Prova scritta di

`Linguaggi` e `Compilatori`

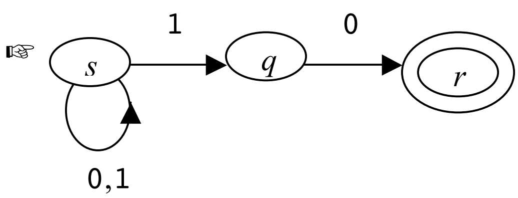
`Compilatori`

del 15 febbraio 2011

ESERCIZIO 1.

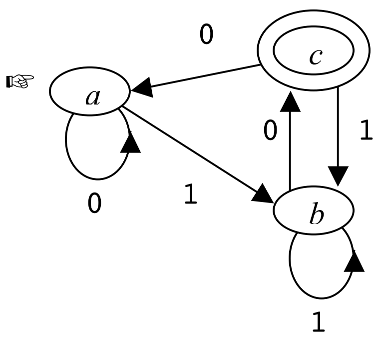
(*a*)	Disegnare un automa finito nondeterministico **A** privo di transizioni spontanee che accetti tutte e solo le stringhe binarie che terminano con ‘`10`’.

Il diagramma delle transizioni di **A** è mostrato in figura.



(*b*)	Costruire (con il metodo dei sottoinsiemi) un automa finito deterministico completo **D** equivalente ad **A**.

Il diagramma delle transizioni di **D** è mostrato in figura.



con $a = \{s\}$, $b = \{s, q\}$ e $c = \{s, r\}$.

(*c*)	Applicando il Lemma di Arden una volta ad **A** ed un’altra a **D**, trovare le espressioni regolari che denotano i linguaggi accettati rispettivamente da **A** e da **D**. (Si osservi che le due espressioni regolari sono equivalenti).

Per **A** abbiamo il sistema di equazioni:

$\lambda_s = \{0, 1\} \cdot \lambda_s \cup \{1\} \cdot \lambda_q$
$\lambda_q = \{0\} \cdot \lambda_r$
$\lambda_r = \{\varepsilon\}$

che ammette la soluzione:

$\lambda_s = \{0, 1\}^* \{10\}$
$\lambda_q = \{0\}$
$\lambda_r = \{\varepsilon\}$

sicché un'espressione regolare che denota il linguaggio accettato da **A** è

$(0|1)^*(10)$.

Per **D** abbiamo il sistema di equazioni:

$\lambda_a = \{0\} \cdot \lambda_a \cup \{1\} \cdot \lambda_b$
$\lambda_b = \{1\} \cdot \lambda_b \cup \{0\} \cdot \lambda_c$
$\lambda_c = \{0\} \cdot \lambda_a \cup \{1\} \cdot \lambda_b \cup \{\varepsilon\}$

che ammette la soluzione:

$\lambda_a = (\{0\} \cup \{1\}\{1, 01\}^*\{00\})^*\{1\}(\{1\} \cup \{01)\}^*\{0\}$
$\lambda_b = \dots$
$\lambda_c = \dots$

sicché un'espressione regolare che denota il linguaggio accettato da **D** è

$(0|1(1|01)^*00)^*1(1|01)^*0$.

(*d*)	Applicare a **D** l'algoritmo di minimizzazione per ottenere un automa finito deterministico completo equivalente a **D** di dimensione minima.

La tabella delle transizioni di **D** è

stato	0	1
a	*a*	*b*
b	*c*	*b*
c	*a*	*b*

La partizione iniziale degli stati è formata dalle due classi $X = \{a, b\}$ ed $R = \{c\}$. Consideriamo la sottotabella della tabella delle transizioni indotta da X e nelle colonne riportiamo, anziché gli stati, le classi a cui appartengono.

stato	0	1
a	X	X
b	R	X

Pertanto, X viene scomposto nelle due classi elementari: $\{a\}$ e $\{b\}$. Avendo così ottenuta la partizione puntuale degli stati di **D**, si conclude che **D** è di dimensione minima.

ESERCIZIO 2.

Si consideri la grammatica acontestuale **G** con produzioni

$$\mathtt{S} \rightarrow \mathtt{0S1S} \mid \mathtt{1S0S} \mid \varepsilon$$

(*a*) Costruire una grammatica equivalente a **G** in Forma Normale di Chomsky.

Una grammatica $\mathbf{G}^{\#}$ equivalente a **G** in Forma Normale di Chomsky ha simbolo speciale $\mathtt{S}^{\#}$ e le seguenti produzioni

$$\mathtt{S}^{\#} \rightarrow \mathtt{AC} \mid \mathtt{AD} \mid \mathtt{AE} \mid \mathtt{AB} \mid \mathtt{BF} \mid \mathtt{BG} \mid \mathtt{BH} \mid \mathtt{BA} \mid \varepsilon$$
$$\mathtt{S} \rightarrow \mathtt{AC} \mid \mathtt{AD} \mid \mathtt{AE} \mid \mathtt{AB} \mid \mathtt{BF} \mid \mathtt{BG} \mid \mathtt{BH} \mid \mathtt{BA}$$
$$\mathtt{A} \rightarrow \mathtt{0}$$
$$\mathtt{B} \rightarrow \mathtt{1}$$
$$\mathtt{C} \rightarrow \mathtt{SE}$$
$$\mathtt{D} \rightarrow \mathtt{SB}$$
$$\mathtt{E} \rightarrow \mathtt{BS}$$
$$\mathtt{F} \rightarrow \mathtt{SH}$$
$$\mathtt{G} \rightarrow \mathtt{SA}$$
$$\mathtt{H} \rightarrow \mathtt{AS}$$

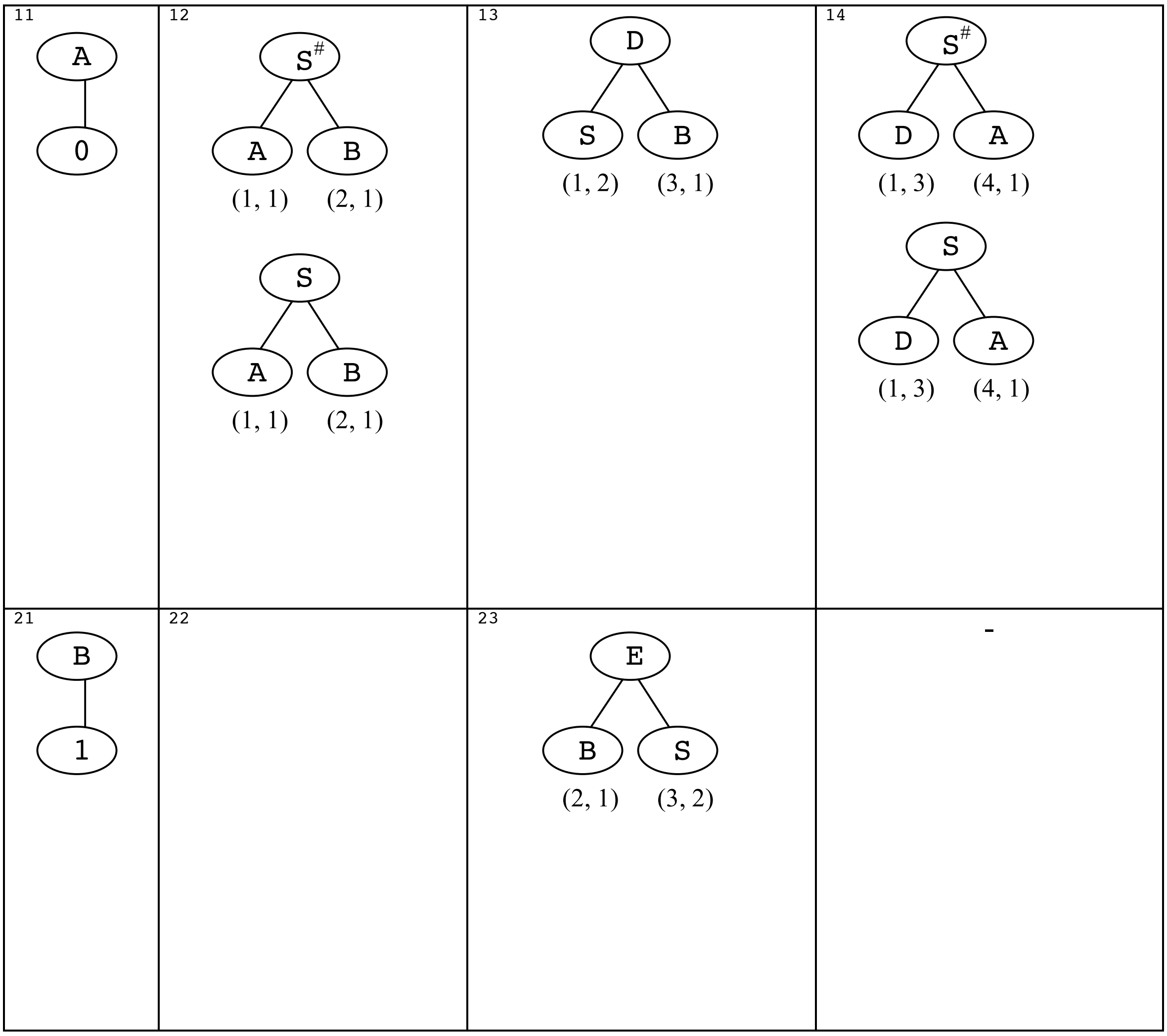
(*b*) Costruire con l'algoritmo CYK tutti gli alberi di derivazione per la frase `0110` di **G**.

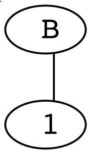
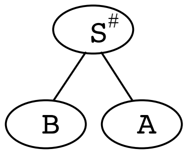
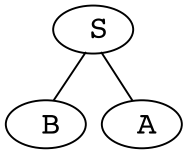
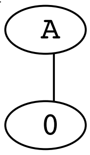
Sia x = `0110`. Le sottostringhe $x_{i,j}$ di x sono di seguito riportate.

11 0	12 01	13 011	14 0110
21 1	22 11	23 110	-
31 1	32 10	-	-
41 0	-	-	-

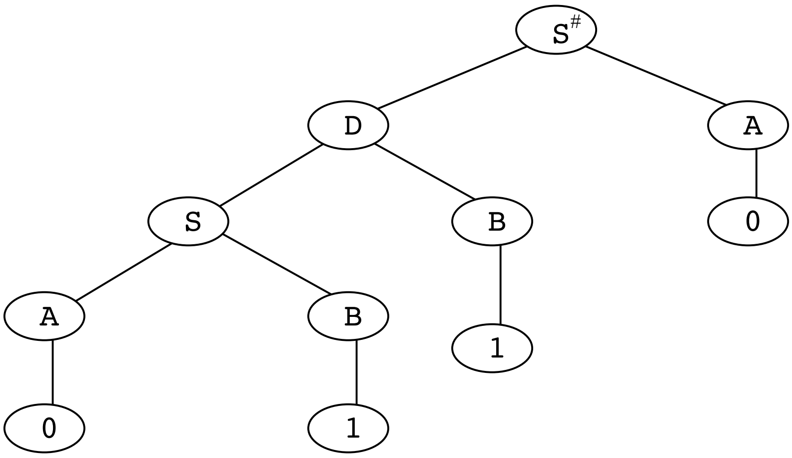
Tabella delle sottostringhe $x_{i,j}$

Con l'algorimo CYK si costruisce la Tabella delle foreste $\mathcal{F}(i, j)$



31 	32  (3, 1) (4, 1)  (3, 1) (4, 1)	-	-
41 	-	-	-

e, sviluppando l'S$^{\#}$-albero di $\mathcal{F}(1, 4)$, si ottiene l'unico albero di derivazione:

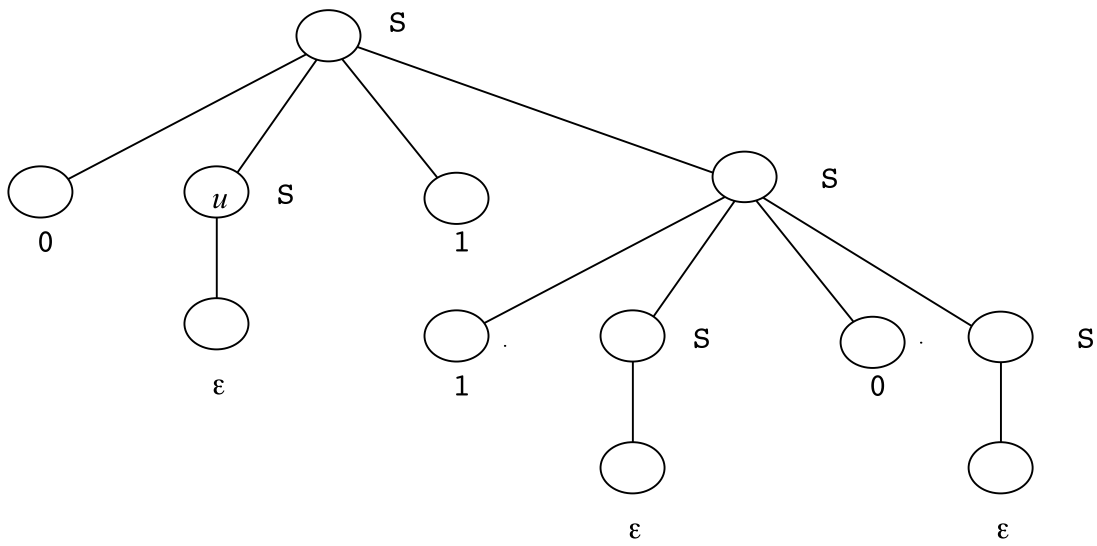


(*c*)	Provare che **G** non appartiene alla classe *LL*(1), ed illustrare il `backtracking` nella costruzione dell'albero di derivazione per la frase `0110` di **G** con il metodo deduttivo.

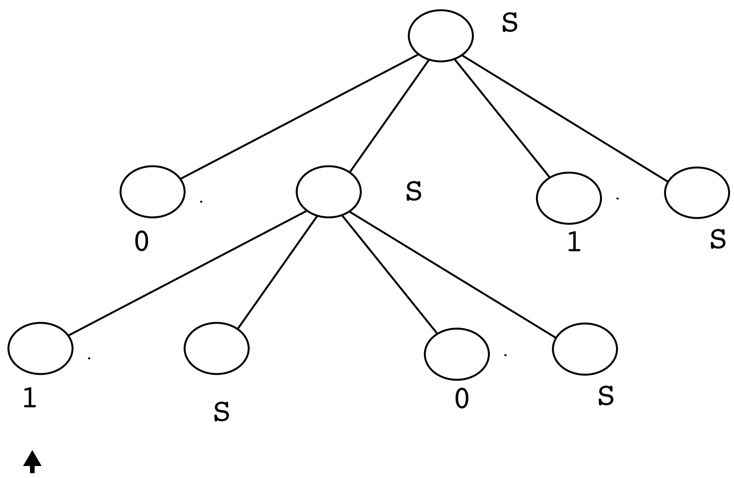
Dopo aver calcolato $\mathcal{I}$(`0S1S`) = {`0`}, $\mathcal{I}$(`1S0S`) = {`1`}, $\mathcal{I}$(ε) = {ε}, e $\mathcal{J}$(`S`) = {`0`, `1`, #}, si costruisce la Tabella di Controllo

	0	1	#
S	0S1S, ε	1S0S, ε	ε

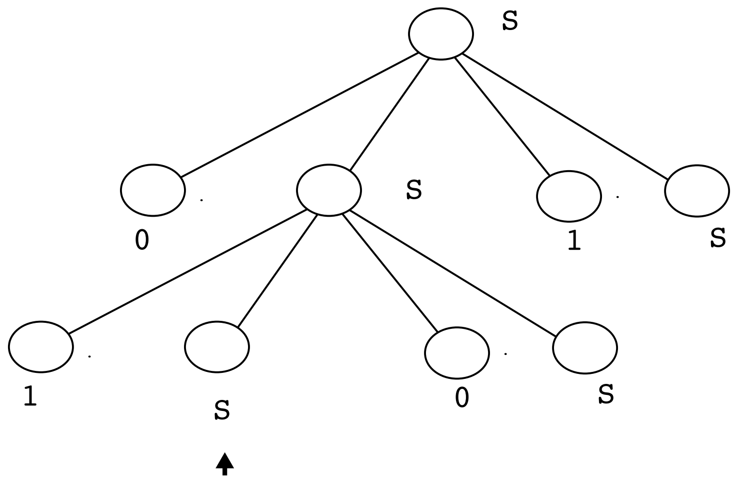
e la presenza di celle doppie dimostra che **G** non appartiene alla classe *LL*(1).
La stringa `0110` ammette un unico albero di derivazione (mostrato in figura)



cosicché, durante la costruzione dell'albero, quando il nodo di controllo è il vertice *u* il simbolo corrente è `1` e la Tabella di Controllo ci consente di utilizzare sia la produzione nulla che la produzione `S` → `1S0S`. Se venisse utilizzata la seconda, avremmo prima l'albero



e poi l’albero



e, solo dopo altri tentativi andati a vuoto, ci accorgeremmo che al nodo *u* andava utilizzata la produzione nulla, la qual cosa obbliga a far arretrare il cursore e a far risalire la sonda.