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

Giovedì, 21 giugno 2007

Esercizio 1 - (10 Punti)

Si consideri il linguaggio $\mathcal{L} = \{a^m b^{2n} c^{3n} | m \geq 1, n \geq 0\}$ e si risolvano i seguenti punti:

1. Determinare la classe del linguaggio $\mathcal{L}$ in accordo alla classificazione di Chomsky definendo altresì le differenti componenti di un opportuno automa capace di accettare il linguaggio $\mathcal{L}$
2. Derivare una grammatica G, che non contenga ε-produzioni, tale che $L(G) = \mathcal{L}$

Svolgimento :
Punto 1: Il linguaggio presenta una struttura che richiede ad un automa di memorizzare quanti simboli b sono stati incotrati per poter successivamente decidere se il numero di simboli c che seguono è corretto. Come più volte rimarcato tale caratteristica esclude la possibilit di utilizzare Automi a Stati Finiti.

Per decidere se il linguaggio è incluso nella classe dei linguaggi liberi dal contesto si procede nella derivazione di un automa a pila capace di accettare il linguaggio.

Quello che segue è effettivamente un automa a pila, che accetta per stato finale, capace di accettare tutte le stringhe appartenenti ad $\mathcal{L}$. Le idee alla base della costruzione dell'automa sono le seguenti:

- Si procede nella lettura di a senza prendere nessuna particolare attenzione finché non venga identificata una b.
- al fine di verificare che le b incontrate siano in numero pari si prevedono due stati il primo denota il fatto che sono state incontrate un numero

dispari di b il secondo un numero pari. Il secondo stato annotera sulla pila anche il fatto che è stata letta dall'input una coppia di b.

- analogamente a quanto fatto per il simbolo b si prevederanno tre stati capaci di contare il numero di triple di simboli c letti. L'ultimo di questi stati provvede anche a rimuovere dalla pila il simboli introdotto nel conteggio delle coppi di simboli b.

Dunque sia $A = \langle \Sigma, \Gamma, Z_0, Q, q_0, F, \delta \rangle$ l'automa per cui dobbiamo definire le varie componenti. Avremo che un possibile automa per cui risulti $L(A) = \mathcal{L}$ si ottiene definendo le varie componenti nel seguente modo:
$\Sigma = \{a, b, c\}$, $\Gamma = \{C, Z_0\}$, $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$, $F = \{q_6\}$ e la seguente tabella per la funzione di transizione δ:

	Z_0				C		
	a	b	c	ε	a	b	c
q_0	(q_0, ε)	(q_2, ε)		(q_6, ε)			
q_1						(q_2, C)	(q_4, C)
q_2		(q_1, C)				(q_1, CC)	
q_3				(q_6, ε)			(q_4, C)
q_4							(q_5, C)
q_5							(q_3, ε)

Ovviamente questo non è l'unico automa a pila possibile. In particolare sarebbe stato possibile ridurre il numero di stati incrementando il numero di simboli sulla pila. Lo studente pu provare a trovare questa soluzione alternativa.
Punto 2: Per derivare la grammatica si osservi che le stringhe del linguaggio possono essere scomposte in tre blocchi distinti due dei quali però risultano correlati dal fatto che devono contenere un numero di simboli legato allo stesso valore n. Nella generazione della grammatica si prevederanno dunque regole che permetteranno di generare sequenze di a indipendentemente dal resto e regole che invece generano coppie di simboli b seguite da triple di simboli c.

Tenendo conto di questo fatto si aggiunge una prima regola che permetterà di disaccopiare la generazione delle a dal resto della stringa. Dopodiché a partire dai simboli non terminali inseriti nella prima regola si deriveranno le regole che ci permettono di avere il comportamento desiderato. Dunque una possibile grammatica G tale che $L(G) = \mathcal{L}$, tenendo anche conto del fatto che $m \geq 1$ e $n \geq 0$, sarà:

$$S \longrightarrow AB \mid A \quad A \longrightarrow aA \mid a \quad B \longrightarrow bbBccc \mid bbccc \tag{1}$$

□

Esercizio 2 - (14 Punti)

Si consideri la seguente grammatica G:

$$S \longrightarrow B \mid C \quad B \longrightarrow bB \mid b \quad C \longrightarrow bbCa \mid a \tag{2}$$

e si risolvano i seguenti punti:

1. Senza aver derivato gli insiemi FIRST e FOLLOW si decida se la grammatica G è LL(1) e perché?
2. Si derivino gli insiemi FIRST, FOLLOW e *nullable* per G indicando tutte le iterazioni necessarie.
3. Si costruisca l'automa LR(0) e le corrispondenti tabelle di parsing LR(0) ed SLR(1) decidendo per ogni tipo di parsing se è applicabile e perché.
4. Applicando una tra le due tipologie di parsing (a scelta LR(0) o SLR(1) se entrambe possibili) si mostrino le azioni del parser sulla stringa "*bbbbaaa*"

Svolgimento :
Punto 1: La grammatica presenta due produzioni, per lo stesso simbolo non terminale B che iniziano con la stessa sottostringa b nella parte sinistra ($B \longrightarrow bB \mid b$). Un parser LL(1) non potrà dunque essere in grado di decidere quale produzione applicare, dall'osservazione di un singolo simbolo di lookahead, quando, cercando di identificare la produzione da applicare per il simbolo non terminale B si trover ad osservare nello stream di input il simbolo b. In questa situazione si potrà cercare di intervenire applicando una fattorizzazione sinistra che potrebbe in alcuni casi fornire una grammatica per cui il parsing LL(1) diventa possibile. Si veda esercizio 4 per dettagli.
Punto 2: La tabella seguente mostra la derivazione dei differenti insiemi a partire dal seguente ordinamento delle regole:

$$1.B \longrightarrow b \;\; 2.C \longrightarrow a \;\; 3.B \longrightarrow bB \;\; 4.C \longrightarrow bbCa; \;\; 5.S \longrightarrow B \;\; 6.S \longrightarrow C \tag{3}$$

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

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 1 e la tabella per il parsing SLR(1) mostrata in tabella 2.

Osservando le due tabelle di parsing si può concludere che la la grammantica non può essere utilizzata con un parser LR(0). La corrispondente tabella

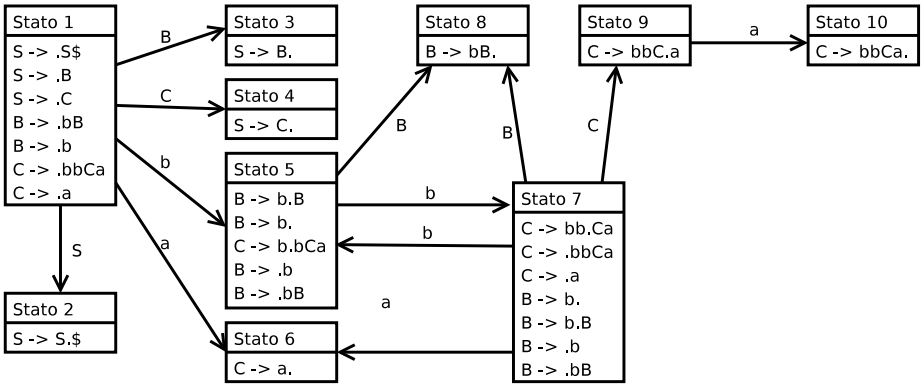

Figura 1: Automa LR(0) per la grammatica G

	a	b	\$	S	B	C
1	S6	S5		G2	G3	G4
2	acc	acc	acc			
3	R($S \longrightarrow B$)	R($S \longrightarrow B$)	R($S \longrightarrow B$)			
4	R($S \longrightarrow C$)	R($S \longrightarrow C$)	R($S \longrightarrow C$)			
5	R($B \longrightarrow b$)	**R($B \longrightarrow b$)/S7**	R($B \longrightarrow b$)		G8	
6	R($C \longrightarrow a$)	R($C \longrightarrow a$)	R($C \longrightarrow a$)			
7	**R($B \longrightarrow b$)/S6**	**R($B \longrightarrow b$)/S5**	R($B \longrightarrow b$)		G8	G9
8	R($B \longrightarrow bB$)	R($B \longrightarrow bB$)	R($B \longrightarrow bB$)			
9	S10					
10	R($B \longrightarrow bbCa$)	R($B \longrightarrow bbCa$)	R($B \longrightarrow bbCa$)			

Tabella 1: Tabella di parsing LR(0)

	a	b	\$	S	B	C
1	S6	S5		G2	G3	G4
2			acc			
3			R($S \longrightarrow B$)			
4			R($S \longrightarrow C$)			
5		S7	R($B \longrightarrow b$)		G8	
6	R($C \longrightarrow a$)		R($C \longrightarrow a$)			
7	S6	S5	R($B \longrightarrow b$)		G8	G9
8			R($B \longrightarrow bB$)			
9	S10					
10	R($B \longrightarrow bbCa$)		R($B \longrightarrow bbCa$)			

Tabella 2: Tabella di parsing SLR(1)

contiene infatti tre conflitti di tipo shift/reduce. È possibile invece utilizzare un analizzatore sintattico di tipo SLR, la corrispondente tabella infatti non

contiene alcun conflitto.
Punto 4: Dallo svolgimento del punto precedente si deduce che l'unico parsing applicabile (tra LR(0) e SLR(1)) è quello di tipo SLR(1). Corrispondentemente i passi dell'analizzatore sintattico SLR saranno quelli mostrati nella tabella 3.

STACK	INPUT	
$\$_1$	$bbbbaaa\$$	shift 5
$\$_1 b_5$	$bbbaaa\$$	shift 7
$\$_1 b_5 b_7$	$bbaaa\$$	shift 5
$\$_1 b_5 b_7 b_8$	$baaa\$$	shift 7
$\$_1 b_5 b_7 b_8 b_7$	$aaa\$$	shift 6
$\$_1 b_5 b_7 b_8 b_7 a_6$	$aa\$$	R($C \longrightarrow a$)
$\$_1 b_5 b_7 b_8 b_7 C_9$	$aa\$$	shift 10
$\$_1 b_5 b_7 b_8 b_7 C_9 a_{10}$	$a\$$	R($C \longrightarrow bbCa$)
$\$_1 b_5 b_7 C_9$	$a\$$	shift 10
$\$_1 b_5 b_7 C_9 a_{10}$	$\$$	R($C \longrightarrow bbCa$)
$\$_1 C_4$	$\$$	R($S \longrightarrow C$)
$\$_1 S_2$	$\$$	accept

Tabella 3: Parsing SLR(1) per la stringa *bbbbaaa*

Dalla tabella 3 si deriva la seguente sequenza di derivazioni che permette di ottenere la stringa *bbbbaaa* dal simbolo iniziale S:
$S \longrightarrow C$ [da cui applicando la produzione $C \longrightarrow bbCa$] $\longrightarrow bbCa$ [da cui applicando la produzione $C \longrightarrow bbCa$] $\longrightarrow bbbbCaa$ [da cui applicando la produzione $C \longrightarrow a$] *bbbbaaa*

□

Esercizio 3 - (5 Punti)

Siano dati i seguenti linguaggi regolari $\mathcal{L}$, $\mathcal{L}_1$ e $\mathcal{L}_2$. Si dimostrino le seguenti proprietà derivando gli opportuni automi:

1. $\mathcal{L}_1 \cup \mathcal{L}_2$ è un linguaggio regolare
2. $\mathcal{L}^*$ è un linguaggio regolare

Svolgimento :
Punto 1: Siano A_1 ed A_2 due automi che accettano rispettivamente i linguaggi $\mathcal{L}_1$ ed $\mathcal{L}_2$, per cui dunque risulta $L(A_1) = \mathcal{L}_1$ e $L(A_2) = \mathcal{L}_2$. Posti dunque: $A_1 = \langle \Sigma_1, Q_1, q_0^1, F_1, \delta_1 \rangle$ e $A_2 = \langle \Sigma_2, Q_2, q_0^2, F_2, \delta_2 \rangle$ l'automa $A_\cup = \langle \Sigma_\cup, Q_\cup, q_0^\cup, F_\cup, \delta_\cup \rangle$ così definito:

- $\Sigma_\cup = \Sigma_1 \cup \Sigma_2$
- $Q_\cup = Q_1 \cup Q_2 \cup \{q_0^\cup\}$
- $F_\cup = F_1 \cup F_2$ oppure $F_\cup = F_1 \cup F_2 \cup \{q_0^\cup\}$ nel caso uno dei due automi A_1 o A_2 accetti la stringa vuota ε.
- e funzione di transizione così definita:

$$\delta_\cup(q, a) = \delta_1(q, a) \text{ se } q \in Q_1 \text{ e } a \in \Sigma_1$$
$$\delta_\cup(q, a) = \delta_2(q, a) \text{ se } q \in Q_2 \text{ e } a \in \Sigma_2$$
$$\delta_\cup(q_0^\cup, a) = \delta_1(q_0^1, a) \cup \delta_1(q_0^2, a),\ \forall a \in \Sigma$$

accetta il linguaggio $\mathcal{L}_1 \cup \mathcal{L}_2$ ovvero $L(A_\cup) = \mathcal{L}_1 \cup \mathcal{L}_2$
Punto 2: Sia A un automa che accetta il linguaggio $\mathcal{L}$, per cui dunque risulta $L(A) = \mathcal{L}$. Definiamo l'automa $A^* = \langle \Sigma, Q \cup \{q_0^*\}, q_0^*, F \cup \{q_0^*\}, \delta^* \rangle$ dove la funzione di transizione è così definita:

$$\delta^*(q, a) = \delta(q, a) \quad \forall q \in Q - F$$

$$\delta^*(q, a) = \delta(q, a) \cup \delta(q_0, a) \quad \forall q \in F$$

$$\delta^*(q_0^*, a) = \delta(q_0, a)$$

L'automa A^* accetta il linguaggio $\mathcal{L}^*$.

□

Esercizio 4 - ($\star$ punti)

Si consideri la grammatica definita nell'esercizio 2. Si derivi una nuova grammatica G' rimuovendo i problemi identificati nel punto 1 dello stesso esercizio. Si discuta se esiste un $k \geq 1$ tale che la grammatica G' risulti LL(k).

Suggerimento: può essere utile osservare che il linguaggio generato dalla grammatica G' è descritto dalla seguente unione d'insiemi:

$$L(G') = \{b^n \mid n \geq 1\} \cup \{b^{2m}a^{m+1} \mid m \geq 0\} \tag{4}$$

Svolgimento :
Come discusso nel punto 1 dell'esercizio 2, due o più produzioni per uno stesso simbolo non teminale che abbiano lo stessa sottostringa come prefisso nel lato sinistro, non permettono di utilizzare un parser LL(1). La tecnica di fattorizzazione sinistra ci permette di risolvere questo problema introducendo un simbolo non terminale aggiuntivo. Applicando la fattorizzazione sinistra possiamo dunque derivare la seguente grammatica G':

$$S \longrightarrow B \mid C \quad B \longrightarrow bD \quad D \longrightarrow B \mid \varepsilon \quad C \longrightarrow bbCa \mid a \tag{5}$$

A questo punto dobbiamo decidere se sia possibile identificare un $k \geq 1$ tale che la grammatica G' sia analizzabile con un parser corrispondente LL(k). A tal fine si può osservare che le stringhe del linguaggio sono generate da due differenti blocchi. Il primo basato su B che genera stringhe del tipo b^n con $n \geq 1$ ed il secondo basato su C che genera stringhe del tipo $b^{2m}a^{m+1}$ con $m \geq 0$. Dunque nei due casi la prima produzione da applicare in una derivazione di una stringa dal simbolo iniziale S sarà rispettivamente $S \longrightarrow B$ o $S \longrightarrow C$. Queste due produzioni dovranno essere le prime identificate da un parser di tipo LL. Nel prendere questa decisione il parser avrà bisogno di osservare se nella stringa appare un simbolo a dopo una sequenza di n simboli b.[1] Ma allora qualsiasi sia il valore di k un parser LL(k) non sarà capace di produrre una derivazione corretta per ogni stringa di input "$b^n\alpha$" per cui $n \geq k$.

□

[1] Il valore n utilizzato qui non si riferisce a quello della definizione dell'insieme