memorizzare quante "coppie" di occorrenze sono state incontrate.

- alla prima occorrenza del carattere d ci si comporter in maniera differente a seconda che il carettere sulla pila sia A o B
- si deve porre particolare attenzione al fatto che la stringa abc fa parte del linguaggio.

Il seguente automa a pila $A = \langle \Sigma, \Gamma, Z_0, Q, q_0, F, \delta \rangle$ accetta per stato finale il linguaggio $\mathcal{L}$:

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

	Z_0				
	a	b	c	d	ε
q_0	(q_1, ε)				
q_1		(q_2, ε)			
q_2			(q_0, A)		
q_3					
q_4					(q_f, ε)
q_5					
q_6					(q_f, ε)

	A				
	a	b	c	d	ε
q_0	(q_1, A)			(q_3, ε)	(q_f, ε)
q_1		(q_2, A)			
q_2			(q_0, B)		
q_3					
q_4					
q_5					
q_6					

	B				
	a	b	c	d	ε
q_0	(q_1, B)			(q_5, B)	
q_1		(q_2, B)			
q_2			(q_0, BA)		
q_3				(q_4, ε)	
q_4				(q_3, B)	
q_5				(q_6, ε)	
q_6				(q_5, B)	

Tabella 1: Automa a pila che riconosce il linguaggio $\mathcal{L}$

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

Punto 2: Nella definizione della grammatica può essere utile distinguere da principio il caso delle occorrenze pari e dispari. Fatta tale distinzione la derivazione della grammatica è piuttosto semplice.

$$S \longrightarrow A \mid B \quad A \longrightarrow abcabcAdd \mid abcabcdd \quad B \longrightarrow abcabcBdd \mid abc \tag{1}$$

Esercizio 2 - (15 Punti)

Si consideri la seguente grammatica G:

$$S \longrightarrow AC \quad A \longrightarrow Bb \mid a \quad B \longrightarrow Aa \mid b \quad C \longrightarrow aC \mid c \tag{2}$$

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

1. si discuta l'applicabilità del parsing LL(1);
2. 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;
3. si derivi l'automa LR(0) e tabelle di parsing LR(0) e SLR(1) discutendo altresì l'applicabilità dei due differenti tipi di parsing.

Svolgimento :
Punto 1: Se si osservano le produzioni si può immediatamente dedurre che la grammatica non può essere correttamente gestita da un parser LL(1) a causa della ricorsione sinistra che si genera a partire dai caratteri non terminali A e B. Infatti si osserva che:

$$A \longrightarrow Bb \longrightarrow Aab \quad B \longrightarrow Aa \longrightarrow Bba \tag{3}$$

Punto 2: La seguente tabella mostra la composizione degli insiemi FIRST, FOLLOW e nullable a partire dal seguente ordine delle produzioni:

$$1.A \longrightarrow a, 2.B \longrightarrow b, 3.C \longrightarrow c, 4.C \longrightarrow aC, 5.A \longrightarrow Bb, 6.B \longrightarrow Aa, 7.S \longrightarrow AC \tag{4}$$

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

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(0) di figura 1. A partire da questo automa è possibile derivare la tabella di parsing LR(0) mostrata in tabella 2 e la tabella per il parsing SLR(1) mostrata in tabella 3.

Osservando le due tabelle di parsing si può concludere che la grammantica non può essere utilizzata con un parser LR(0). La corrispondente tabella contiene infatti due conflitti di tipo shift/reduce. È possibile invece utilizzare un analizzatore sintattico di tipo SLR, la corrispondente tabella infatti non contiene alcun conflitto.

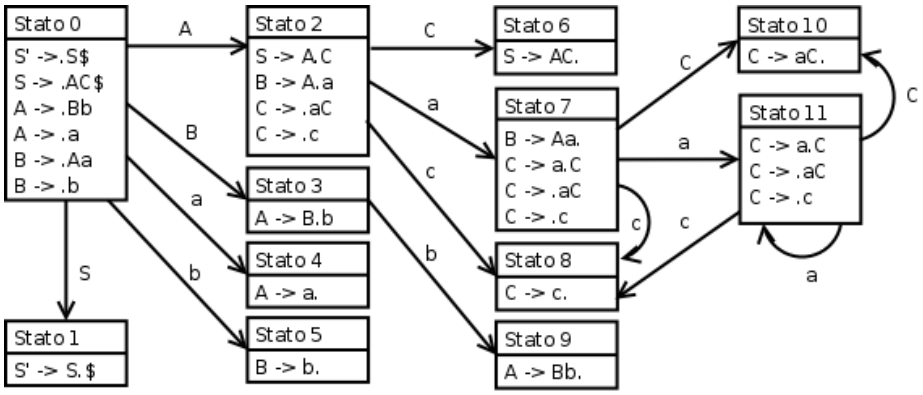

Figura 1: Automa LR(0) per la grammatica G

	a	b	c	\$	S	A	B	C
0	S4	S5			G1	G2	G3	
1	acc	acc	acc	acc				
2	S7		S8				G6	
3		S9						
4	R($A \longrightarrow a$)	R($A \longrightarrow a$)	R($A \longrightarrow a$)	R($A \longrightarrow a$)				
5	R($B \longrightarrow b$)	R($B \longrightarrow b$)	R($B \longrightarrow b$)	R($B \longrightarrow b$)				
6	R($S \longrightarrow AC$)	R($S \longrightarrow AC$)	R($S \longrightarrow AC$)	R($S \longrightarrow AC$)				
7	**R($B \longrightarrow Aa$)/S11**	R($B \longrightarrow Aa$)	**R($B \longrightarrow Aa$)/S8**	R($B \longrightarrow Aa$)				G10
8	R($C \longrightarrow c$)	R($C \longrightarrow c$)	R($C \longrightarrow c$)	R($C \longrightarrow c$)				
9	R($A \longrightarrow Bb$)	R($A \longrightarrow Bb$)	R($A \longrightarrow Bb$)	R($A \longrightarrow Bb$)				
10	R($C \longrightarrow aC$)	R($C \longrightarrow aC$)	R($C \longrightarrow aC$)	R($C \longrightarrow aC$)				
11	S11		S8					G10

Tabella 2: Tabella di parsing LR(0)

	a	b	c	\$	S	A	B	C
0	S4	S5			G1	G2	G3	
1	acc	acc	acc	acc				
2	S7		S8				G6	
3		S9						
4	R($A \longrightarrow a$)		R($A \longrightarrow a$)					
5		R($B \longrightarrow b$)						
6				R($S \longrightarrow AC$)				
7	S11	R($B \longrightarrow Aa$)	S8					G10
8				R($C \longrightarrow c$)				
9	R($A \longrightarrow Bb$)		R($A \longrightarrow Bb$)					
10				R($C \longrightarrow aC$)				
11	S11		S8					G10

Tabella 3: Tabella di parsing SLR(1)

Esercizio 3 - (5 Punti)

Si dimostri che per ogni linguaggio regolare $\mathcal{L}$ esiste una costance n tale che se $z \in \mathcal{L}$ e $|z| \geq n$ allora è possibile scrivere $z = uvw$, con $|uv| \leq n$, e $v \geq 1$ e ottenere che $uv^iw \in \mathcal{L}$, per ogni $i \geq 0$

Svolgimento :
Il teorema che chiede di dimostrare quello del pumping lemma. Per i detagli si veda dimostrazione nel capitolo dei linguaggi regolari sul libro di testo.

Esercizio 4 - ($\star$ punti)

Si discuta perché la seguente grammatica non è di tipo LL(1) e si proponga una tecnica per trasformarla in modo che la grammatica risultante sia di tipo LL(1).

$$S \longrightarrow P \mid Q \quad P \longrightarrow Qq \mid p \quad Q \longrightarrow Pp \mid q \tag{5}$$

Svolgimento :
Come osservato nell'esercizio 2 le produzioni della grammatica in oggetto generano delle ricorsioni sinistre per derivazioni più lunghe di un passo a partire dai simboli non terminali P e Q. In particolare vale che:

$$P \longrightarrow Qq \longrightarrow Ppq \quad Q \longrightarrow Pp \longrightarrow Qqp \tag{6}$$

Al fine di rendere la grammatica LL(1) si possono esplicitare le possibili produzioni di lunghezza 1 che non generano stringhe di soli terminali e le sequenze di produzioni di lunghezza minore o uguale ad 1 che generano solo terminali. In questo modo otteremo che dal simbolo non terminale P si possono generare le seguenti sottostringhe (terminali e non):

$$P \longrightarrow Qq \longrightarrow Ppq \quad P \longrightarrow Qq \longrightarrow qq \quad P \longrightarrow p \tag{7}$$

Similmente possiamo operare per Q ottenendo:

$$Q \longrightarrow Pp \longrightarrow Qqp \quad Q \longrightarrow Pp \longrightarrow pp \quad Q \longrightarrow q \tag{8}$$

A questo punto la seguente grammatica equivalente a quella di partenza:

$$S \longrightarrow P \mid Q \quad P \longrightarrow Ppq \mid qq \mid p \quad Q \longrightarrow Qqp \mid pp \mid q \tag{9}$$

Che presenta una ricorsione sinistra "standard". Applicando le tecniche di rimozione della ricorsione sinistra note si giunge alla soluzione.