Linguaggi e Traduttori – Tempo: 2 ore

Prof. Marco Gavanelli

17 gennaio 2018

Esercizio 1 (3 punti)

Si consideri il linguaggio $L = \{a^n b^n | n > 0\} \cup \{aab^k | k > 0\}$.

Si classifichi il linguaggio secondo Chomsky.

Si scriva una grammatica non ambigua che genera il linguaggio L; si fornisca la grammatica di livello più basso possibile nella classificazione secondo Chomsky (intendendo il livello 3 come minimo e il livello 0 come massimo).

Se è possibile, si mostri l'albero di derivazione della stringa *aabb*.

Esercizio 2 (6 punti)

Si consideri la grammatica $G = \langle \{a, b, c, d\}, \{S, A, B, C\}, P, S \rangle$, dove:

$$P = \begin{array}{l} S \to ABC \\ A \to aAd \mid \epsilon \\ B \to bBd \mid \epsilon \\ C \to c \mid dA \end{array}$$

1. Si classifichi la grammatica secondo Chomsky.
2. La grammatica è LL(1)? Se sì, si scriva la parsing table del PDA riconoscitore. Se no, si motivi il perché.
3. Se possibile, si scriva una grammatica G' equivalente a G che non abbia produzioni con la stringa vuota. Se non è possibile, si motivi il perché.
4. Qualora al punto 2 si sia ottenuto un riconoscitore, si mostri il riconoscimento delle stringhe *daadd*, *adc*, *adbdd*, mostrando l'evoluzione dello stack.

Esercizio 3 (3 punti)

Sia data la seguente grammatica: $G = \langle \{a, b, c\}, \{S, A, B\}, P, S \rangle$, dove

$$P = \begin{array}{l} S \to aA \mid aB \mid bB \\ A \to b \mid aC \mid bC \\ B \to a \mid bC \\ C \to b \mid aB \end{array}$$

Si disegni il corrispondente automa riconoscitore. L'automa è deterministico? Se non lo è, si produca un automa deterministico equivalente indicando chiaramente gli stati iniziali e finali.

Soluzione 1

Il linguaggio è di tipo 2.

Si noti come la stringa $aabb$ sia in comune ad entrambi gli insiemi con cui il linguaggio è definito; la grammatica più immediata

$G = \langle \{a, b\}, \{S, A, B\}, P, S \rangle$

$$P = \begin{array}{l} S \to A \mid aaB \\ A \to aAb \mid ab \\ B \to bB \mid b \end{array}$$

risulta quindi ambigua, in quanto ci sono due derivazioni per la stringa $aabb$.

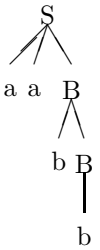
Per avere una grammatica non ambigua bisognerà togliere la stringa $aabb$ dal linguaggio generato da A o da aaB. Scegliamo ad esempio di eliminarla da $L(A)$:

$$P = \begin{array}{l} S \to A \mid aaB \\ A \to aAb \mid aaabbb \\ B \to bB \mid b \end{array}$$

A questo punto, non viene più generata la stringa ab; la aggiungiamo esplicitamente:

$$P = \begin{array}{l} S \to A \mid aaB \mid ab \\ A \to aAb \mid aaabbb \\ B \to bB \mid b \end{array}$$

L'albero di derivazione:



Soluzione 2

1. La grammatica è di tipo 2 (context-free).
2. Per verificare se la nuova grammatica è LL(1) e scrivere la parsing table, scriviamo gli insiemi FIRST e FOLLOW dei simboli non terminali

	$FIRST$	$FOLLOW$
S	$\{a, b, c, d\}$	$\{\$\}$
A	$\{a, \epsilon\}$	$\{b, c, d, \$\}$
B	$\{b, \epsilon\}$	$\{c, d\}$
C	$\{c, d\}$	$\{\$\}$

La parsing table risulta:

	a	b	c	d	\$
S	$S \to ABC$	$S \to ABC$	$S \to ABC$	$S \to ABC$	
A	$A \to aAd$	$A \to \epsilon$	$A \to \epsilon$	$A \to \epsilon$	$A \to \epsilon$
B		$B \to bBd$	$B \to \epsilon$	$B \to \epsilon$	
C			$C \to c$	$C \to dA$	

Non presenta conflitti, quindi G è LL(1).

3. La gramamtica originaria:

$$P = \begin{array}{l} S \to ABC \\ A \to aAd \mid \epsilon \\ B \to bBd \mid \epsilon \\ C \to c \mid dA \end{array}$$

I non terminali che possono riscriversi nella stringa vuota sono A e B. Togliamo le regole $A \to \epsilon$ e $B \to \epsilon$ e sostituiamo A e B rispettivamente con $(A \mid \epsilon)$ e $(B \mid \epsilon)$:

$$P = \begin{array}{l} S \to (A \mid \epsilon)(B \mid \epsilon)C \\ A \to a(A \mid \epsilon)d \\ B \to b(B \mid \epsilon)d \\ C \to c \mid d(A \mid \epsilon) \end{array}$$

distribuendo:

$$P = \begin{array}{l} S \to ABC \mid BC \mid AC \mid C \\ A \to aAd \mid ad \\ B \to bBd \mid bd \\ C \to c \mid dA \mid d \end{array}$$

Riconoscimento:

Input	Stack
daadd\$	S\$
daadd\$	ABC\$
daadd\$	BC\$
daadd\$	C\$
daadd\$	C\$
daadd\$	dA\$
aadd\$	A\$
aadd\$	aAd\$
add\$	Ad\$
add\$	aAdd\$
dd\$	Add\$
dd\$	dd\$
d\$	d\$
\$	\$

Stringa riconosciuta.

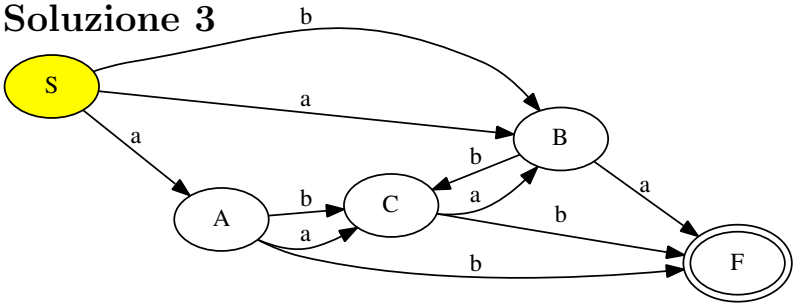
Input	Stack
adc\$	S\$
adc\$	ABC\$
adc\$	aAdBC\$
dc\$	AdBC\$
dc\$	dBC\$
c\$	BC\$
c\$	C\$
c\$	c\$
\$	\$

Stringa riconosciuta.

Input	Stack
adbdd$	S$
adbdd$	ABC$
adbdd$	aAdBC$
dbdd$	AdBC$
dbdd$	dBC$
bdd$	BC$
bdd$	bBdC$
dd$	BdC$
dd$	dC$
d$	C$
d$	dA$
$	A$
$	$

stringa riconosciuta

Soluzione 3



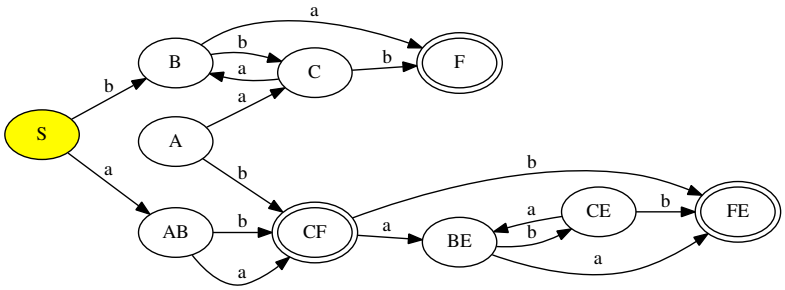
L'automa non è deterministico, come si vede dal fatto che con input a dallo stato S si può passare sia in A sia in B e con input b da A si può andare in C o in F.

Calcoliamo un automa deterministico che riconosce lo stesso linguaggio.

La tabella di transizione, aggiungendo uno stato di errore E e aggiungendo nuovi stati man mano che si presentano:

	a	b
S	AB	B
A	C	CF
B	F	C
C	B	F
F	E	E
AB	CF	CF
CF	BE	FE
BE	FE	CE
CE	BE	FE
FE	E	E

L'automa risultante:



Chiaramente l'automa non è minimo (ad esempio, lo stato A è irraggiungibile), ma la minimizzazione non era richiesta.