Linguaggi Formali e Compilatori
(Formal Languages and Compilers)

prof. S. Crespi Reghizzi, prof. Angelo Morzenti
(prof. Luca Breveglieri)

Prova scritta - 8 settembre 2011 - Parte I: Teoria

CON SOLUZIONI - A SCOPO DIDATTICO LE SOLUZIONI SONO MOLTO ESTESE E COMMENTATE VARIAMENTE - NON SI RICHIEDE CHE IL CANDIDATO SVOLGA IL COMPITO IN MODO ALTRETTANTO AMPIO, BENSÌ CHE RISPONDA IN MODO APPROPRIATO E A SUO GIUDIZIO RAGIONEVOLE

NOME:

MATRICOLA: FIRMA:

ISTRUZIONI - LEGGERE CON ATTENZIONE:

- L'esame si compone di due parti:
  - I (80%) Teoria:
    1. espressioni regolari e automi finiti
    2. grammatiche libere e automi a pila
    3. analisi sintattica e parsificatori
    4. traduzione sintattica e analisi semantica
  - II (20%) Esercitazioni Flex e Bison
- Per superare l'esame l'allievo deve sostenere con successo entrambe le parti (I e II), in un solo appello oppure in appelli diversi, ma entro un anno.
- Per superare la parte I (teoria) occorre dimostrare di possedere conoscenza sufficiente di tutte le quattro sezioni (1-4), rispondendo alle domande obbligatorie.
- È permesso consultare libri e appunti personali.
- Per scrivere si utilizzi lo spazio libero e se occorre anche il tergo del foglio; è vietato allegare nuovi fogli o sostituirne di esistenti.
- Tempo: Parte I (teoria): 2h.30m - Parte II (esercitazioni): 45m

1 Espressioni regolari e automi finiti 20%

1. Sono dati i linguaggi regolari L_1 e L_2 seguenti, di alfabeto $\Sigma = \{ a, \ b \}$:

   $$L_1 = (\ a \ a \ \mid \ b \ )^+ \qquad L_2 = (\ b \ b \ \mid \ a \ )^*$$

   Spiegando il procedimento seguito, si costruisca l'automa A che riconosce il linguaggio regolare L_{XOR} seguente:

   $$L_{XOR} = \{ \ x \mid \ \ x \in \Sigma^* \quad \text{and} \quad (\ x \ \in \ L_1 \text{ xor } x \in L_2 \ ) \ \}$$

   dove *xor* è lo *or* exclusivo.

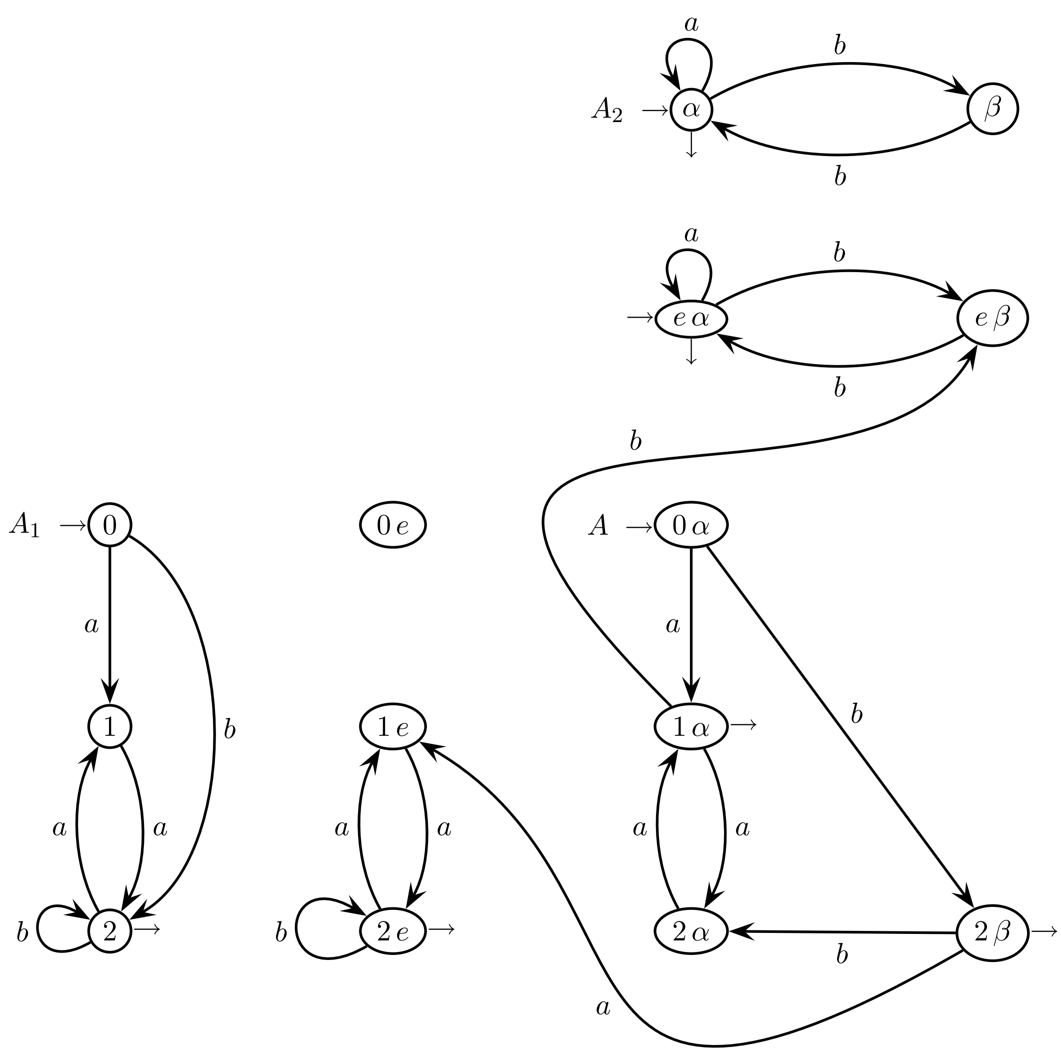
   Esempi:

   $$a \in L_{XOR} \qquad b \in L_{XOR} \qquad a \, b \, b \in L_{XOR} \qquad a \, a \notin L_{XOR}$$

Soluzione

La costruzione è quella del prodotto cartesiano, leggermente modificata. I due automi A_1 e A_2 dei linguaggi L_1 e L_2 lavorano simultaneamente sulla stringa e la coppia di stati dei due automi è rappresentata nello stato della macchina prodotto A.

La regola di riconoscimento richiede che o l'uno o l'altro degli automi riconosca la stringa, ma non entrambi. Di conseguenza quando uno dei due automi A_1 o A_2 entra nello stato d'errore e, l'altro prosegue analizzando la stringa.



L'automa prodotto A non è in forma ridotta poiché ha uno stato inutile (stato $0\,e$), e potrebbe non essere in forma minima (il lettore può verificare da sé).

2. Sono date le due espressioni regolari R_1 e R_2 seguenti:

$$R_1 = a \ (\ b \ b \ \mid \ a \ a \ )^+ \qquad R_2 = a \ (\ a \ b \ )^*$$

Si risponda alle domande seguenti:

(a) Tramite il metodo di Berry-Sethi si costruisca un automa finito A_1 che riconosce il linguaggio regolare $L(R_1)$.

(b) Tramite un metodo qualunque si disegni un automa finito A_2 che riconosce il linguaggio regolare $L(R_2)$ e si dica se è minimo o no, giustificando la risposta.

(c) (facoltativa) Si dimostri che i linguaggi regolari $L(R_1)$ e $L(R_2)$ sono disgiunti ossia che non hanno stringhe in comune, costruendo quanto basta l'automa finito A che riconosce il linguaggio $L(A_1) \cap L(A_2)$ e mostrando che $L(A) = \emptyset$.

Soluzione

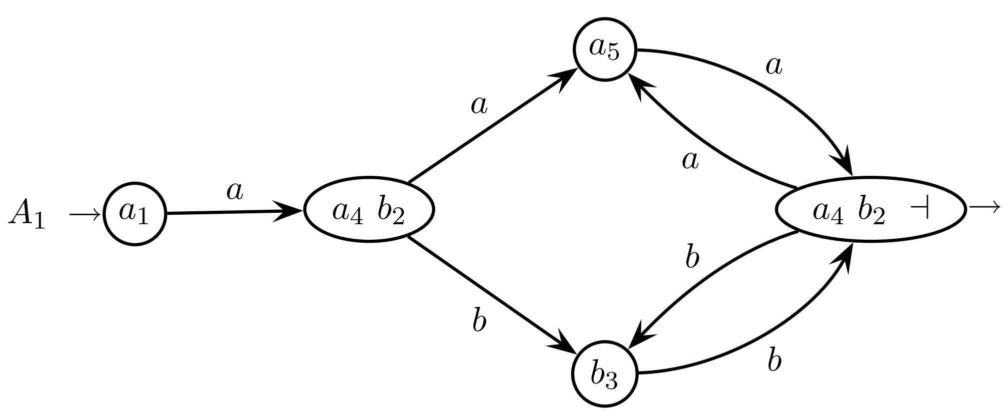
(a) Ecco l'espressione regolare $R_1^{\#}$ numerata e terminata:

$$R_1^{\#} = a_1 \ (\ b_2 \, b_3 \ | \ a_4 \, a_5 \ )^+ \ \dashv$$

Ecco gli insiemi degli inizi e dei seguiti:

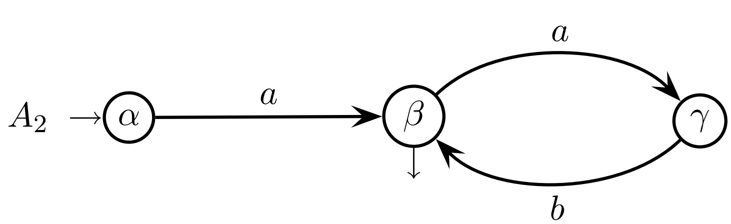
inizi	a_1
generatore	**seguiti**
a_1	$b_2 \ a_4$
b_2	b_3
b_3	$b_2 \ a_4 \ \dashv$
a_4	a_5
a_5	$b_2 \ a_4 \ \dashv$

Ed ecco l'automa deterministico A_1 di Berry-Sethi, con cinque stati:



Si vede subito che l'automa A_1 è già in forma deterministica, ridotta e mimima.

(b) Ecco l'automa A_2 ottenuto intuitivamente:



Si vede subito che l'automa A_2 è anch'esso già in forma ridotta e mimima.

(c) Si potrebbe costruire il prodotto cartesiano per intero e verificare che non c'è nessun cammino tra stato iniziale e finale. Ma basta osservare che nel prodotto c'è un arco iniziale etichettatto con a, poi che segue un secondo arco etichettato con a che porta gli automi A_1 e A_2 negli stati a_5 e γ, e infine che dallo stato prodotto $a_5\ \gamma$, non finale, non esce nessun arco. Pertanto l'automa prodotto non rinonosce nessuna stringa.

2 Grammatiche libere e automi a pila 20%

1. Si consideri il linguaggio di Dyck con parentesi tonde aperta '$($' e chiusa '$)$', generato per esempio dalla grammatica G seguente notoriamente non ambigua:

$$G \begin{cases} S \rightarrow (\, S \,) \, S \\ S \rightarrow \varepsilon \end{cases}$$

All'alfabeto si vuole aggiungere la parentesi quadra chiusa '$]$', la quale può comparire una sola volta nella stringa. La parentesi quadra chiusa funziona come una tonda chiusa e bilancia una corrispondente tonda aperta, ma in più essa bilancia anche tutte le tonde aperte comprese tra la quadra stessa e la corrispondente tonda aperta, le quali eventualmente siamo rimaste sbilanciate.

Esempi che usano la quadra chiusa per bilanciare una o più tonde aperte:

$$(_1 \,]_1 \qquad (_1 \, (_2 \,]_{21} \qquad (_1 \, (_2 \,)_2 \,]_1 \qquad (_1 \, (_2 \, (_3 \,]_{32} \,)_1$$

$$(_1 \, (_2 \,)_2 \, (_3 \,]_{31} \qquad (_1 \, (_2 \, (_3 \,]_{32} \, (_4 \,)_4 \,)_1$$

I pedici numerici mostrano la corrispondenza tra parentesi aperte e chiuse; qui servono per maggiore chiarezza, ma non fanno parte del linguaggio.

Il linguaggio di Dyck così esteso contiene le stringhe con parentesi quadra chiusa descritte sopra e le stringhe di Dyck ordinarie costituite di sole parentesi tonde.

Si risponda alle domande seguenti:

(a) Si scriva una grammatica G', non importa se ambigua, che genera il linguaggio di Dyck esteso descritto sopra (suggerimento: si modifichi e/o estenda la grammatica G senza curarsi se diventa ambigua).

(b) (facoltativa) Si discuta se la grammatica G' è ambigua o no, ed eventualmente se ne scriva una G'' equivalente ma non ambigua.

Soluzione

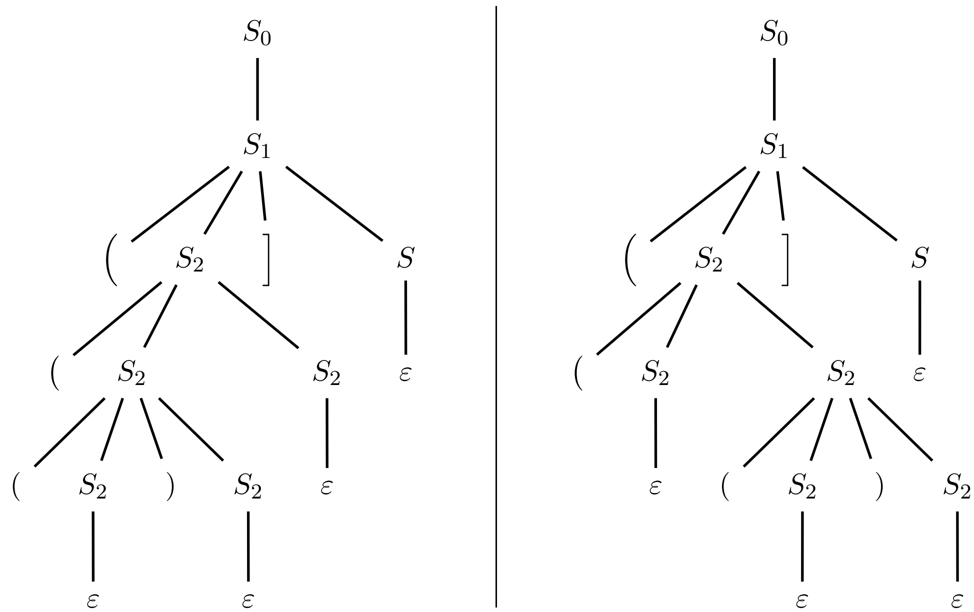
(a) Ecco la grammatica G' (assioma S_0), facilmente ottenuta ampliando e trasformando la grammatica G data nel testo:

$$
G' \left\{
\begin{array}{lcll}
S_0 & \rightarrow & S \mid S_1 & \text{stringhe senza quadra oppure con una sola quadra} \\
\hline
S & \rightarrow & (\; S \;) \; S & \text{stringhe bilanciate senza quadra} \\
S & \rightarrow & \varepsilon & \\
\hline
S_1 & \rightarrow & (\; S_1 \;) \; S \mid (\; S \;) \; S_1 & \text{stringhe con una sola quadra} \\
S_1 & \rightarrow & (\; S_2 \;] \; S & \\
\hline
S_2 & \rightarrow & (\; S_2 \;) \; S_2 \mid (\; S_2 \; S_2 & \text{stringhe sbilanciate senza quadra} \\
S_2 & \rightarrow & \varepsilon &
\end{array}
\right.
$$

La grammatica G' genera tutte le stringhe di Dyck senza parentesi quadra chiusa, tramite le regole che espandono il nonterminale S. In aggiunta essa genera tutte quelle con una sola quadra chiusa, tramite le regole che espandono il nonterminale S_1, e, all'interno della coppia tonda-quadra ( ] che prima o poi comparirà nella stringa, genera, tramite le regole che espandono il nonterminale S_2, tutte le sottostringhe dove le parentesi tonde aperte possono, ma non necessariamente devono, essere sbilanciate ossia prive della tonda chiusa corrispondente. Quest'ultima regola è quella base di Dyck senza la parentesi tonda chiusa.

La grammatica G' è ottenuta semplicemente replicando il gruppo di regole di Dyck G dato nel testo, con le modifiche opportune. Le linee orizzontali separano questi gruppi di regole più o meno modificati e il commento li illustra. Si badi che nonostante il modello G di partenza non sia ambiguo, la grammatica G' potrebbe essere ambigua. Qui però la questione non ha interesse.

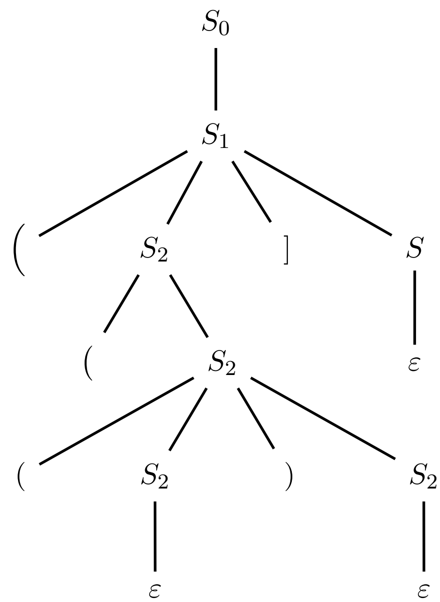
(b) La grammatica G' è palesemente ambigua. Ecco due alberi sintattici per la stessa stringa estesa ((( ]:



Quest'ambiguità si elimina così, accorciando la regola ambigua $S_2 \to ($ S_2 S_2 e ottenendo facilmente la grammatica G''_a (assioma S_0):

$$G''_a \left\{ \begin{array}{ccl} S_0 & \to & S \mid S_1 \\ \hline S & \to & (\ S\ )\ S \\ S & \to & \varepsilon \\ \hline S_1 & \to & (\ S_1\ )\ S \mid (\ S\ )\ S_1 \\ S_1 & \to & (\ S_2\ ]\ S \\ \hline S_2 & \to & (\ S_2\ )\ S_2 \mid (\ S_2 \\ S_2 & \to & \varepsilon \end{array} \right.$$

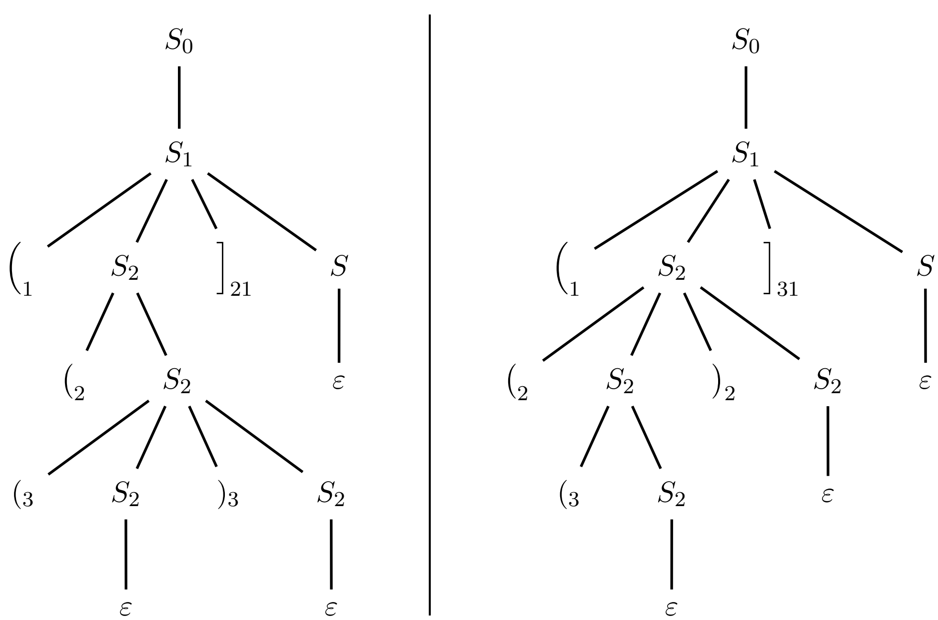
e i due alberi sintattici si unificano così:



Ma resta un'ambiguità più sottile (di origine semantica), che deriva da queste due interpretazioni possibili per la parentesi quadra chiusa:

$$\Big(_1 \ \big(_2 \ (_3 \ )_3 \Big]_{21} \quad \text{oppure} \quad \Big(_1 \ \big(_2 \ (_3 \ \big)_2 \Big]_{31}$$

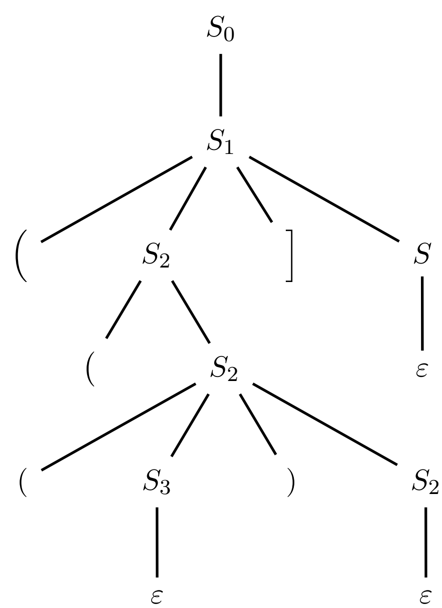
ossia che deriva da se la tonda chiusa bilanci l'aperta numero 3 oppure 2. Ecco i due alberi sintattici, entrambi possibili in G_a''':



Un modo per disambiguare consiste nel rispettare il bilanciamento usuale delle parentesi, basato sulla vicinanza, e dunque consiste nell'imporre che la parentesi quadra chiusa bilanci la tonda aperta numero 2 (oltre alla numero 1), cioè nell'imporre che la tonda chiusa bilanci la tonda aperta che le sta più vicina a sinistra ossia la numero 3; vale a dire nel tenere l'albero sintattico a sinistra e scartare quello a destra. Ecco la soluzione G_b'' (assioma S_0):

$$
G_b'' \left\{
\begin{array}{rcl}
S_0 & \rightarrow & S \mid S_1 \\
\hline
S & \rightarrow & (\ S\ )\ S \\
S & \rightarrow & \varepsilon \\
\hline
S_1 & \rightarrow & (\ S_1\ )\ S \mid (\ S\ )\ S_1 \\
S_1 & \rightarrow & (\ S_2\ ] \\
\hline
S_2 & \rightarrow & (\ S_3\ )\ S_2 \mid (\ S_2 \\
S_2 & \rightarrow & \varepsilon \\
\hline
S_3 & \rightarrow & (\ S_3\ )\ S_3 \\
S_3 & \rightarrow & \varepsilon
\end{array}
\right.
$$

ed ecco l'unico albero sintattico rimasto:

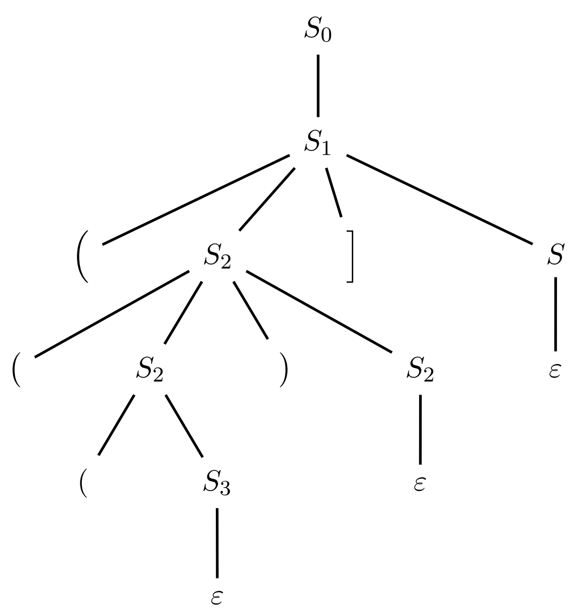


Si noti come dal nonterminale S_3 in giù le parentesi tonde devono essere bilanciate. La grammatica G_b'' non è ambigua poiché l'albero sintattico a destra non può formarsi dato che il nonterminale S_3 genera solo parentesi bilanciate.

Però va detto che si potrebbe anche tenere l'albero sintattico a destra e scartare quello a sinistra. La grammatica genererebbe comunque senza ambiguità il linguaggio di Dyck esteso, seppure con un modo curioso e inusuale di bilanciare le parentesi. Ecco questa possibile soluzione alternativa G_c'' (assioma S_0):

$$G_c'' \left\{ \begin{array}{rcl} S_0 & \rightarrow & S \mid S_1 \\ \hline S & \rightarrow & (\; S \;) \; S \\ S & \rightarrow & \varepsilon \\ \hline S_1 & \rightarrow & (\; S_1 \;) \; S \mid (\; S \;) \; S_1 \\ S_1 & \rightarrow & (\; S_2 \;] \\ \hline S_2 & \rightarrow & (\; S_2 \;) \; S_2 \mid (\; S_3 \\ S_2 & \rightarrow & \varepsilon \\ \hline S_3 & \rightarrow & (\; S_3 \\ S_3 & \rightarrow & \varepsilon \end{array} \right.$$

ed ecco l'unico albero sintattico rimasto:



Si noti come dal nonterminale S_3 in giù tutte le parentesi tonde devono essere sbilanciate. Però è strano che nell'albero sintattico la parentesi tonda chiusa figuri sorella di una aperta lontana invece che di quella adiacente nella stringa. Tuttavia il linguaggio $L(G_c'')$ è identico al linguaggio $L(G_b'')$ e per la domanda la grammatica G_c'' vale quanto la G_b''. Chiaramente se la grammatica G_c'' andasse usata come supporto sintattico per analisi semantica, potrebbe essere strutturalmente inadeguata per tale scopo.

2. Si vuole definire la sintassi della porzione del linguaggio $JSON$ specificata come segue.

value: può essere una string, un number, un object o un array.

string: è una sequenza di zero o più char racchiusi tra doppie virgolette " "; la virgoletta non conta come char.

number: può essere intero, decimale o esponenziale.

Esempi:

$27 \qquad 0.15 \qquad 27e-3 \qquad 3.0e12$

object: è una collezione, racchiusa tra parentesi graffe, di coppie key/value; la key è separata dal value mediante due-punti " : "; le coppie sono separate da virgola; la key è una string; il value è uno qualsiasi dei value di $JSON$.

Esempio:

```
{
   "name"   : "square table" ,
   "format" : {
                 "type"   : "standard" ,
                 "width"  : 150 ,
                 "height" : 80
              }
}
```

array: è una sequenza di value separati da virgola e racchiusa tra parentesi quadre.

Esempi:

```
[ "summer" ,  "autumn" ,  "winter" ,  "spring" ]

[
   [ 0, 2, 1 ] ,
   [ 0, 0, 1 ]
]
```

L'assioma del linguaggio $\langle$JSON_text$\rangle$ genera una sequenza non vuota di value.

Si scriva la grammatica Extended BNF ($EBNF$) per la porzione specificata di $JSON$.

Soluzione

I diagrammi sintattici si trovano in

JSON: The Fat-Free Alternative to XML

Presented at *XML* 2006 in Boston, December 6 by Douglas Crockford

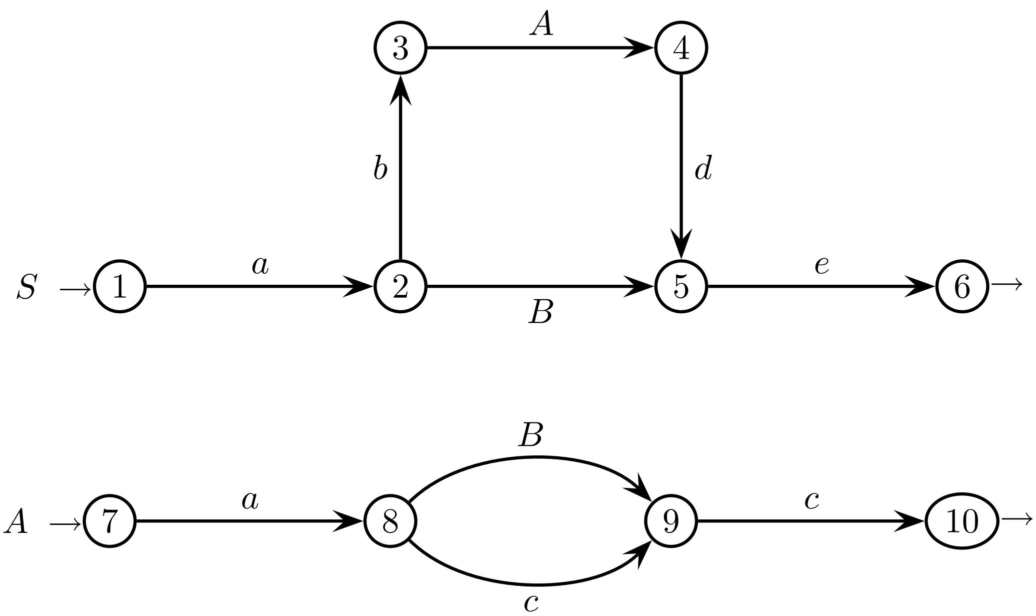
3 Analisi sintattica e parsificatori 20%

1. Si consideri la grammatica G seguente in forma estesa $EBNF$) (assioma S):

$$G \begin{cases} S \rightarrow a \ (\ b \ A \ d \ | \ B \ ) \ e \\ A \rightarrow a \ (\ B \ | \ c \ ) \ d \\ B \rightarrow b \ A \ d \end{cases}$$

   Si risponda alle domande seguenti:

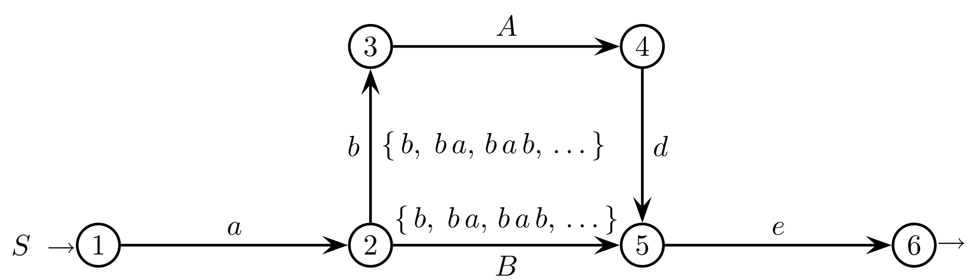
   (a) Si stabilsca se la grammatica è $LL(k)$ per qualche $k \geq 1$, calcolando gli insiemi guida per gli automi di S e A già qui sotto (l'automa di B non è rilevante):

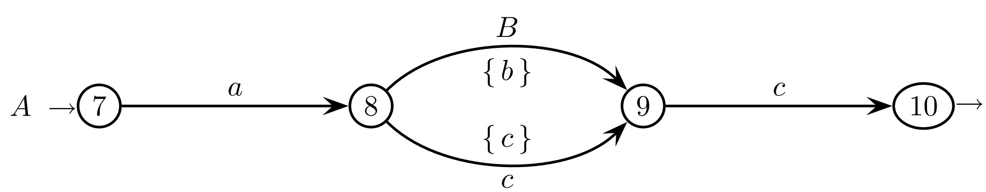


   (b) Se è necessario, si definisca una grammatica G' di tipo $LL(1)$ equivalente a G.

Soluzione

(a) Ecco gli insiemi guida rilevanti (o parte di essi):





La grammatica G non è $LL(k)$ per alcun $k \geq 1$ poiché gli insiemi guida di qualsiasi lunghezza sulla diramazione nell'automa S non sono disgiunti.

(b) Analizzando il linguaggio, si trova quanto segue:

$$L(G) = \left\{ \ a \ \ (\ b \ a \ )^n \ \ c \, d^{2\,n} \ e \ | \quad n > 1 \ \right\}$$

Il linguaggio ammette quindi la seguente grammatica G' di tipo $LL(1)$:

$$G' \begin{cases} S & \rightarrow \quad a \ b \ a \ D \ d \ d \ e \\ D & \rightarrow \quad b \ a \ D \ d \ d \ \mid \ c \end{cases}$$

come si vedrebbe subito tracciandone gli automi.

2. Si consideri la grammatica G seguente (assioma S), che è ambigua:

$$G \begin{cases} S \rightarrow a\ S\ b\ S \\ S \rightarrow a\ S \\ S \rightarrow \varepsilon \end{cases}$$

Per esempio, la stringa $a\ a\ b$ generata da G ha due alberi sintattici differenti.

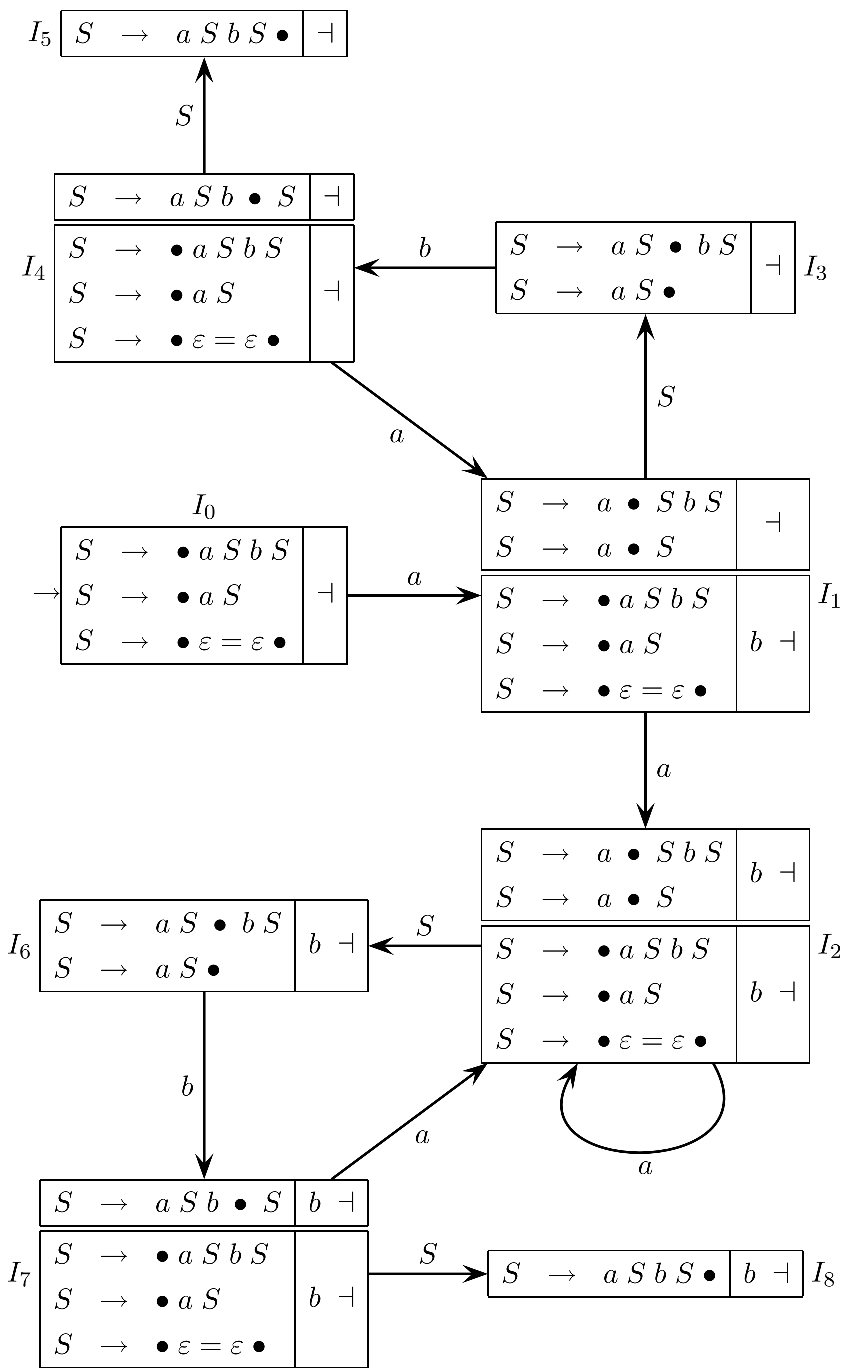
Si risponda alle domande seguenti:

(a) Si tracci per intero il grafo pilota $LR(1)$ di G e si indichino tutti i conflitti, certamente presenti poiché G è ambigua e dunque non è di tipo $LR(1)$.

(b) (facoltativa) Si dica qual è il primo conflitto che si incontra analizzando la stringa ambigua $a\ a\ b$ e qual è il contenuto della pila dell'analizzatore nel momento del conflitto.

Soluzione

(a) Ecco il grafo pilota $LR(1)$ completo della grammatica G:



C'è un solo conflitto di tipo riduzione-spostamento nel macrostato I_6: riduzione $S \to a\ S\ \bullet$ con prospezione b e spostamento da $S \to a\ S\ \bullet\ b\ S$ a $S \to a\ S\ b\ \bullet\ S$ leggendo b. Non ci sono altri conflitti.

(b) Ecco la tabella delle mosse analizzando la stringa $a\ a\ b$, con il contenuto dell'ingresso e della pila:

	ingresso iniziale	**pila iniziale**
–	$a\ a\ b$	I_0
mossa dell'analizzatore	**ingresso post mossa**	**pila post mossa**
spostamento : $I_0 \xrightarrow{a} I_1$	$a\ b$	$I_0\ I_1$
spostamento : $I_1 \xrightarrow{a} I_2$	b	$I_0\ I_1\ I_2$
riduzione : $S \to \varepsilon$	$S\ b$	$I_0\ I_1\ I_2$
spostamento : $I_2 \xrightarrow{S} I_6$	b	$I_0\ I_1\ I_2\ I_6$
in I_6 c'è conflitto tra riduzione $S \to a\ S$ e spostamento $I_6 \xrightarrow{b} I_7$		

Dopo la riduzione $S \to \varepsilon$, nell'ingresso viene fittiziamente scritto il simbolo S che rappresenta l'avvenuta riduzione e che viene subito letto dallo spostamento su S che necessariamente segue la riduzione. Questo artificio serve semplicemente per maggiore chiarezza.

Pertanto il primo conflitto che si incontra nell'analisi è quello nel macrostato I_6 e il corrispondente contenuto della pila è $I_0\ I_1\ I_2\ I_6$.

Il conflitto nel macrostato I_6 riflette l'ambiguità della stringa in analisi: se tramite la regola $S \to a\ S\ b\ S$, mettere la lettera b finale in corrispondenza con la prima o la seconda lettera a della stringa $a\,a\,b$.

4 Traduzione e analisi semantica 20%

1. È data la seguente grammatica G del linguaggio di Dyck (assioma S):

$$G \begin{cases} 1: & S \rightarrow a\ S\ a'\ S \\ 2: & S \rightarrow b\ S\ b'\ S \\ 3: & S \rightarrow \varepsilon \end{cases}$$

   Per calcolare la derivazione sinistra delle frasi è dato il seguente schema sintattico di traduzione G_t (assioma S):

$$G_t \begin{cases} S \rightarrow \frac{a}{1}\ S\ a'\ S \\ S \rightarrow \frac{b}{2}\ S\ b'\ S \\ S \rightarrow \frac{\varepsilon}{3} \end{cases}$$

   Per esempio, la traduzione della stringa $x = a\ b\, b'\ a\, a'\ a'$ è la stringa $s(x) = 1\,2\,3\,1\,3\,3\,3$, dove le etichette delle regole rappresentano la derivazione di x nell'ordine canonico sinistro.

   Si risponda alle domande seguenti:

   (a) Si scriva uno schema di traduzione analogo G'_t per calcolare la forma *riflessa* $d(x)^R$ della derivazione canonica *destra* $d(x)$ della stringa data $x \in L(G)$. Per esempio, per $x = a\ b\, b'\ a\, a'\ a'$ si ha $d(x)^R = 3\,3\,3\,1\,2\,3\,1$.

   (b) (facoltativa) Sia L_s il linguaggio delle derivazioni sinistre sopra descritte, ossia:

$$L_s = \{\ y \mid \quad y = s(x) \in \{\ 1,\ 2,\ 3\ \}^+ \quad \text{and} \quad x \in L(G)\ \}$$

   Esso è generato dalla grammatica G_s seguente (assioma S):

$$G_s \begin{cases} S \rightarrow 1\ S\ S \\ S \rightarrow 2\ S\ S \\ S \rightarrow 3 \end{cases}$$

   Si scriva una schema di traduzione sintattico G'_s per tradurre tali stringhe nelle corrispondenti derivazioni destre $d(x)$ (non riflesse). Per esempio, la traduzione di $1\,2\,3\,1\,3\,3\,3$ è $1\,3\,2\,1\,3\,3\,3$.

Soluzione

(a) Lo schema di traduzione G'_t per calcolare la derivazione destra riflessa è il seguente (assioma S):

$$G'_t \begin{cases} S & \rightarrow & a\ S\ a'\ S\ \frac{\varepsilon}{1} \\ S & \rightarrow & b\ S\ b'\ S\ \frac{\varepsilon}{2} \\ S & \rightarrow & \frac{\varepsilon}{3} \end{cases}$$

Esso emette a destra il numero della regola utilizzata nella derivazione.

(b) Lo schema di traduzione G'_s che traduce da L_s alle derivazioni destre è il seguente (assioma S):

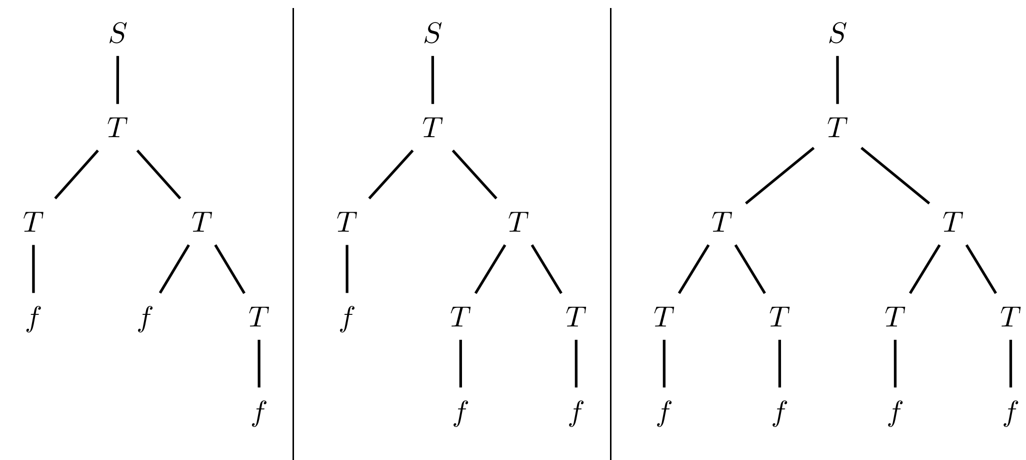
$$G'_s \begin{cases} S & \rightarrow & \frac{1}{\varepsilon}\ S\ S\ \frac{\varepsilon}{1} \\ S & \rightarrow & \frac{2}{\varepsilon}\ S\ S\ \frac{\varepsilon}{2} \\ S & \rightarrow & \frac{3}{3} \end{cases}$$

Esso traduce la numerazione della regola, data a sinistra, nelle stessa numerazione ma emessa a destra.

2. Si consideri il supporto sintattico G seguente (assioma S):

$$G \begin{cases} S \;\rightarrow\; T \\ T \;\rightarrow\; T\,T \;\mid\; f\,T \;\mid\; T\,f \;\mid\; f \end{cases}$$

Ecco tre esempi di albero sintattico di G:



(a) (b) (c)

Si risponda alle domande seguenti, ricorrendo agli attributi nelle tabelle predisposte:

(a) Definizione: l'albero è *completo* se ogni nodo T ha zero o due figli di tipo T; l'albero (a) non è completo, mentre gli alberi $(b,\, c)$ lo sono.

Domanda: si progetti una grammatica con attributi per stabilire se l'albero sintattico è *completo*.

(b) (facoltativa) Definizioni:

- l'*altezza* dell'albero è il numero di livelli che contengono nodi T; gli alberi $(a,\, b,\, c)$ hanno altezza 3
- la *profondità* di un nodo T è la sua distanza dal nodo T figlio della radice S; negli alberi $(a,\, b,\, c)$ la profondità dei nodi T varia tra 0 e 2
- l'*indice di riempimento* R dell'albero t è la quantità seguente:

$$R(t) = \frac{\sum_{\text{nodi } n \text{ di tipo } T} 2^{-profondita(n)}}{altezza(t)}$$

esempio: $R(a) = \frac{2^0+2^{-1}+2^{-1}+2^{-2}}{3} = \frac{2{,}25}{3} = \frac{3}{4} = 0,75$

Osservazione: si dice che l'albero è *pieno* se è completo e tutti i nodi T con un solo figlio f si trovano allo stesso livello; l'albero (c) è pieno, gli alberi $(a,\, b)$ no; l'indice R varia tra 0 e 1, e vale 1 se e solo se l'albero è pieno; p. es. $R(c) = 1$.

Domanda: si progetti una grammatica con attributi per calcolare l'*indice di riempimento* R dell'albero sintattico.

attributi da usare per la grammatica - domanda (a)

tipo	nome	(non)terminali	dominio	significato
sin	c	S, T	booleano	vero se e solo se il sottoalbero corrente è completo

attributi da usare per la grammatica - domanda (b)

tipo	nome	(non)terminali	dominio	significato
sin	a	T	intero	altezza del sottoalbero corrente
des	p	T	intero	profondità di un nodo di tipo T
des	v	T	reale	si calcola come $v = 2^{-p}$ e indica il valore associato a un nodo T che si trova a profondità p
sin	s	T	reale	indica la somma dei valori v di tutti i nodi T contenuti nel sottoalbero corrente
sin	R	S	reale	indice di riempimento dell'intero albero

sintassi	calcolo attributi - domanda (a)
$S_0 \rightarrow T_1$	
$T_0 \rightarrow T_1\ T_2$	
$T_0 \rightarrow f\ T_1$	
$T_0 \rightarrow T_1\ f$	
$T_0 \rightarrow f$	

sintassi	calcolo attributi - domanda (b)
$S_0 \rightarrow T_1$	
$T_0 \rightarrow T_1\ T_2$	
$T_0 \rightarrow f\ T_1$	
$T_0 \rightarrow T_1\ f$	
$T_0 \rightarrow f$	

Soluzione

(a) Ecco le funzioni sematiche (c'è un solo attributo ed è sintetizzato):

sintassi	calcolo attributi - domanda (a)
$S_0 \to T_1$	$c_0 = c_1$
$T_0 \to T_1\ T_2$	$c_0 = c_1 \ \wedge \ c_2$
$T_0 \to f\ T_1$	$c_0 = false$
$T_0 \to T_1\ f$	$c_0 = false$
$T_0 \to f$	$c_0 = true$

(b) Ecco le funzioni sematiche (con attributi ereditati e sintetizzati):

sintassi	calcolo attributi - domanda (b)	
$S_0 \to T_1$	$p_1 = 0$ $v_1 = 1$	- ereditati (o se si preferisce $v_1 = 2^{-p_1}$)
	$R_0 = s_1/a_1$	- sintetizzato
$T_0 \to T_1\ T_2$	$p_1 = p_0 + 1$ $p_2 = p_0 + 1$ $v_1 = 2^{-p_1}$ $v_2 = 2^{-p_2}$	- ereditati
	$s_0 = s_1 + s_2 + v_0$ $a_0 = \max(\ a_1,\ a_2\ ) + 1$	- sintetizzati
$T_0 \to f\ T_1$	$p_1 = p_0 + 1$ $v_1 = 2^{-p_1}$	- ereditati
	$s_0 = s_1 + v_0$ $a_0 = a_1 + 1$	- sintetizzati
$T_0 \to T_1\ f$	$p_1 = p_0 + 1$ $v_1 = 2^{-p_1}$	- ereditati
	$s_0 = s_1 + v_0$ $a_0 = a_1 + 1$	- sintetizzati
$T_0 \to f$	$s_0 = v_0$ $a_0 = 1$	- sintetizzati