SOLUZIONI
Linguaggi Formali e Compilatori
Prof. Crespi Reghizzi
Prova scritta[1]
10/02/2004

Revisione 25.01.2005

	punti %	annotazioni	VOTO
1. Espressioni regolari e automi finiti			
2. Grammatiche			
3. Laboratorio Flex Bison			
4. Grammatiche e analisi sintattica			
5. Traduzione e semantica			
VOTO			

Per superare la prova è l'allievo deve dimostrare la conoscenza di tutte e 5 le parti.

1 Espressioni regolari e automi finiti 20%

1. Progetto di automa finito
   Alfabeto terminale $\{[,], (,)\}$. Una frase è una stringa non vuota, ben parentesizzata che soddisfa le seguenti condizioni:

   (a) La profondità di annidamento non supera tre.

   (b) Numerati i livelli 1, 2, 3 dall'esterno all'interno, le parentesi tonde occupano sempre e soltanto il livello 2.

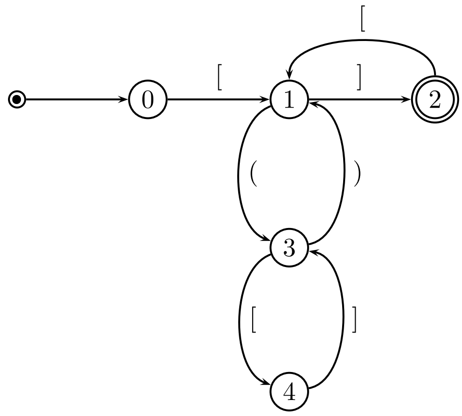
   Esempi: [] , [()] , [()([])][] , [][][()]
   Controesempi: () , [([[]])] , ([])

   (a) Si disegni l'automa

   (b) Si verifichi se l'automa è minimo.

   Soluzione

[1] Tempo 2 ore 30'. Libri e appunti personali possono essere consultati. È consentito scrivere a matita. Scrivere il proprio nome sugli eventuali fogli aggiuntivi.



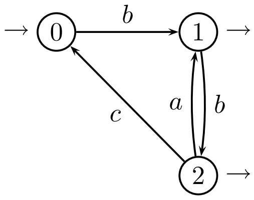
Criteri di valutazione:

(a) L'automa riconosce il ling. specificato?
   i. l'automa accetta soltanto le stringhe valide?
   ii. L'automa accetta tutte le stringhe valide?

(b) Minimalità dell'automa

(c) Qualità della presentazione.

2. Calcolare l'espressione regolare del ling. riconosciuto dall'automa seguente.

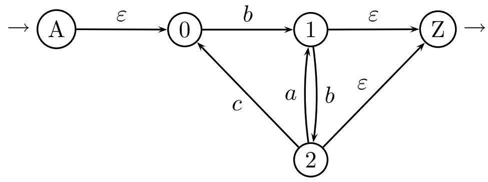


Mostrare i passaggi.

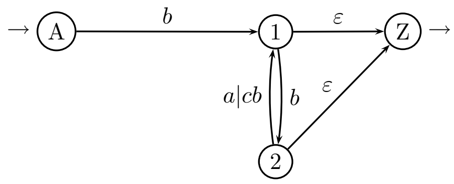
Soluzione

L'espr. reg. si può calcolare con diversi metodi, quello di eliminazione o la risoluzione delle equazioni insiemistiche. Ecco i passi del primo metodo:

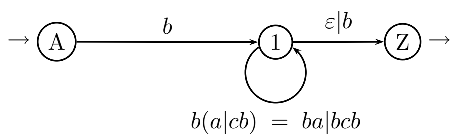
(a) Aggiunta di un nuovo stato iniziale e finale:



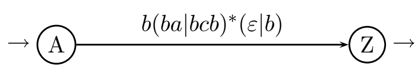
(b) Eliminazione del nodo 0



(c) Eliminazione del nodo 2



(d) Eliminazione del nodo 1



Criteri di valutazione

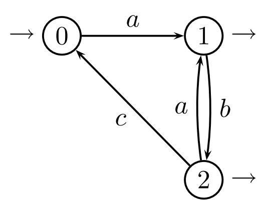
(a) Scelta del procedimento appropriato

(b) Applicazione corretta e intelligente del procedimento

(c) Capacità di scoprire eventuali errori di calcolo attraverso l'analisi critica del risultato finale

(d) Qualità della presentazione.

3. Per il ling. speculare $(L(A))^R$, dove A è il seguente automa, costruire l'automa deterministico riconoscitore, spiegando il procedimento applicato.

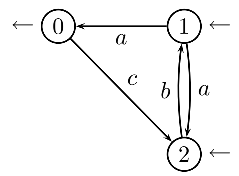


Soluzione. Si possono seguire procedimenti diversi:

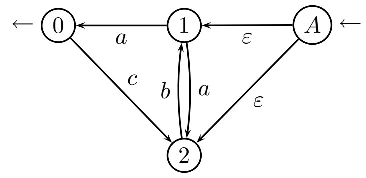
1. Si trasforma direttamente l'automa nel riconoscitore di $(L(A))^R$, poi lo si determinizza.
2. Si calcola la espressione regolare di $L(A)$ (ad es. con il metodo di eliminazione); poi da questa l'espressione di $(L(A))^R$, e infine il riconoscitore deterministico, con il metodo di Mc Naughton e Yamada.

Soluzione con il primo procedimento:

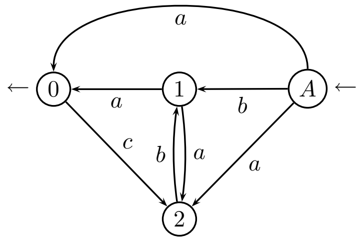
1. Inversione delle frecce e scambio tra stato iniziale e stati finali:



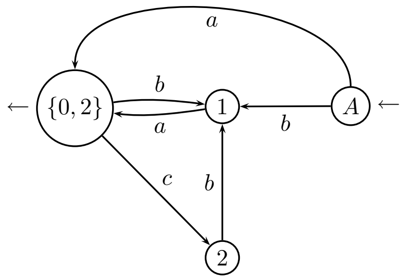
2. Aggiunta di un nuovo stato iniziale, unico:



3. Determinizzazione; prima fase, eliminazione mosse spontanee:

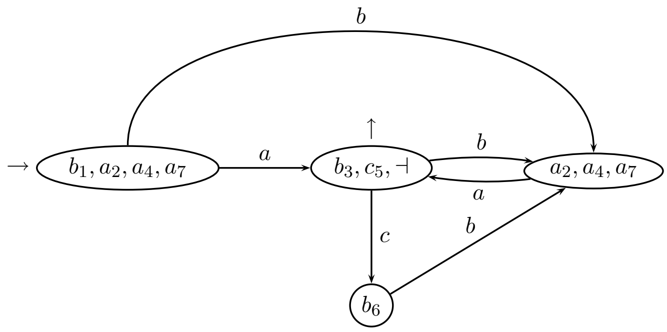


4. Determinizzazione; seconda fase, costruzione delle parti finite dell'insieme degli stati:



Soluzione con il secondo procedimento:

1. L'espressione regolare di $L(A)$, calcolata come nel problema precedente, è
$$a(ba|bca)^*(\varepsilon|b)$$
2. L'espressione del ling. speculare si ottiene riflettendo i termini concatenati dell'espressione :
$$(\varepsilon|b)(ab|acb)^*a$$
3. Numerata l'espressione
$$(\varepsilon|b_1)(a_2b_3|a_4c_5b_6)^*a_7$$
si costruisce infine l'automa deterministico:



Criteri di valutazione

1. Capacità di immaginare un percorso risolutivo appropriato
2. Esecuzione accurata e completa del procedimento
3. Capacità di scoprire eventuali errori di calcolo attraverso l'analisi critica del risultato finale
4. Qualità della presentazione.

2 Grammatiche 20%

1. Progettare una grammatica per definire un ling. di programmazione avente i seguenti costrutti.

   - chiamate di procedure con almeno un argomento
     `call nome1(arg1, arg2, ...)`
     Un argomento può essere una espressione aritmetica ad es.
     `-(3+p1)*p2`
     con gli operatori di somma, prodotto e con il segno meno unario.
   - dichiarazioni di procedure
     `proc n1(p1, p2, ...)begin call n2(3,p2+p1) call n4(-p2)end n1`
     L'identificatore della procedura è ripetuto dopo `end`. Il corpo della procedura contiene una o più chiamate. Prima del corpo ci può essere una lista di dichiarazioni di variabili, che possono essere inizializzate con una espressione aritmetica costante:
     `proc n1(p1, p2, ...) int i1, i2=(3+5)*2 begin ....end n1`
   - il programma inizia con le dichiarazioni delle procedure, seguite dalle chiamate:

     ```
     program proc n1 ... end n1 proc n2 ...end n2
     ...call ...call...
     end.
     ```

   (a) Scrivere le regole usando la forma EBNF

   (b) Disegnare un albero sintattico

Soluzione:

$S \to$ program D^+C^+ end $\bullet$

- - D dichiarazioni, C chiamate

$D \to$ proc $I\ ''(''I(,I)^*\ '')''\ [V]$ begin C^+ end I

- - I identificatore, V dich. di variabili

$C \to$ call $I''(''E(,E)^*\ '')''$
$E \to T(+T)^*$
$T \to F(\times F)^*$
$F \to [-]''(''E'')'' \mid [-](I \mid N)$

- - N numero

$V \to$ int $A(,A)^*$
$A \to I[= E_c]$

- - E_c espressione costante

$E_c \to T_c(+T_c)^*$
$E_c \to T_c(\times T_c)^*$

Criteri di valutazione

(a) Ling. generato è corretto
   i. non genera stringhe estranee:
   ii. non perde delle frasi
(b) Chiarezza e buona struttura della grammatica
   i. non ha inutili ridondanze
   ii. non è un tentativo a caso ma ha una struttura ragionata.
   iii. non è ambigua.
(c) Qualità della presentazione.

2. Grammatica e automa a pila
   Il ling. da definire, di alfabeto $\{a, b\}$, è

$$L = \{x \mid |x|_a = |x|_b + 1 \geq 1\}$$

   (a) Scrivere la grammatica
   (b) Definire la funzione di transizione di un automa a pila che riconosce il linguaggio

Soluzione:

$$L = L_X a L_X \text{ dove}$$

$$L_X = \{x \mid |x|_a = |x|_b \geq 0\}$$

Grammatica G:

$$\begin{array}{ll} 1 & S \to XaX \\ 2 & X \to aXbX \\ 3 & X \to bXaX \\ 4 & X \to \varepsilon \end{array}$$

Giustificazione:
Osserviamo che X genera solo stringhe con $|x|_a = |x|_b$. Tali stringhe possono essere viste come una generalizzazione del ling. di Dyck con parentesi a, b, in cui le parentesi possono essere nell'ordine a, b o b, a. Ad es. in L_X sta la stringa:

$$\underbrace{b\,\underbrace{b\,\underbrace{ab}\,a}\,a}_{X}$$

Confrontiamo ora il ling. desiderato con quello generato da G.

$L(G) \subseteq L$ Infatti la frase più corta $a \in L$. La prima regola allunga una frase x mantenendo invariante la differenza $|x|_a - |x|_b$.

$L \subseteq L(G)$ Sia $x \in L$, si mostra che $S \stackrel{+}{\Rightarrow} x$.
Necessariamente x è il concatenamento di una stringa del tipo X seguite da a e poi da una stringa del tipo X, come ad es.:

$$\underbrace{\underbrace{aabb}\underbrace{ba}\underbrace{ba}\underbrace{bbbaaa}}_{X}a$$

$$\underbrace{ab}_{X}\ a\ \underbrace{\underbrace{aabb}\,\underbrace{aaabbb}}_{X} = a\ \underbrace{\underbrace{ab}\,\underbrace{aabb}\,\underbrace{aaabbb}}_{X} \qquad \text{ambigua}$$

$$\underbrace{b\,\underbrace{ab}\,a}_{X}\ a\ \underbrace{aabb}_{X}$$

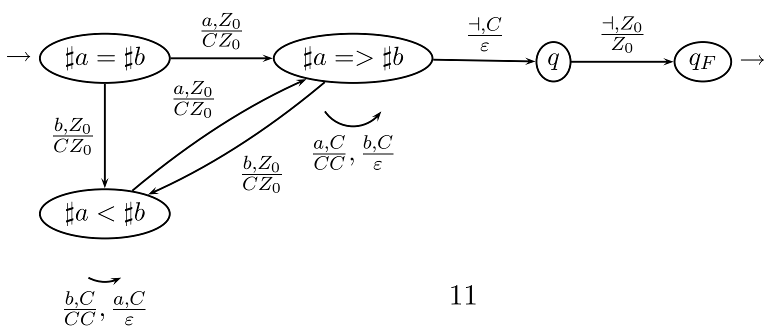
e tali stringhe sono generate da G.

Criteri di valutazione

1. Ling. generato è corretto
   (a) non genera stringhe estranee:
   (b) non perde delle frasi
2. Chiarezza e buona struttura della grammatica
   (a) non ha inutili ridondanze
   (b) non è un tentativo a caso ma ha una struttura ragionata.
3. Qualità della presentazione e dei ragionamenti giustificativi.

Automa a pila. Vi sono due modi per costruirlo. Partendo dalla grammatica si può applicare il procedimento spiegato a pag. 99 del libro. Oppure si può costruire direttamente l'automa ragionando sul linguaggio, la via che ora seguiremo.

La pila viene usata come un contatore del valore assoluto della differenza $|x|_a - |x|_b$, mentre lo stato $\sharp a > \sharp b$ oppure $\sharp a < \sharp b$ indica il segno. L'automa può leggere più volte il terminatore $\dashv$.



Criteri di valutazione

1. validità del metodo formale, se scelto per costruire l'automa
2. oppure buona comprensione intuitiva del funzionamento della pila e descrizione precisa e completa delle mosse dell'automa
3. qualità della presentazione e dei ragionamenti giustificativi.

3 Domanda relativa alle esercitazioni 20%

Considerate l'implementazione del compilatore *Simple*, fornita integralmente nelle pagine che seguono, e contenente le aggiunte relative al costrutto `for`, sviluppate per il tema d'esame scorso. Modificatela in modo che venga riconosciuta l'istruzione `break` nei cicli `for`. Assumete per `break` la stessa semantica che essa ha nel linguaggio C. Il compilatore da voi modificato deve riconoscere il costrutto e generare una traduzione corretta nel linguaggio assembly della macchina *SimpleVM*. Un esempio banale di uso della `break` segue:

```
for (c:=1; c<20; c:=c+1)
do
  write c;
  if c=10 then break;
  else fi;
od;
```

Notate che in *Simple*, è legale annidare i cicli, quindi `break` deve saltare al termine del ciclo `for` corretto.

Per vostra utilità vi forniamo un contenitore generico di puntatori, che potete usare sia come pila (push, pop, top) che come array dinamico o coda (add, get-size, get-at). Il contenitore puo' esservi utile per memorizzare dati di contesto o i punti dove effettuare il backpatching.

- `void * gpc_top(GenericPtrContainer pc);`
  restituisce l'elemento al top della pila;
- `void   gpc_push     (GenericPtrContainer * pc, void * ptr);`
  impila un nuovo elemento;
- `void   gpc_pop      (GenericPtrContainer * pc);`
  disimpila un elemento;
- `void   gpc_add      (GenericPtrContainer * pc, void * ptr);`
  aggiunge un elemento in coda;
- `int    gpc_get_size (GenericPtrContainer pc);`
  restituisce la lunghezza della coda;
- `void * gpc_get_at   (GenericPtrContainer pc, int i);`
  restituisce l'elemento i-esimo;
- `GenericPtrContainer var = {NULL,0,0};`
  dichiarazione di un contenitore inizialmente vuoto di nome *var*;

***la soluzione sarà diffusa prossimamente

4 Grammatiche e analisi sintattica 20%

1. Progetto di grammatica adatta all'analisi deterministica
   Alfabeto terminale: $\{<, >, [,], \sharp\}$. Si considerino le stringhe ben parentesizzate uncinate e quadre, di alfabetto rispettivo $<, >$ e $[,]$. Il ling. da definire contiene le liste di stringhe ben parentesizzate alternativamente del tipo uncinato e quadro, con il diesis come separatore. La prima componente deve essere uncinata. Esempi:

   $$<>\ \sharp[\,] \quad , \quad <>\ \sharp[\,]\sharp <<>> \quad , \quad <<><>>\ \sharp[\,]\sharp <>\ \sharp[[\,][\,][\,]]$$

   Si precisa che al posto di una stringa parentesizzata quadra può stare la stringa vuota, mentre le stringhe uncinate non possono essere vuote. Pertanto sono valide anche le stringhe

   $$<>\ \sharp \quad , \quad <<><>>\ \sharp\ \sharp <>\ \sharp[[\,][\,][\,]]$$

   Invece sono scorrette le stringhe

   $$<<><>>\ \sharp[\,]\sharp\ \sharp[[\,][\,][\,]] \quad , \quad \sharp[[\,][\,][\,]] \quad , \quad [\,]\sharp <>$$

   Si deve progettare una grammatica adatta, a scelta dell'allievo, all'analisi LL oppure LR.

   (a) Si progetti la grammatica BNF (non è consentita la forma BNF estesa)
   (b) Si verifichi, mostrando i passi del procedimento, se la grammatica è LL opp. LR.
   (c) Se necessario si modifichi la grammatica per ottenere la proprietà scelta.

   Soluzione

   Siano D_U (risp. D_Q) i ling. di Dyck, anche vuoti, con parentesi uncinate (risp. quadre). Sia $\overline{D}_U$ il ling. di Dyck non vuoto, con parentesi uncinate.

   Una frase inizia con $\overline{D}_U$. Poi vi può essere una sequenza di D_Q e di $\overline{D}_U$ alternati e separati dal diesis, ossia una espressione:

   $$\Big((D_Q\sharp\overline{D}_U)^+(D_Q|\varepsilon)\Big)|\varepsilon$$

L'ultimo elemento può essere sia $\overline{D}_U$ che D_Q (come mostra il secondo esempio).

Di qui si costruisce la grammatica:

$S \to \overline{D}_U Q$	
$Q \to \sharp D_Q U$	$\sharp$
$Q \to \varepsilon$	$\dashv$
$U \to \sharp \overline{D}_U Q$	$\sharp$
$U \to \varepsilon$	$\dashv$
$D_Q \to [D_Q] D_Q$	$[$
$D_Q \to \varepsilon$	$], \dashv, \sharp$
$\overline{D}_U \to < D_U > \overline{D}_U$	$<$ Non LL(1)
$\overline{D}_U \to < D_U >$	$<$ Non LL(1)
$D_U \to < D_U > D_U$	$<$
$D_U \to \varepsilon$	$>$

i cui insiemi guida sono riportati a fianco. Non essendo essi disgiunti per le alternative di $\overline{D}_U$, la grammatica non è LL(1). Applichiamo la fattorizzazione sinistra alle alternative che violano la condizione LL(1)e calcoliamo poi gli insiemi guida:

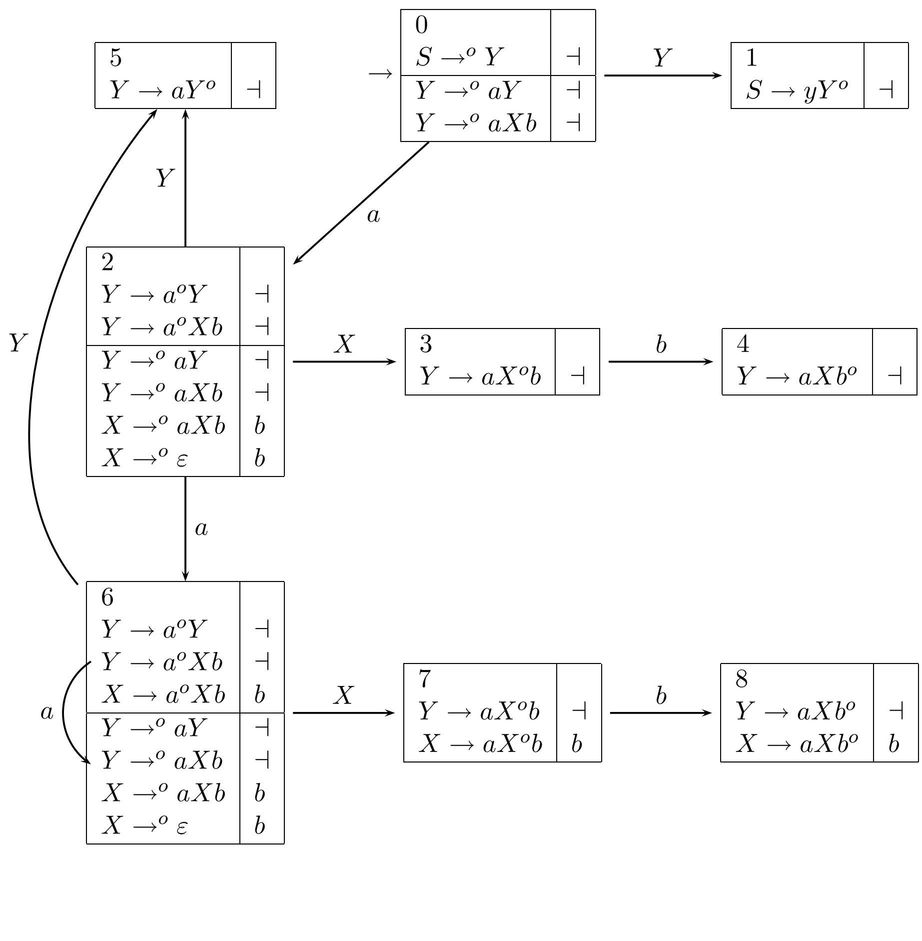
$\overline{D}_U \to < D_U > X$	
$X \to \overline{D}_U$	$<$
$X \to \varepsilon$	$\sharp, \dashv$

La grammatica è LL(1).

2. Grammatica LR(1)
   Per la seguente grammatica:
   $S \to Y$
   $Y \to aY \qquad Y \to aXb$
   $X \to aXb \qquad X \to \varepsilon$

   (a) Si costruisca il riconoscitore dei prefissi ascendenti di G
   (b) Se necessario si trasformi la grammatica per ottenere una grammatica LR(1).

Soluzione:



La grammatica è LR(1). Infatti, nello stato 2 la riduzione richiede propsezione b, né alcun arco uscente ha tale etichetta. Lo stato 7 contiene due candidate di riduzione, con prospezioni distinte.

5 Traduzione e semantica 20%

1. Dato lo schema di traduzione

sorgente G_1	pozzo G_2
$S \to aS$	$S \to bS$
$S \to aS$	$S \to Sc$
$S \to \varepsilon$	$S \to \varepsilon$

Nota: la traduzione permette la scelta indeterministica tra la prima e la seconda regola.

(a) Definire mediante un predicato la relazione di traduzione

$$\tau = \{(x, y) | \ldots\} \subseteq \{a\}^* \times \{b, c\}^*$$

definita dallo schema.

(b) Esaminare se traduzione definita dallo schema può essere calcolata da un trasduttore finito.

(c) Esaminare se la traduzione è ambigua.

(d) Esaminare se la traduzione è invertibile.

Soluzione:

(a)

$$\tau = \{(x, y)|x = a^n \wedge y = b^r c^s \wedge n = r + s \geq 0\}$$

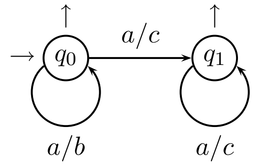
o anche

$$\tau = \{(x, y)|x = a^* \wedge y = b^* c^* \wedge |x| = |y|\}$$

(b) Poiché la grammatica di traduzione G_t definisce il ling. non regolare

$$\{(ab)^* a^n b^n \mid n \geq 0\}$$

non è possibile affermare a priori che la traduzione è calcolabile da un trasduttore finito. Osservando però la definizione (a) è facile costruire il trasduttore non deterministico:



Pertanto il teorema di Nivat permette di affermare l'esistenza di uno schema di traduzione equivalente al precedente, la cui grammatica di traduzione definisce un ling. regolare, eccolo:

sorgente G_1	pozzo G_2
$S \to aS$	$S \to bS$
$S \to \varepsilon$	$S \to \varepsilon$
$S \to aC$	$S \to cC$
$C \to aC$	$C \to cC$
$C \to \varepsilon$	$C \to \varepsilon$

(c) La traduzione è ambigua, per es.

$$\tau(a) = \{b, c\}$$

L'ambiguità è causata dal fatto che la stessa regola sorgente ($S \to aS$) è associata a due regole pozzo diverse.

(d) La traduzione è invertibile, perché la traduzione inversa τ^{-1} può essere definita tramite l'omomorfismo alfabetico

$$h(b) = a, h(c) = a$$

che produce ad es.

$$\tau^{-1}(bc) = aa$$

2. Si progetti uno schema di traduzione sintattico per trasformare una stringa di Dyck di alfabeto $\{(,)\}$ nel modo seguente. Si dice che una parentesi aperta è *pari* (risp. *dispari*) se al suo interno sta un numero pari (risp. dispari) di parentesi aperte. Ad es. nelle frasi

$$(()(()))\quad , \quad ((()(()))())\quad , \quad ((()(())))$$

sono marcate le parentesi dispari :

$$(_d\,()(_d\,()))\quad , \quad (_d\,(_d\,()(_d\,()))()),\quad ((_d\,()(_d\,())))$$

L'alfabeto pozzo contiene le parentesi quadre e uncinate. Le parentesi pari vanno tradotte in parentesi quadre e le altre in parentesi uncinate, ad es.:

$$<[\,]<[\,]>>\quad , \quad <<[\,]<[\,]>>[\,]>,\quad [<[\,]<[\,]>>]$$

(a) Si scriva uno schema di traduzione

(b) Si disegni l'albero della traduzione per il primo esempio

(c) Si verifichi se la traduzione definita dallo schema è ambigua.

Soluzione
La nota grammatica sorgente del ling. di Dyck

$$S \to (S)S|\varepsilon$$

non è evidentemente adatta a produrre una traduzione diversa per le parentesi pari e dispari. A tale fine, la grammatica deve usare regole diverse per generare le parentesi pari e dispari. Chiamiamo P (risp. D) l'insieme delle frasi tali che il numero delle parentesi aperte sia pari (risp. D), ad es.

$$P : \varepsilon, (()), ()() \;\; D : (), ((())), ()(()), ()()()$$

Lo schema richiesto è:

sorgente G_1	pozzo G_2
$S \to P$	$S \to P$
$S \to D$	$S \to D$
$P \to (P)D$	$S \to [P]D$
$P \to (D)P$	$S \to [D]P$
$P \to \varepsilon$	$S \to \varepsilon$
$D \to (D)D$	$S \to < D > D$
$D \to (P)P$	$S \to < P > P$

Esso non è ambiguo perché la grammatica sorgente non lo è.

3. Si progetti una grammatica a attributi per trasformare una stringa di Dyck di alfabeto $\{(,)\}$ nel modo seguente[2]. Si dice che una parentesi aperta è *pari* (risp. *dispari*) se al suo <u>interno</u> sta un numero pari (risp. dispari) di parentesi aperte. Ad es. nelle frasi

$$(()(())) \quad , \quad ((()(()))()) \quad , \quad ((()(())))$$

alle parentesi dispari è stata assegnata la marca d:

$$(_d\,()(_d\,())) \quad , \quad (_d\,(_d\,()(_d\,()))()) \quad , \quad ((_d\,()(_d\,())))$$

La grammatica a attributi da progettare farà le seguenti operazioni:

- Assegnerà a ogni sottoalbero ben parentesizzato l'attributo $marca \in \{pari, disp\}$
- Assegnerà all'assioma l'attributo $trad$, la traduzione della stringa, secondo la regola seguente:
  L'alfabeto pozzo contiene le parentesi quadre e uncinate. Le parentesi pari vanno tradotte in parentesi quadre e le altre in parentesi uncinate. Per gli esempi precedenti:

$$<[]<[]>> \quad , \quad <<[]<[]>>[]>, \quad [<[]<[]>>]$$

(a) Si progettino la sintassi e le funzioni semantiche.

(b) Si scriva, almeno in parte, il programma del valutatore semantico integrato con quello sintattico usando la tecnica LL o (a scelta dell'allievo) LR.

Soluzione
La nota grammatica sorgente del ling. di Dyck

$$S \to (S)S|\varepsilon$$

è il semplice supporto sintattico. La grammatica a attributi calcola due attributi sintetizzati $m = \text{marca}, t = \text{trad}$. Convenzione:

$$m = \text{true} \Leftrightarrow m = \text{pari}$$

[2] È quasi lo stesso problema dell'esercizio precedente, ma ora la soluzione richiesta usa gli attributi.

sorgente G_1	funzioni semantiche
$S_0 \rightarrow (S_1)S_2$	$m_0 \leftarrow (m_1 \text{ xor } m_2)$ $t_0 \leftarrow$ if m_1 then cat$('[', t_1, ']', t_2)$ else cat$('<', t_1, '>', t_2)$
$S \rightarrow \varepsilon$	$m_0 \leftarrow$ true $t_0 \leftarrow$ null

La grammatica ha solo attributi sintetizzati dunque è valutabile con una passata da sin. a destra. Poiché la sintassi è facilmente LL(1) si può integrare il parsificatore con il valutatore semantico.