L. Breveglieri, S. Crespi Reghizzi
Dipt. di Elettronica e Informazione, Politecnico di Milano

Linguaggi formali e compilatori: temi di esame risolti

13 maggio 2005

Casa Editrice
Milano

0.1 Prefazione

Questa raccolta non contiene le domande relative alla esercitazioni sui compilatori costruiti con *flex* e *bison*.

Indice

1

Linguaggi Formali e Compilatori Soluzioni del Compitino 09/12/003

Tempo 2 ore. Per superare la prova è necessario ottenere la sufficienza in tutte e quattro le parti: espr. regolari, automi finiti, grammatiche, esercitazioni.

	punti %	annotazioni	VOTO
1. Espressioni regolari			
2. Automi finiti			
3. Grammatiche			
4. Esercitazioni			
VOTO			

1.1 Espressioni e linguaggi regolari 25%

1. Progetto di espressioni regolari
   Il linguaggio di alfabeto $\{a, b\}$ è tale che:
   (il numero dei caratteri a è dispari) $\wedge$ (vi è almeno un b).
   Definire il linguaggio con una espressione regolare con i soli operatori di base $\cup, *, .$ e con la croce.

   Risposta

   $$A_p b A_d | A_d b A_p$$

   dove

   $$A_p = b^*(ab^*ab^*)^* \quad , \quad A_d = b^*ab^*(ab^*ab^*)^*$$

2. Analisi di espr. regolari
   Dire, spiegando le ragioni, quali delle seguenti coppie di espr. sono delle identità e quali no.

$$a^*|a^*b(a|b)^* \quad \stackrel{?}{=} \quad a^+|a^*((ba)^*)^+ \tag{1.1}$$

   Risposta No: il primo ling. può terminare con b, il secondo no.

$$a^*|a^*b(a|b)^* \quad \stackrel{?}{=} \quad a^*(b^+a^+)^*b^* \tag{1.2}$$

   Risposta Sì $(a|b)^* = (a|b)^*$

3. Trasformazioni di espr.
   Trasformare le seguenti espr. di alfabeto $\{a, b, c\}$ in altre equivalenti che usino soltanto gli operatori di base $\cup, *, .$ e $+$.

$$(ba^+)^*c^* \ \cap \ (b^*a^+c^*)^+$$

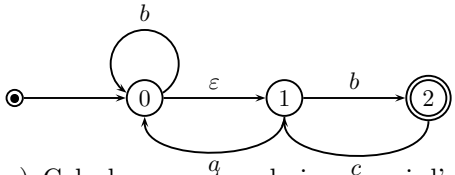
   Risposta $(ba^+)^+c^*$

$$\neg((a|b)(\neg\emptyset))$$

   Risposta $= \neg((a|b)(a|b|c)^*) = \varepsilon|c(a|b|c)^*$

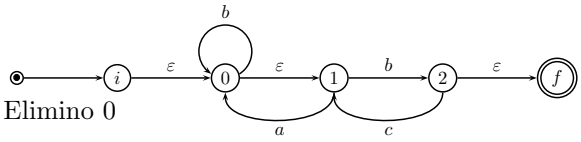
1.2 Automi finiti 25%

1. Per l'automa N dato

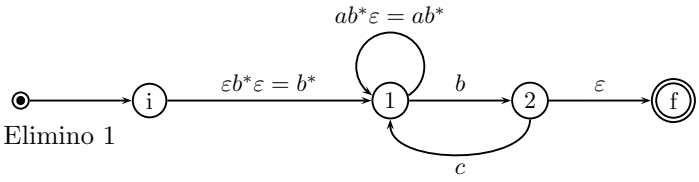


   a) Calcolare, mostrando i passaggi, l'espressione regolare del linguaggio riconosciuto da N.
   b) Costruire l'automa deterministico equivalente a N.

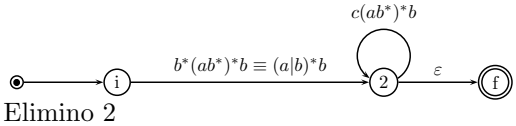
   Risposta: Metodo di eliminazione di Brzozowski e McClusky



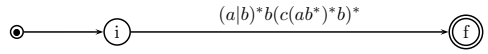
Elimino 0



Elimino 1



Elimino 2



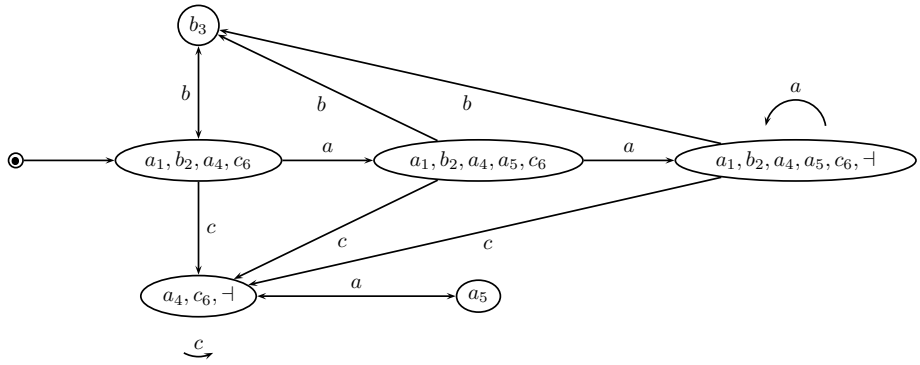
2. È data la espressione regolare

$$(a|bb)^*(aa|c)^+$$

a) Costruire con il metodo di McNaughton & Yamada l'automa deterministico del linguaggio
b) Verificare se l'automa costruito è minimo, e minimizzarlo se necessario.

Risposta
Espressione numerata

$$(a_1|b_2b_3)^*(a_4a_5|c_6)^+$$

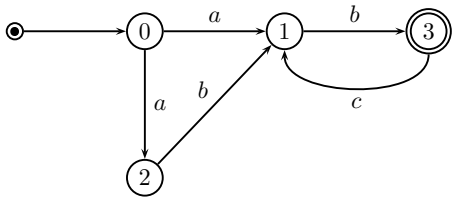


Risposta: l'automa è minimo.

3. Il *quoziente* di due stringhe f e g è la operazione, definita soltanto se f è prefisso di g, nel modo seguente
$f^{-1}g = \begin{cases} h & \text{se } g = fh \\ \text{indefinito,} & \text{altrimenti} \end{cases}$
Il quoziente di un ling. L rispetto a una stringa f è definito come

$$f^{-1}L = \bigcup_{g \in L} \{f^{-1}g\} = \{h | fh \in L\}$$

a) Per il linguaggio definito dall'automa A



- presa a come stringa, elencare 3 stringhe del quoziente $(a)^{-1}L(A)$
- costruire l'automa che riconosce il linguaggio $(a)^{-1}L(A)$

b) Il quoziente di un ling. regolare rispetto a una stringa qualsiasi risulta sempre regolare. Sapreste dimostrarlo?

risposta: b, bb, bcb

risposta: il riconoscitore del quoziente è un automa finito costruito nel modo seguente. Ogni stato r tale che $r \in \delta(q_0, f)$ viene trasformato in uno stato iniziale, mentre q_0 cessa di essere iniziale. L'automa così ottenuto deve essere ripulito se alcuni stati non sono raggiungibili. Poi, per ottenere un solo stato iniziale, si crea un nuovo nodo r_0 e lo si collega con archi epsilon ai nodi iniziali r.

1.3 Grammatiche e linguaggi liberi 25%

1. Si progetti una grammatica libera, possibilmente non ambigua, per i seguente due linguaggi. E' consentito l'uso di regole BNF estese. Si disegni per ogni caso un albero sintattico abbastanza rappresentativo.
   a) Istruzioni condizionali nello stile di Ada. L'alfabeto contiene: le parole chiave $\{if, then, else, elseif, fi\}$,
   gli operatori booleani $\{\neg, \wedge, \vee\}$ (elencati nell'ordine di precedenza che la grammatica deve imporre),
   il car. a che sta per una istruzione di assegnamento,
   il car. b che sta per una variabile booleana,
   e il punto e virgola.
   La frase è una istruzione condizionale, a uno, due o più rami, chiusa tra if e fi; il primo ramo è introdotto da $then$, i successivi rami (opzionali) da $elsif$, l'ultimo ramo (opzionale) da $else$. Ogni ramo può contenere una sequenza non vuota di istruzioni (assegnamenti o istr. condizionali) separate da punto e virgola.

```
        if      b ∨ ¬b
        then    a; a
        elseif  ¬b ∨ b ∧ b
                then        a
        elseif  b
Esempio         then        a; a; a;
                if          b ∨ ¬b
                then        a; a
                else        a
                fi
        fi
```

b) L'alfabeto è $\{a$ = aperta, c = chiusa, e = elemento$\}$. Una frase è una stringa ben parentetizzata rispetto a $\{a, c\}$, tale che vi sia un solo elemento e ed esso si trovi all'interno delle parentesi. Ad es.

$$aec,\ aacecac$$

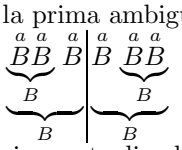
ma non

$$eac,\ aacecaec$$

risposta
$S \rightarrow aZcS \mid aScZ \mid aeZcZ \mid aZecZ \mid aec$
$Z \rightarrow aZcZ \mid ac$

2. Ambiguità e equivalenza. Mostrare un es. di ambiguità nella seguentie grammatica e costruire una grammatica equivalente non ambigua.
$S \to BC \mid BCC$
$B \to BB|a$
$C \to b|S$

risposta
la prima ambiguità



si cura togliendo la doppia ricorsione:
$B \to aB|a$

La seconda ambiguità è la stessa delle istruzioni condizionali

`if ... then ... [else ... ]`
come si vede dalla corrispondenza

$$B \Leftrightarrow \text{condizione},\ C \Leftrightarrow \text{parte then},\ C \Leftrightarrow \text{parte else}$$

La soluzione è a pag. 221 del libro di testo.

3. Per il linguaggio

$$L = \{(a^{2n}cb^{2n} \mid n \geq 1\} \cup \{(a^{2n+1}b^{2n+1} \mid n \geq 0\}$$

progettare un automa riconoscitore a pila, preferibilmente deterministico, disegnare il suo grafo, e eseguire la traccia del funzionamento sulla stringa $aabb \dashv$

risposta: metto in pila le a sotto forma di A registrando in uno stato la parità. Se nello stato pari incontro una c cambio stato e poi disimpilo una A a ogni lettura di b. Se nello stato dispari incontro una b cambio stato e disimpilo una A a ogni successiva lettura di b.

2

Linguaggi Formali e Compilatori: soluzioni del Compitino 16/12/004

Tempo 2h. 30. Per la sufficienza è necessario dimostrare di conoscere tutte e tre le parti.

2.1 Espressioni e linguaggi regolari 25%

1. Progetto di espressioni regolari
   Il linguaggio di alfabeto $\{a, b, c\}$ è tale che:
   (il numero dei caratteri a è dispari) $\wedge$ (tra due b non può trovarsi nessuna c).
   Definire il linguaggio con una espressione regolare con i soli operatori di base $\cup, *, .$ e con la croce.

   Soluzione

   Il ling. è ottenuto mischiando i due ling. $a(aa)^*$ e $c^*b^*c^*$, corrispondenti alle due condizioni dell'enunciato. Il ling. si può scrivere come

   $$\begin{array}{lll} (c^*ac^*ac^*)^* & (b^*ab^*ab^*)^* & (c^*ac^*ac^*)^*ac^* \mid \\ (c^*ac^*ac^*)^* & (b^*ab^*ab^*)^*ab^* & (c^*ac^*ac^*)^* \mid \\ (c^*ac^*ac^*)^*ac^* & (b^*ab^*ab^*)^* & (c^*ac^*ac^*)^* \mid \\ (c^*ac^*ac^*)^*ac^* & (b^*ab^*ab^*)^*bc^* & (c^*ac^*ac^*)^*ac^* \end{array}$$

   Per illustrare, la prima riga produce frasi della forma $R_1R_2R_3$ in cui R_1 e R_2 contengono un numero pari di a e R_3 un numero dispari. Al contempo la sottoparola, formata dalle lettere b e c presenti nella stringa, ha la forma $c^*b^*c^*$.

2. Analisi di espr. regolari
Dire, spiegando le ragioni, se le seguenti sono delle identità:

$$(\neg a)b \quad \stackrel{?}{=} \quad b \mid b(a \mid b)^* b \tag{2.1}$$

$$((a \mid b \mid \varepsilon)^2)^+ \quad \stackrel{?}{=} \quad (a \mid b)^* \tag{2.2}$$

Soluzione

- (1) No:
$$(\neg a)b = (\Sigma^* b \setminus \{ab\}) \ni aab \notin (b \mid b(a \mid b)^* b)$$
- (2) Sì:
$$(a \mid b \mid \varepsilon)^2 = \{x \mid |x| \leq 2\} = \varepsilon \mid (a \mid b) \mid (a \mid b)^2$$
Inoltre essendo
$$(a \mid b)^+ \supset ((a \mid b)^2)^+$$
risulta
$$(\varepsilon \mid (a \mid b) \mid (a \mid b)^2)^+ = (\varepsilon \mid a \mid b)^+ = (a \mid b)^*$$

Soluzione

3. Verificare, motivando la risposta, se la seguente espr. reg. è ambigua:

$$(aa \mid ba)^* a \mid b(aa \mid ba)^*$$

Soluzione

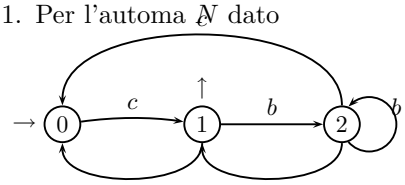
L'espr. numerata

$$(a_1 a_2 \mid b_3 a_4)^* a_5 \mid b_6 (a_7 a_8 \mid b_9 a_{10})^*$$

definisce le frasi $b_3 a_4 a_5$ e $b_6 a_7 a_8$ che si proiettano in modo ambiguo nella stessa frase baa.
Ossia esistono due diverse implicazioni sinistre producenti la stessa stringa.

2.2 Automi finiti 25%

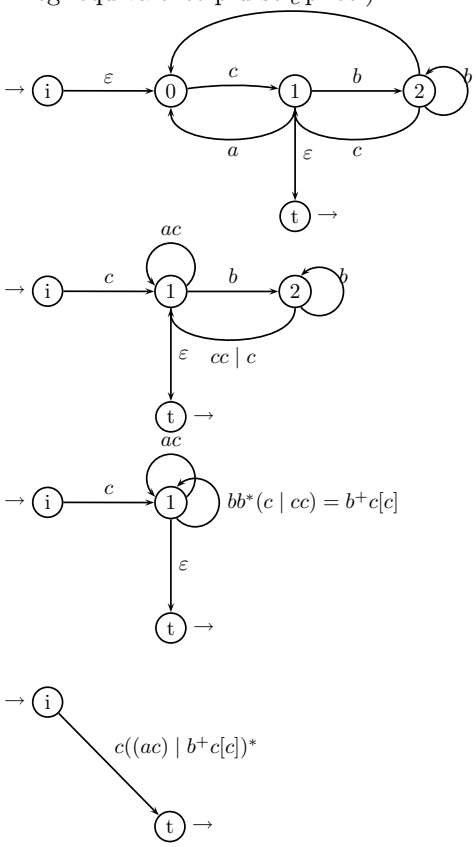
1. Per l'automa N dato



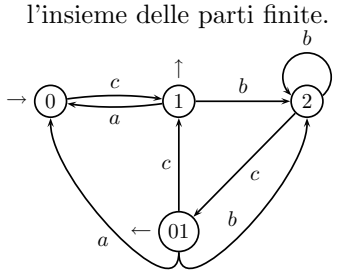
   a) Calcolare, mostrando i passaggi, l'espressione regolare del linguaggio riconosciuto da N.
   b) Costruire l'automa deterministico equivalente a N.

   Soluzione

   a) Normalizziamo l'automa, per poi applicare il metodo di eliminazione di Brzozowsky e Mc Cluskey. (Un altro metodo sarebbe quello di scrivere la gramm. lineare a destra e risolvere il sistema di eq. insiemistiche.). Nell'applicare il metodo scegliamo di eliminare i nodi del grafo nell'ordine 0,1,2. (In effetti l'ordine 2,0,1 produrrebbe una espr. reg. equivalente più semplice.)



b) Automa deterministico :
non essendovi archi epsilon, si applica direttamente la costruzione dell'insieme delle parti finite.



2. È data la espressione regolare

$$((a \mid \varepsilon)b^+ \mid a^*b)^+$$

a) Costruire, mostrando i passaggi, l'automa deterministico del linguaggio
b) Verificare se l'automa costruito è minimo, e minimizzarlo se necessario.

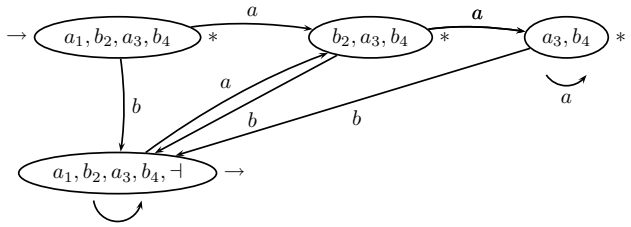
Soluzione

a) Automa deterministico del linguaggio. Applichiamo il metodo di McNaughton-Yamada, costruendo direttamente il riconoscitore deterministico con l'algoritmo di Berry-Sethi.
Espressione numerata:

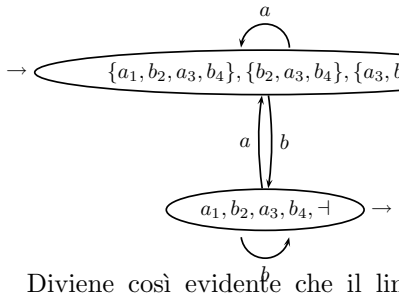
$$((a_1 \mid \varepsilon)b_2^+ \mid a_3^*b_4)^+ \dashv$$

Inizi: a_1, b_2, a_3, b_4
Fini: b_2, b_4
Seguiti:

	Séguiti
a_1	b_2
b_2	$a_1, b_2, a_3, b_4, \dashv$
a_3	a_3, b_4
b_4	$a_1, b_2, a_3, b_4, \dashv$



b) Automa costruito minimo.
Gli stati asteriscati sono indistinguibili e vanno fusi insieme.



Diviene così evidente che il ling. comprende tutte le stringhe che terminano per b, ossia è definito dalla espr. reg.

$$(a^*b)^+$$

Come verifica finale osserviamo che la espr. reg. data nell'enunciato equivale a quest'ultima.

3. Costruire, spiegando il ragionamento, un riconoscitore, non importa se indeterministico, con il minor numero di stati possibile, per il linguaggio di alfabeto $\{a, b\}$ così definito:
   - la penultima lettera è b
     $\wedge$
   - la seconda lettera è b.

   Soluzione

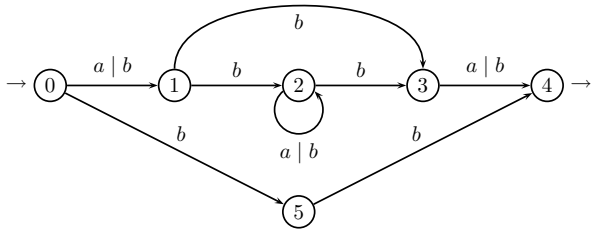
   - Poiché il ling. è formulato come congiunzione di due condizioni, esso è l'intersezione dei due corrispondenti ling.

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

     e un riconoscitore può essere costruito come il prodotto cartesiano dei due riconoscitori, ognuno con 3 stati. La macchina prodotto ha 9 stati di cui 3 irraggiungibili.
   - Il riconoscitore si può costruire direttamente osservando che il ling. è l'unione di tre casi:

     $$L = (a \mid b) b (a \mid b) \quad | \quad (a \mid b) b (a \mid b)^* b (a \mid b) \quad | \quad bb$$

     da cui si ricava l'automa



2.3 Grammatiche e linguaggi liberi 50%

1. Data la espr. reg.

$$(a(bc)^+b^* \mid (bc)^*a)^*$$

costruire una gramm. libera, non EBNF, del linguaggio.

Soluzione

Fatta l'analisi sintattica della espr. reg. R nelle sue sottoespressioni,

$$\underbrace{\underbrace{(a\,\underbrace{(bc)^+}_{D}\,\underbrace{b^*}_{E}}_{B}\mid\underbrace{(bc)^*}_{F}\,a)^*}_{C}}_{A}$$

si scrivono le regole:

$$R \to AR \mid \varepsilon \qquad A \to B \mid C \qquad B \to aDE \qquad C \to Fa$$

$$D \to bcD \mid bc \qquad E \to bE \mid \varepsilon \qquad F \to bcF \mid \varepsilon$$

Osservando che $(bc)^+ = bc(bc)^*$, si semplifica la grammatica:

$$R \to AR \mid \varepsilon \qquad A \to B \mid C \qquad B \to abcFE \qquad C \to Fa$$

$$E \to bE \mid \varepsilon \qquad F \to bcF \mid \varepsilon$$

2. Progettare una grammatica libera, BNF o EBNF, per il linguaggio (meta-linguaggio) che rappresenta l'insieme di regole delle grammatiche EBNF, con le precisazioni seguenti:
   - L'alfabeto terminale delle grammatiche è a, b
   - Gli operatori sono:
     `\ast` per la stella
     `\cup` per l'unione
     concatenamento (non si scrive il puntino)
   - la freccia è scritta `\rightarrow`
   - la precedenza tra gli operatori è quella solita: stella, concatenamento, e infine unione
   - Le regole di una gramm. sono separate da `\\`
   - I simboli nonterminali sono del tipo `S301`, cioè iniziano per `S` seguito da un numero.
   - Esempi di regole:

     ```
     S301 \rightarrow (a S301 b)\ast \cup ab \\
     S3 \rightarrow (a S9 b\cup  a\ast )\ast
     ```
   - Per ogni aspetto non definito è lasciata libertà di scelta.
   - La grammatica non deve essere ambigua.

   Disegnare un albero sintattico abbastanza rappresentativo.

   Soluzione

   Lista non vuota di regole R:

   $$S \to R(\backslash\backslash R)^* \qquad R \to N\texttt{\textbackslash rightarrow}D$$

   La parte sinistra è un simbolo nonterminale:

   $$N \to' S'(1\ldots9)(0\ldots9)^* \mid \ 'S'0$$

   La parte destra delle regole è una lista di espressioni E separate dal segno di unione:

   $$D \to E(\texttt{\textbackslash cup}E)^*$$

   Una espr. E è il concatenamento di fattori F eventualmente seguiti dal simbolo della stella:

   $$E \to (F[\texttt{\textbackslash ast}])^+$$

   ognuno dei quali è uno dei seguenti casi:

   $$F \to N \mid a \mid b \mid \ '(' D ')'$$

   Sebbene l'enunciato non consideri la presenza della stringa vuota nelle parti destre, essa si può ammettere, aggiungendo la regola:

   $$F \to \texttt{\textbackslash epsilon}$$

   (non si può usare il carattere ε perché fa parte del metaalfabeto).

3. Progettare una grammatica, BNF o EBNF, per il seguente linguaggio di alfabeto $\{a, b, c\}$. Una frase x è così definita:
   - $x = x_1 c^+ x_2 c^+ \dots c^+ x_n$ dove $n \geq 2$,
     ogni sottostringa x_i ha la forma $a^+ b^+$
     ma le stringhe di posto dispari e pari sono definite diversamente:

$$x_{2j+1} = a^h b^k, h \geq k \geq 1 \qquad x_{2j} = a^k b^h, h \geq k \geq 1$$

   - Esempi: *aabcabb*, *aaabcccabcab*
     Controesempi: *abbcab*, *aababbcab*

   Soluzione

   Lista di elementi Dc^+P, separati da c^+, eventualmente chiusa dal termine c^+D:

$$S \to \underbrace{Dc^+P}_{elem.}(c^+ \underbrace{Dc^+P}_{elem.})^*[c^+D]$$

$$D \to a^*E \qquad P \to Eb^* \qquad E \to aEb \mid ab$$

4. Per la grammatica G:

$$S \to ScSdS \mid bS \mid B \qquad B \to bB \mid \varepsilon$$

Mostrare un es. di ambiguità e costruire una grammatica (non EBNF) equivalente e non ambigua.

Soluzione

Due sono le cause di ambiguità presenti:
Ricorsione bilaterale: La prima regola, essendo ricorsiva a sin. e destra, causa l'ambiguità di associazione.
Due modi di produrre le b: Ad es. la frase b è ambigua:

$$S \Rightarrow bS \Rightarrow bB \Rightarrow b \qquad S \Rightarrow B \Rightarrow bB \Rightarrow b$$

Il ling. generato è una variante di quello di Dyck di alfabeto $\{c, d\}$, definito dalla gram. non ambigua $S \to cSdS \mid \varepsilon$. La variante è ottenuta inserendo le b in qualsiasi posizione. Una grammatica equivalente non ambigua è:

$$S \to BcSdS \mid B \qquad B \to bB \mid \varepsilon$$

o anche

$$S \to cSdS \mid bS \mid \varepsilon$$

5. Dati l'alfabeto $\Sigma = \{'\{', '\}', '[', ']', '(', ')'\}$ e la grammatica G

$$S \to \{S\}S \mid [S]S \mid (S)S \mid \varepsilon$$

a) Scrivere la grammatica (non EBNF) del ling. in cui nessuna coppia di graffe può contenere direttamente una coppia di tonde. Vedere gli esempi proibiti: $\{(\ldots)\}$, $\{\{\ldots\}(\ldots)[\ldots]\}$
e gli esempi ammessi $\{[(\ldots)]\}$, $\{[\{\ldots\}(\ldots)][\ldots]\}$

b) (Facoltativo) Dato il ling. regolare

$$R = \neg(\Sigma^* \,'\{'\,'('\,\Sigma^*)$$

scrivere la grammatica (non EBNF) del ling. $L(G) \cap R$

Soluzione

a) Differenziamo il simbolo nonterminale all'interno delle graffe, in modo da evitare la scelta delle parentesi tonde:

$$S \to \{X\}S \mid [S]S \mid (S)S \mid \varepsilon$$

$$X \to \{X\}X \mid [S]X \mid \varepsilon$$

b) Questo linguaggio include strettamente il precedente, ad es. esso contiene il secondo es. proibito. La grammatica è dunque più permissiva, e consente le parentesi tonde in posizioni prima proibite:

$$S \to \{X\}S \mid [S]S \mid (S)S \mid \varepsilon$$

$$X \to \{X\}S \mid [S]S \mid \varepsilon$$

6. (Facoltativo) Dato il linguaggio

$$L = \{a^m b^n c^m \mid m \geq n \geq 0\}$$

a) Scrivere la grammatica contestuale (tipo 1) di L
b) Giustificare che L non è un ling. libero
c) Il complemento $\neg L$ è libero?

Soluzione

a) Una grammatica contestuale è

$$S \to aSBC \mid \varepsilon \qquad CB \to BC \mid C$$

La prima regola genera ricorsivamente le forme di frase

$$a^m (BC)^m$$

La regola $CB \to BC$ raccoglie tutte le B prima di tutte le C, producendo le stringhe:

$$a^m B^m C^m$$

oppure, se si sceglie l'alternativa $CB \to C$, le stringhe:

$$a^m B^n C^m, \qquad m \geq n$$

Le seguenti regole sostituiscono i caratteri terminali al posto dei simboli nonterminali.

$$bB \to bb \qquad aB \to ab \qquad C \to c$$

b) Data una stringa sorgente $a^r b^s c^t$, un automa a pila, dopo il necessario controllo che sia $r \geq s$, non può tenere memoria dei valori di r e s, ma soltanto della loro differenza. Dunque esso non può controllare l'eguaglianza di r e di t.
c) Il complemento $\neg L$ è essenzialmente l'unione di due sottolinguaggi:
- Il ling. delle stringhe che non hanno la forma $a^* b^* c^*$, ossia il ling. $\neg(a^* b^* c^*)$ che è regolare.
- Il ling. delle stringhe $a^r b^s c^t$ che violano la condizione

$$r \geq s \wedge r = t$$

Ossia le stringhe soddisfano una delle tre condizioni

$$r \neq s \vee r < s \vee t < s$$

Esso è dunque l'unione di tre ling.:

$$\{a^r b^s c^t \mid r \neq s\} \cup \{a^r b^s c^t \mid r < s\} \cup \{a^r b^s c^t \mid t < s\}$$

ognuno dei quali è facilmente libero. In conclusione il complemento $\neg L$ è l'unione di un ling. regolare e di tre linguaggi liberi ossia è un ling. libero.

3

Linguaggi Formali e Compilatori: soluzioni della prova scritta 06/10/2004

Tempo 2 ore 30' Revisione 25.01.2005 Per superare la prova l'allievo deve dimostrare la conoscenza di tutte e 5 le parti.

3.1 Espressioni regolari e automi finiti 20%

1. Progetto di espr. regolare. Alfabeto $\Sigma = \{a, b\}$. Con riferimento ai linguaggi
   $R_1 = \{x \in \Sigma^* \mid x \text{ contiene due sottostringhe } ab\}$
   $R_2 = \{x \in \Sigma^* \mid x \text{ contiene un numero(anche nullo) pari di } a\}$
   $R = R_1 \setminus R_2$
   a) Scrivete le espr. reg. di R_1, R_2, R con i soli operatori di base $\{\cup, *, .\}$
   b) Scrivete l'espr. reg. di R usando anche l'operatore $\neg$ e $\cap$ (ma non la differenza insiemistica).

   Soluzione

   a)
   $$R_1 = b^*a^*abb^*a^*abb^*a^*$$
   $$R_2 = (b^*ab^*a)^*b^* \text{ opp.}(b^*(ab^*a)^*)^*$$
   Una frase di R è una frase di R_1 con il vincolo di avere un numero dispari di a.
   Da cui:
   $$R = b^*\{a^rabb^*a^sabb^*a^t \mid r + s + t \text{ dispari}\}$$
   Si danno quattro casi: r, s pari, t dispari; r, t pari, s dispari; s, t pari, r dispari; r, s, t dispari.
   $$R = b^*((a^2)^*abb^*(a^2)^*abb^*(a^2)^*a \mid \ldots \mid \ldots \mid \ldots)$$
   b) $R = R_1 \cap \neg R_2$

2. Dato il linguaggio regolare

$$L = (\neg(b^+) \cap (aa \mid bb)^*)^+$$

a) Costruite, mostrando i passaggi, il riconoscitore deterministico minimo del linguaggio.
b) Calcolate l'espr. reg. senza gli operatori $\neg, \cap$

Soluzione

Vediamo due approcci per costruire il riconoscitore di L:

Composizione di automi

a) Costruire il riconoscitore minimo A_1 di $\neg(b^+)$
b) Costruire il riconoscitore minimo A_2 di $(aa \mid bb)^*$
c) Costruire la macchina $A = A_1 \times A_2$ prodotto che riconosce $R = \neg(b^+) \cap (aa \mid bb)^*$
d) Osservare che $L = R^+ \equiv R$ per la idempotenza di stella e croce.
e) Calcolare la e.r. di $L(A)$ con il metodo di eleiminazione opp. risolvendo le eq.

Osservazione del linguaggio

Si vede facilmente che l'intersezione R contiene la stringa vuota e le stringhe contenenti almeno una a tra quelle appartenenti a $(aa \mid bb)^*$, ossia

$$R = \varepsilon \mid (a^2 \mid b^2)^* a^2 (a^2 \mid b^2)^*$$

da cui è facile ricavare, a occhio (o con il metodo di McNaughton e Yamada) l'automa.

3.2 Grammatiche 20%

1. Progettate una grammatica (consentita la forma EBNF) per il ling. di alfabeto $\{(,), v, +, \times\}$ così definito, con riferimento al linguaggio delle espressioni aritmetiche.
   - Il prodotto ha precedenza sulla somma
   - Gli operatori sono associativi a sinistra
   - Una espressione moltiplicativa con tre o più operandi $o_1 \times o_2 \times o_3 \times \ldots$ deve essere racchiusa tra parentesi. (Le parentesi sono possibili anche con uno o due operandi.)
     Ad es. $v \times v \times (v \times v)$ non è valida, ma è valida $(v \times v \times (v \times v))$

   a) Progettate la grammatica
   b) Disegnate l'albero sint. della frase $v + (v \times (v + v) \times v) \times (v \times v)$

   Soluzione

   $E \to T(+T)^*$
   $T \to F \mid F \times F \mid \ '('F \times F(\times F)^+\ ')'$
   $F \to v \mid \ '('E')'$

2. Il ling. L descrive la serie di operazioni nel carrello di un negozio virtuale.

Alfabeto delle azioni	Le regole di validità
i inserisce un articolo nel carrello	sempre possibile
t toglie un articolo dal carrello	t è lecito solo se il carrello non è vuoto
a azzera il prezzo degli articoli presi	a è possibile e obbligatorio immediatamente dopo l'azione t, se essa ha svuotato il carrello

Esempi validi	Esempi non validi
$i, iit, iitta, itaiittai$	$t, ti, iita, ia, ait$

a) Descrivete il ling. a parole o con un predicato caratteristico.
b) Scrivete la grammatica (consentita la forma EBNF)
c) Disegnate un albero sintattico sufficientemente rappresentativo.

Soluzione

Astraendo dall'alfabeto si nota che i e t possono essere visti come una marca di apertura e di chiusura di un ling. a parentesi.
Sia D il ling. di Dyck di alfabeto i, t
sia N il ling. $\{i^n t^n \mid n \geq 1\}$
e sia P il ling. dei prefissi del ling. di Dyck D.

Il ling. L ha la struttura seguente

$$\underbrace{x_1 a x_2 a \dots x_n a}_{x_i \in N}\ i\ \underbrace{y}_{y \in P}$$

La grammatica è:
$S \to (Na)^+ iP \mid (Na)^+ \mid iP$
$N \to iNt \mid it$
$P \to i^+ DtD \mid i^*$

Domanda relativa alle esercitazioni 20%

Vedi fogli separati

3.3 Grammatiche e analisi sintattica 20%

Dato il linguaggio

$$L = \{a^n b^n \mid n \geq 0\} \cup \{a^{2n} c b^{2n} \mid n \geq 0\}$$

1. Progettate una grammatica adatta all'analisi LL oppure LR.
2. Verificate che la grammatica sia, in conformità con la vostra scelta, LL(k) o LR(k).

3.3.1 Soluzione

LL(1): A prima vista il linguaggio potrebbe sembrare non essere LL(k), per ogni k, perché i due sottolinguaggi iniziano allo stesso modo con un numero qualsiasi di a. Ma la solita astuzia di mettere in comune le derivazioni dei due sottolinguaggi risolve facilmente il problema. La grammatica LL(1) è:

$$\begin{array}{l} S \to aAb \\ S \to \varepsilon \\ \underline{S \to c} \\ A \to aBb \\ A \to \varepsilon \\ \underline{A \to c} \\ B \to aAb \\ A \to \varepsilon \end{array}$$

LR(1): La grammatica LR(1) è:

$$\begin{array}{l} S \to X \mid Y \\ X \to aXb \mid \varepsilon \\ Y \to a^2 Y b^2 \mid c \end{array}$$

3.4 Traduzione e semantica 20%

1. Le compagnie aeree che volano tra le città indicate sono

Berlin	Milano	AZ, LH
Berlin	Paris	AF, LH
Milano	Paris	AF, AZ
Milano	Roma	AZ

Il ling. sorgente di alfabeto $\{B, M, P, R\}$ è l'insieme dei cammini, anche ciclici, da Milano a Milano, es.

$$x = \ MBPBMRM$$

La sua traduzione di alfabeto pozzo $\{AF, AZ, LH\}$ contiene le sequenze delle compagnie aeree possibili per quell'itinerario, ad es.

$$\tau(x) = \{AZ\ AF\ AF\ AZ\ AZ\ AZ,\ LH\ AF\ AF\ AZ\ AZ\ AZ,\ \ldots\}$$

a) Progettate un automa trasduttore per calcolare la traduzione.

b) Progettate una gramm. a attributi per calcolare come attributo una stringa di alfabeto $\{AF, AZ, LH\}$ appartenente a $\tau(x)$ tale da approssimare il seguente criterio: la varianza del numero di voli con ogni compagnia deve essere minima. Ad es. per l'itinerario precedente: $LH\ AF\ AF\ LH\ AZ\ AZ$

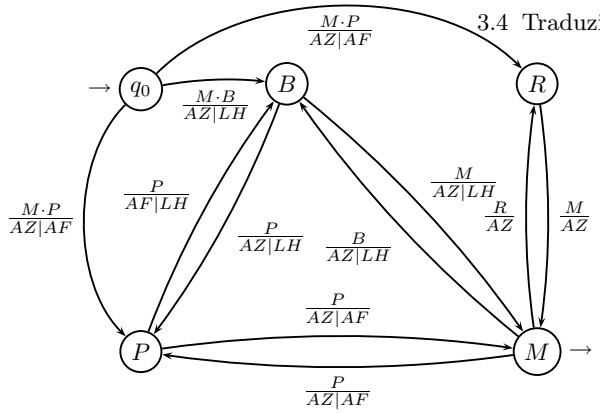
c) Disegnate un albero decorato con gli attributi

d) Disegnate i grafi delle dipendenze funzionali e indicate quali algoritmi di valutazione si possono applicare.

Soluzione

a) Automa trasduttore non deterministico

Le frasi più corte sono $M(B \mid P \mid R)M$.



b) Gramm. a attributi

Sintassi: $\Sigma = \{b, m, p, r\}$

$S \to mM$

$M \to bB$

$M \to pP$

$M \to rR$

$B \to mM$

$B \to pP$

$B \to m$

$P \to \ldots$

$\ldots$

Attributi e funzioni semantiche:

c	ereditato	è un record con i campi h, f, z, i contatori delle 3 compagnie
v	ereditato	sequenza delle compagnie scelte
$iter$	sintetizzato	lla sequenza finalke delle compagnie scelte
$min(a_1, a_2)$		quella tra le compagnie a_1, a_2 che ha viaggiato meno
$aggiorna(c, a)$		aggiorna i totali c aggiungendo la compagnia a

$S \to mM$	$c_1 := (0,0,0)\ v_1 := \varepsilon$
$M \to bB$	$c_1 := aggiorna(c_0, min(c_0.h, c_0.z))\ v_1 := cat(v_1, min(c_0.h, c_0.z)$
$M \to pP$	$\ldots$
$M \to rR$	$c_1 := aggiorna(c_0, 'AZ')\ v_1 := cat(v_1, 'AZ')$
$B \to mM$	$\ldots$
$B \to pP$	$\ldots$
$B \to m$	$iter_0 := cat(v_0, min(c_0.h, c_0.z))$
$P \to \ldots$	$\ldots$

$\ldots$

Valutazione: La grammatica è valutabile con una scansione del tipo L

4

Linguaggi Formali e Compilatori: soluzioni della Prova scritta 08/09/2004

Tempo 2 ore 30'. Libri e appunti personali possono essere consultati. Per superare la prova, l'allievo deve dimostrare la conoscenza di tutte e 5 le parti

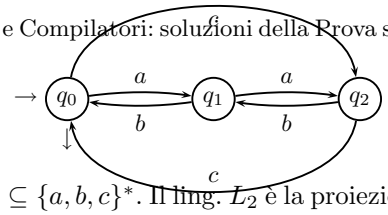
4.1 Espressioni regolari e automi finiti 20%

1. Progetto di espr. regolare. Alfabeto terminale $\{a, b\}$. Una frase del linguaggio R è una stringa non vuota, contenente un numero pari di comparse della sottostringa ab.
   a) Scrivere 3 frasi di lunghezza 6.
   b) Scrivere l'e.r. di R con i soli operatori unione, concatenamento, stella, croce.
   c) Verificare se vale la relazione $R = R^*$.

Soluzione

a) $ba^5, abb^2ab, ababba$
b) $R = b^+a^* \mid a^+ \mid b^*a^*(ab\,b^*a^*ab\,b^*a^*)^+$
c) $R \neq R^*$: ad es. la stringa $ba.ba \in R^2$ contiene un numero dispari di ab. Più semplicemente, R non deve contenere la stringa ε, ma R^* la contiene.

2. E' dato l'automa A_1

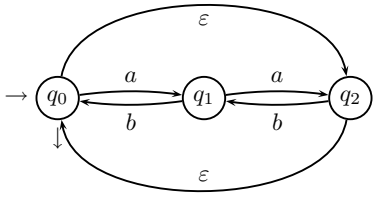


che riconosce il ling. $L_1 \subseteq \{a, b, c\}^*$. Il ling. L_2 è la proiezione di L_1 definita da $L_2 = \pi(L_1)$ dove $\pi(a) = a, \pi(b) = b, \pi(c) = \varepsilon$. Mostrando i passaggi

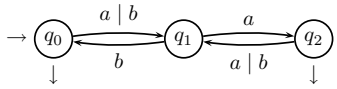
a) costruire il riconoscitore deterministico minimo di L_2
b) calcolare l'espr. reg. di L_2

Soluzione

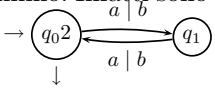
a) Automa non deterministico di L_2:



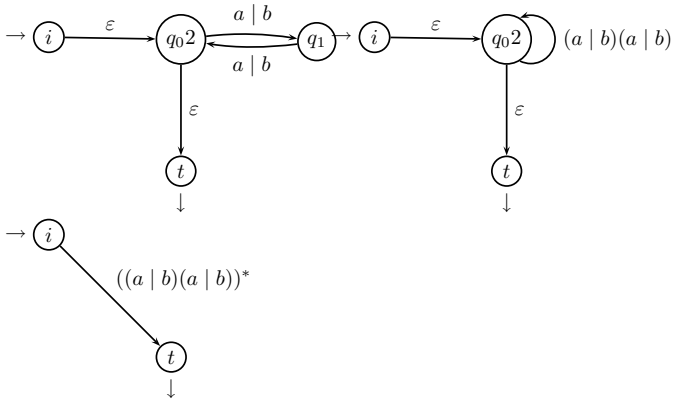
Eliminando le mosse spontanee:



L'automa è deterministico ma non minimo. Infatti sono equivalenti gli stati q_0 e q_2. L'automa minimo è:



b) Si può applicare il metodo di riduzione di B&McC:



4.2 Grammatiche 20%

1. Una frase del ling. è una espressione aritmetica contenente somme e moltiplicazioni, scritta con le seguenti convenzioni
   - la somma è denotata dall'operatore infisso '+'.
     La moltiplicazione è invece denotata dall'operatore prefisso *mult* e ha precedenza sulla somma.
   - L'operatore di somma è associativo a sinistra.
   - Gli operandi sono numeri di una sola cifra $1 \ldots 9$
   - Possono esserci le parentesi $'($ e $')'$.

   Esempi: $2 + 5 +\ mult\ 4\ mult\ 6\ 7$, $\quad mult(3 + 2 + mult\ 4\ 5)8$
   a) Progettare la grammatica in forma BNF (non EBNF).
   b) Disegnare l'albero sintattico di uno degli esempi.

Soluzione

$E \to E + T \mid T$ nota: rispetta l'associazione a sinistra;
$T \to mult\ TT \mid F$ nota: ogni termine T può iniziare con *mult* ;
$F \to' (' E ')' \mid 1 \mid \ldots \mid 9$

2. E' data la grammatica G:

$$1\ S \to XY$$
$$2\ X \to aXb\ 3\ X \to aX\ 4\ X \to ab$$
$$5\ Y \to bYa\ \ 6\ Y \to bY\ \ \ 7\ Y \to ba$$

a) Mostrare che G è ambigua.
b) Costruire una grammatica equivalente non ambigua.
c) Disegnare l'automa che riconosce $L(G)$.

Soluzione

a) Sono tre le cause di ambiguità:
   - Ordine indifferente di uso delle regole 2 e 3
   - Ordine indifferente di uso delle regole 5 e 6
   - Ambiguità di concatenamento:

$$\overbrace{aab}^{X}\overbrace{bba}^{Y} \qquad \overbrace{aabb}^{X}\overbrace{ba}^{Y}$$

b) Il ling. $L(G) = \{a^m b^n b^q a^r \mid m \geq n \geq 1 \wedge q \geq r \geq 1\}$ è anche definito dal predicato caratteristico
$L(G) = \{a^m b^s a^r \mid m \geq 1 \wedge s > r \geq 1\} = \{a^+ b^+ b^r a^r \mid r \geq 1\}$
e dunque dalla gramm. non ambigua:

$$S \to a^+ b^+ Z \qquad\qquad Z \to bZa \mid ba$$

c) Automa a pila. Intuitivamente, esso cambia stato alla prima a letta, poi legge qualsiasi numero di a. Leggendo la prima b cambia stato e impila le b lette. Leggendo poi a cambia stato e per ogni a letta, cancella dalla pila un simbolo. Accetta se la pila non si è vuotata prima del termine della stringa.
Diagramma stato-transizione: omesso.

4.3 Domanda relativa alle esercitazioni

Vedi fogli separati.

4.4 Grammatiche e analisi sintattica 20%

1. E' data la grammatica: $S \to aSbS$ $S \to aS$ $S \to \varepsilon$
   a) Calcolare gli insiemi guida e verificare se la grammatica è LL(1).
   b) Costruire il riconoscitore dei prefissi ascendenti e verificare se la grammatica è LR(1).

Soluzione

a) La grammatica non è LL(1):

	Guida
$S \to aSbS$	a
$S \to aS$	a
$S \to ab$	$b, \dashv$

b) Poiché la gramm. presenta ambiguità nell'ordine di applicazione delle regole

$$S \Rightarrow aSbS \Rightarrow aaSbs \text{ oppure } S \Rightarrow aS \Rightarrow aaSbS$$

a priori sappiamo che non può essere LR(k). Costruendo il riconoscitore dei prefissi si trova lo stato

$S \to aS^{\bullet}bS$	$b, \dashv$
$S \to aS^{\bullet}$	$b, \dashv$

inadeguato perché da esso esce la mossa etichettata b, carattere appartenente alla prospezione della riduzione $S \to aS^{\bullet}$

4.5 Traduzione e semantica 20%

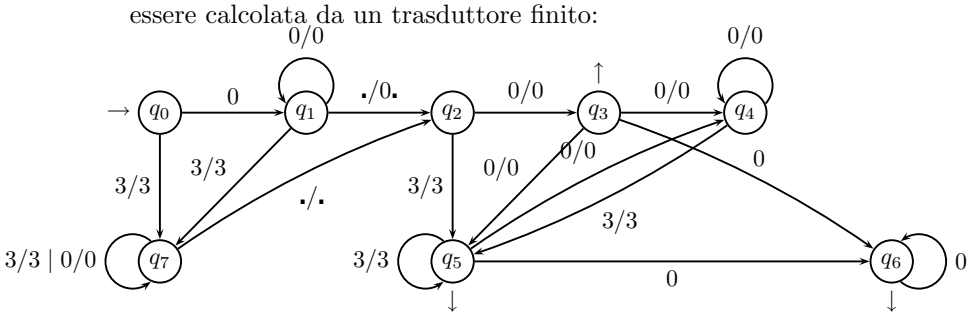
1. E' richiesto di eliminare gli zeri non significativi da un numero decimale quale 00203.0300 ottenendo così il numero 203.03. Si precisa che il punto decimale non può mancare. Altro es.: 000.00 diventa 0.0
   a) Definire la trasformazione mediante uno schema di traduzione (senza attributi semantici) e disegnare per il primo esempio gli alberi sorgente e pozzo.
   b) Progettare un trasduttore per calcolare la traduzione e verificare se esso è deterministico.

Soluzione

a) Schema di traduzione:

Sorgente	Pozzo	Gramm. traduzione
$S \to ZP\,.\,F$	$S \to ZP\,.\,F$	$S \to ZP\{.\}F$
$Z \to 0Z$	$Z \to Z$	$Z \to 0Z$
$Z \to \varepsilon$	$Z \to \varepsilon$	$Z \to \varepsilon$
$P \to (1\|\ldots 9)Q$	$P \to (1\|\ldots 9)Q$	$P \to (1\|\ldots 9)\{(1\|\ldots 9)\}Q$
$P \to 0$	$P \to 0$	$P \to 0\{0\}$
$Q \to (0\|\ldots 9)Q$	$Q \to (0\|\ldots 9)Q$	$Q \to (0\|\ldots 9)\{(0\|\ldots 9)\}Q$
$Q \to \varepsilon$	$Q \to \varepsilon$	$Q \to \varepsilon$
$F \to 0Z$	$F \to 0Z$	$F \to 0\{0\}Z$
$F \to Q(1\|\ldots 9)Z$	$F \to Q(1\|\ldots 9)Z$	$F \to Q(1\|\ldots 9)\{(1\|\ldots 9)\}Z$

b) Poiché la gramm. di traduzione non è autoinclusiva, la traduzione può essere calcolata da un trasduttore finito:



Nota: 3 sta per una cifra non nulla.
L'automa non è deterministico nel calcolo che segue il punto decimale. Per ottenere un automa det. si deve passare a un modello a pila. Esso emetterà il primo 0 della parte frazionaria. Trovando poi degli zeri li impilerà fino a trovare il terminatore, o una cifra positiva. Nel primo

caso svuoterà la pila; nel secondo caso emetterà gli zeri via via tolti dalla pila. Poi emetterà la cifra positiva, ecc.

2. Gramm. a attributi. Un insieme di insiemi, come $\{e_1, e_5\}\{e_5, e_2\}\{e_6, e_1, e_5\}$, è una frase del ling. definito dalla sintassi
   1 $S \to \{I\}S$
   2 $S \to \{I\}$
   3 $I \to e, I$
   4 $I \to e$
   Gli elementi e_i sono simboli terminali aventi un attributo lessicale che li identifica.
   a) Progettare una grammatica a attributi per calcolare:
      - l'intersezione (nell'es. $\{e_5\}$) degli insiemi;
      - la differenza insiemistica di ogni insieme rispetto alla intersezione calcolata (nell'es. $\{e_1\}, \{e_2\}\{e_6, e_1\}$).
   b) Disegnare i grafi delle dipendenze funzionali e stabilire quali tecniche di valutazione degli attributi possono essere applicate.
   c) Scrivere, almeno in parte, il programma di un valutatore semantico.

Soluzione

a) Definiamo gli attributi:
   - s, sint.; $s\ of\ I$ rappresenta un insieme; $s\ of\ S$ è l'intersezione.
   - i, ered.; i è la copia del valore finale di s.
   - $d\ of\ I$, ered., è la differenza insiemistica.

0 $S' \to S$		$i_1 \leftarrow s_1$
1 $S_0 \to \{I_1\}S_2$	$s_0 \leftarrow s_1 \cap s_2$	$i_1 \leftarrow i_0;\ \ i_2 \leftarrow i_0\ d_1 \leftarrow s_1 \setminus i_0$
2 $S_0 \to \{I_1\}$	$s_0 \leftarrow s_1$	$d_1 \leftarrow s_1 \setminus i_0$
3 $I_0 \to e, I_2$	$s_0 \leftarrow id_1 \cup s_2$	
3 $I \to e$	$s_0 \leftarrow id_1$	

b) L'attrib. s è calcolato con la prima passata (asc. o disc.). Poi una passata disc. calcola i e d.
c) . . . . . .

5

Linguaggi Formali e Compilatori: soluzione della Prova scritta 10/02/2004

Revisione 25.01.2005

Tempo 2 ore 30'. Libri e appunti personali possono essere consultati.

	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.

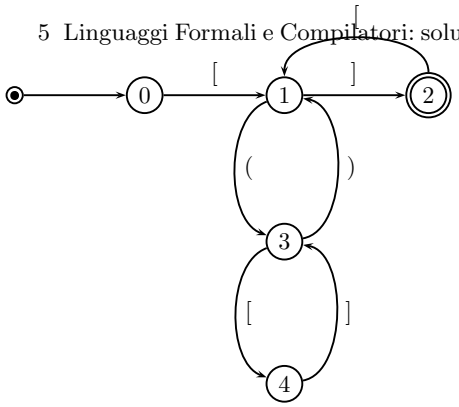
5.1 Espressioni regolari e automi finiti 20%

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.

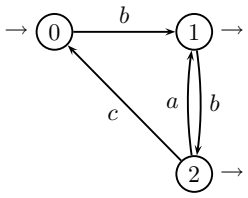
   Soluzione



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.

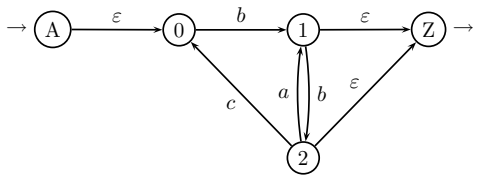


Mostrare i passaggi.

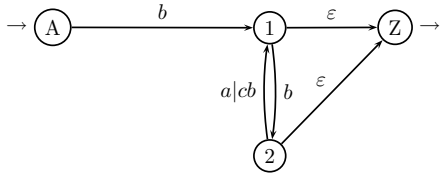
Soluzione

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

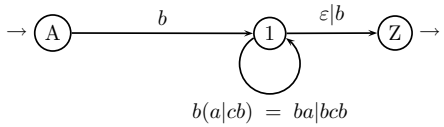
a) Aggiunta di un nuovo stato iniziale e finale:



b) Eliminazione del nodo 0

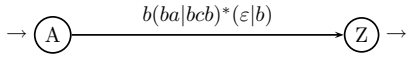


c) Eliminazione del nodo 2



$b(a|cb) \;=\; ba|bcb$

d) Eliminazione del nodo 1

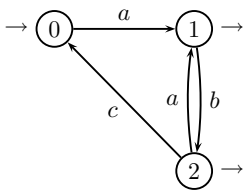


Criteri di valutazione

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.

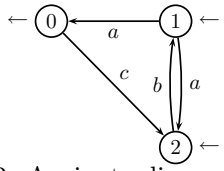


Soluzione. Si possono seguire procedimenti diversi:

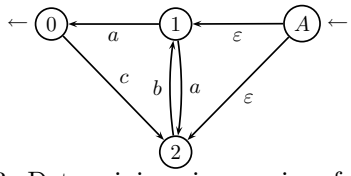
1. Si trasforma direttamente l'automa nel riconoscitore di $(L(A))^R$, poi lo si determinizza.
2. Si calcola la espressione regolare di $L(A)$ (ad es. con il metodo di eliminazione); poi da questa l'espressione di $(L(A))^R$, e infine il riconoscitore deterministico, con il metodo di Mc Naughton e Yamada.

Soluzione con il primo procedimento:

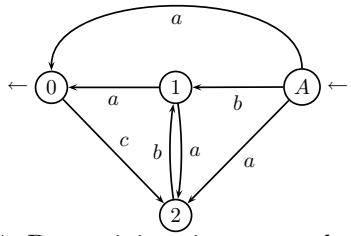
1. Inversione delle frecce e scambio tra stato iniziale e stati finali:



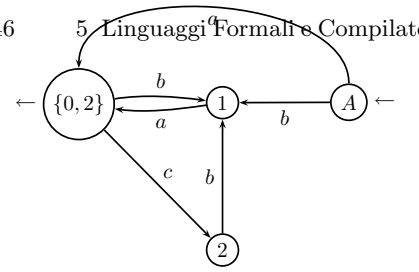
2. Aggiunta di un nuovo stato iniziale, unico:



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

$$a(ba|bca)^*(\varepsilon|b)$$

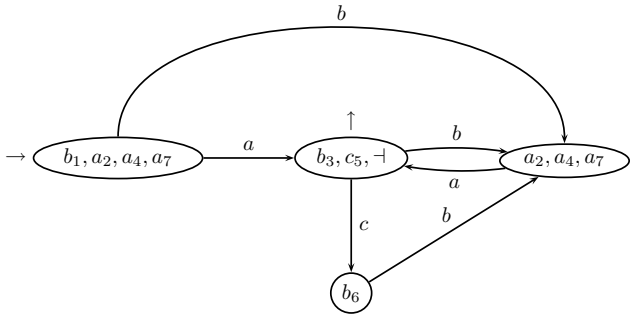
2. L'espressione del ling. speculare si ottiene riflettendo i termini concatenati dell'espressione :

$$(\varepsilon|b)(ab|acb)^*a$$

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.

5.2 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:

   $S \rightarrow$ program D^+C^+ end $\bullet$
   - - D dichiarazioni, C chiamate

   $D \rightarrow$ proc $I\ ''(''I(,I)^*\ '')''\ [V]$ begin C^+ end I
   - - I identificatore, V dich. di variabili

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

   $V \rightarrow$ int $A(,A)^*$
   $A \rightarrow I[= E_c]$
   - - E_c espressione costante

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

   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:

1 $S \to XaX$
2 $X \to aXbX$
3 $X \to bXaX$
4 $X \to \varepsilon$

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:

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

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.:

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

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

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

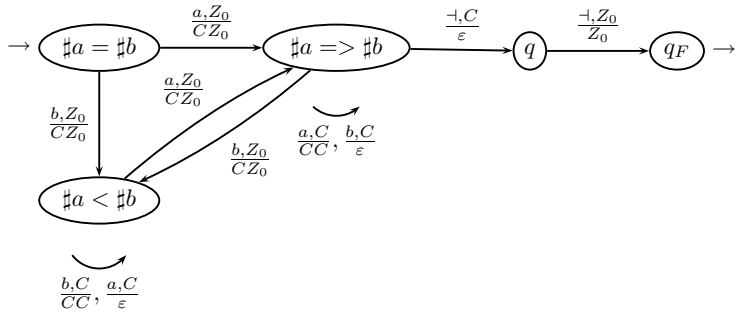
e tali stringhe sono generate da G.

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.

5.3 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

5.4 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 alfabeto 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

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

   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)
$D_U \to < D_U > D_U$	$<$
$D_U \to \varepsilon$	$>$

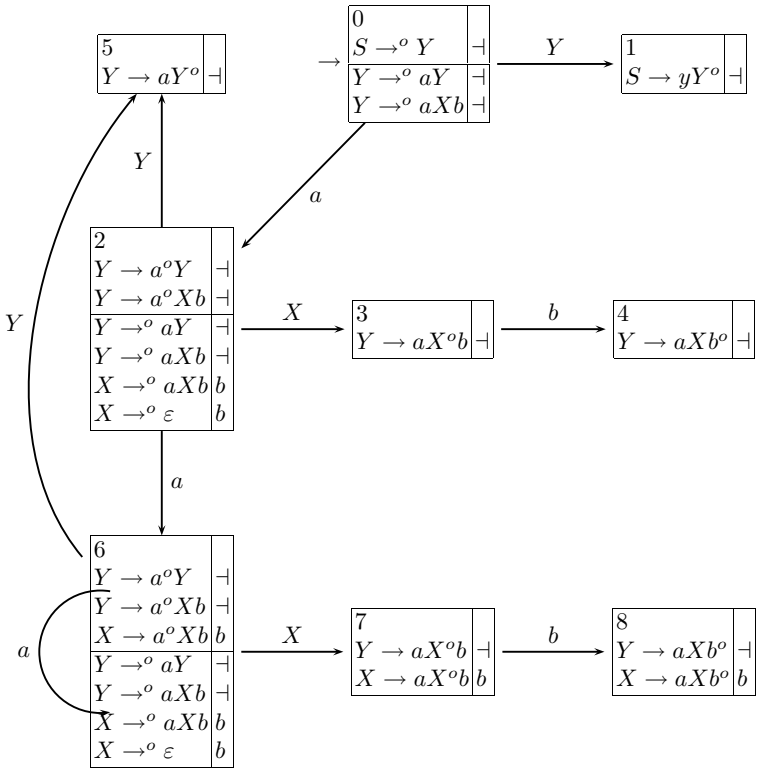
i cui insiemi guida sono riportati a fianco. Non essendo essi disgiunti per le alternative di $\overline{D}_U$, la grammatica non è LL(1). Applichiamo la fattorizzazione sinistra alle alternative che violano la condizione LL(1)e calcoliamo poi gli insiemi guida:

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

La grammatica è LL(1).

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

5.5 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)

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

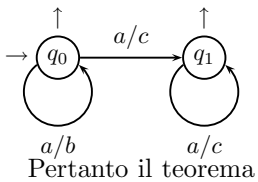
o anche

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

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.

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

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[1]. 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

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

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:

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

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

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

6

Linguaggi Formali e Compilatori: soluzioni Prova scritta 11/03/2004

Tempo 2 ore 30'. Libri e appunti personali possono essere consultati.

	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 l'allievo deve dimostrare la conoscenza di tutte e 5 le parti.

6.1 Espressioni regolari e automi finiti 20%

1. Progetto di espr. regolare
   Alfabeto terminale $\{p, \vee, \wedge\}$. Una frase è una proposizione logica che contiene al più due operatori $\wedge$.
   Esempi: p , $p \vee p$, $p \vee p \wedge p$, $p \wedge p \vee p \wedge p$
   Controesempi: $pp\vee$, $p \wedge p \wedge p \wedge p \vee p$

   Soluzione:
   L è l'unione disgiunta di tre ling., aventi risp. 0, 1 e 2 operatori $\wedge$:

   $$L = L_0|L_1|L_2$$

   Il primo è semplicemente una lista di p separati da $\vee$

   $$L_0 = p(\vee p)^*$$

   Il secondo è

   $$L_1 = L_0 \wedge L_0$$

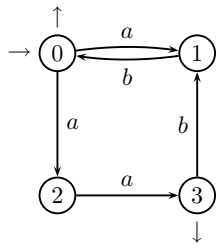
Il terzo è

$$L_1 = L_0 \wedge L_0 \wedge L_0$$

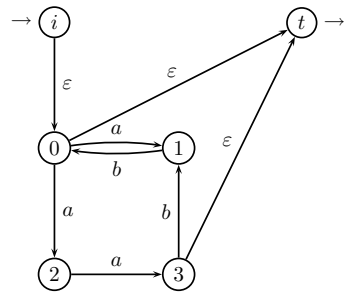
Raccogliendo le sottoespressioni comuni si ha

$$L = L_0|L_1|L_2 = [[L_0\wedge]L_0\wedge]L_0$$

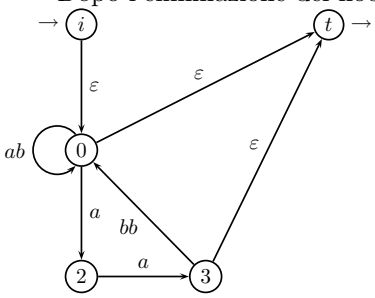
2. Calcolare, mostrando i passaggi, l'espressione regolare del ling. riconosciuto dall'automa seguente.



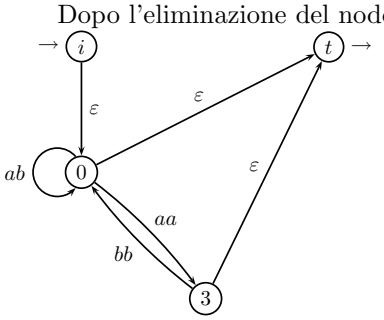
Soluzione. Applicheremo il metodo di eliminazione di Brzozowsky e McCluskey, eliminando i nodi ad es. nell'ordine 1, 2, 3, 0.



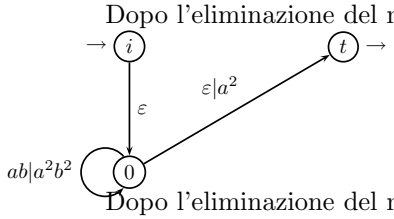
Dopo l'eliminazione del nodo 1:



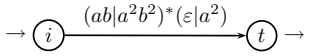
Dopo l'eliminazione del nodo 2:



Dopo l'eliminazione del nodo 3:



Dopo l'eliminazione del nodo 0:



6.2 Grammatiche 20%

1. Progettare una grammatica per generare il ling. definito dalla formula

$$L_1 = L_{Dyck} \cap ((a|c)^* ca (a|c)^*)$$

dove L_{Dyck} è il ling. di Dyck di alfabeto $\{a, c\}$.

a) Elencare le tre stringhe più corte appartenenti a $L_{Dyck} - L_1$
b) Scrivere la gramm. di L_1, preferibilmente non ambigua
c) Disegnare l'albero sintattico di una frase di L_1

Soluzione

a) L'intersezione con l'insieme regolare impone la presenza della sottostringa ca, così escludendo le stringhe di Dyck $\varepsilon, ac, aacc, \ldots$.
b) La gramm. di L_1 si ottiene con il seguente ragionamento. La frase più corta è $acac$, e può essere allungata inserendo una stringa del ling. di Dyck, denotato D, tra a e c:
$S \to aDcaDc$
$D \to aDcD|\varepsilon$ - - la nota grammatica di Dyck
Ma questa soluzione difetta, non generando le frasi, come ad es. $acacac$, che contengono 3 o più nidi al primo livello. Per generarle basta concatenare il ling. di Dyck D al ling. precedente:
$S \to aDcaDcD$
$D \to aDcD|\varepsilon$
Mancano ancora le frasi come $aacacc$ in cui le parentesi concatenate $acac$ stanno all'interno di altre parentesi. Per generare queste frasi si aggiunge la regola $S \to aSc$, ottenendo la grammatica
$S \to aDcaDcD$
$S \to aSc$
$D \to aDcD|\varepsilon$

2. Il ling. di comando di un *plotter* contiene le seguenti istruzioni:
   traccia linea spezzata:

$$\mathrm{TS}'(\text{'spessore,}'\ (\text{'}x_1, y_1\ \text{'})', \ldots, '\ (\text{'}x_m, y_m\ \text{'})'\ \text{'})'$$

   dove spessore è un numero decimale nell'intervallo $0.1 \ldots 9.9$
   il numero di punti è $m \geq 2$. Le coordinate dei punti sono numeri interi.
   traccia arco di cerchio:

$$\mathrm{TA}'(\text{'}\mathit{centro}, \mathit{raggio\_e\_coordinate}\text{'})'$$

   dove è lasciato alla vostra scelta come specificare il centro, il raggio e le coordinate dell'arco da tracciare.
   fissa colori delle linee:

$$\mathrm{FC}'(\text{'COLORE} = \mathit{colore}\text{'})'\text{BEGIN} \ldots \text{END}$$

   dove $\mathit{colore} \in \{g, r, v\}$
   e nel blocco racchiuso tra BEGIN . . . END può stare una serie di istruzioni, TS, TA e anche, nota bene, FC.

a) Scrivere la grammatica (consentita la forma EBNF)
b) Disegnare, almeno in parte, i diagrammi sintattici delle regole della grammatica
c) Disegnare un albero sintattico sufficientemente rappresentativo.

Soluzione.

$S \rightarrow\ F^+$
- - F sta per una frase elementare
$F \rightarrow\ \mathrm{FC}'(\text{'COLORE} = C\text{'})'\text{BEGIN}F^+\text{END} \mid$
- - C sta per un colore
$\mathrm{TS}'(\text{'}W, P, P(, P)^*\ \text{'})' \mid$
- - W spessore, P una coppia di interi tra parentesi
$\mathrm{TA}'(\text{'}P, I, P)')'$
- - P centro, I raggio, P coppia di angoli tra parentesi
$P \rightarrow\ '(\text{'}I, I')'$
$W \rightarrow (1 \ldots 9) \bullet (0 \ldots 9) \mid$
$0 \bullet (1 \ldots 9)$
$C \rightarrow\ g|r|v$
$I \rightarrow\ (1 \ldots 9)(0 \ldots 9)^* \mid 0$

6.3 Domanda relativa alle esercitazioni 20%

Considerate l'implementazione del compilatore *Simple* fornita di seguito e con supporto per il costrutto `for` e l'istruzione `break`. Modificatela in modo che venga riconosciuta anche l'istruzione `continue` nei cicli `for`. Assumete per `continue` 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 `continue` segue:

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

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

Per vostra utilità 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 è 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*;

Buon lavoro.

6.4 Grammatiche e analisi sintattica 20%

1. Data la grammatica EBNF

   $G_1 : \begin{vmatrix} S \rightarrow (B|bc)^* \\ B \rightarrow bBa|\varepsilon \end{vmatrix}$

   - Calcolare gli insiemi guida di G_1 verificando se essa risulta ELL(1)
   - Scrivere una grammatica G_2, in forma non EBNF, equivalente a G_1 adatta all'anlisi deterministica discendente.
   - Verificare che G_2 sia LL(k) calcolandone gli insiemi guida.

   Soluzione

   - Verifichiamo la condizione ELL(1) in tutte le sottoespressioni dove vi sono scelte.

   $$\underbrace{(B|bc)}_{e} : \ (Ini(B) \cup Seg_e(B)) \cap Ini(bc) = \{b, \dashv\} \cap \{b\} \neq \emptyset$$

   dove i seguiti di B vanno calcolati limitatamente alla sottoespressione e considerata, quindi non contengono il carattere a.

   $$(B|bc)^* : \ Ini(B|bc) \cap Seg((B|bc)^*) = \{b\} \cap \{\dashv\} = \emptyset$$

   $$Ini(bBa) \cap Seg(B) = \{b\} \cap \{a, b, \dashv\} \neq \emptyset$$

   La condizione ELL(1) è violata due volte.
   Si nota anche che G_1 è ambigua:

   $$S \Rightarrow \varepsilon \text{ opp. } S \Rightarrow B \Rightarrow \varepsilon$$

   È semplice descrivere il linguaggio definito

   $$L(G_1) = (u|bc)^* \text{ dove } u \in \{b^n a^n | n \geq 1\}$$

   - Scriviamo la grammatica in forma non estesa, eliminando al contempo l'ambiguità:

		Condizione LL(2)
$S \rightarrow$	$DS \mid$	$Ini_2(D) = \{ba, bb\}$
	$bcS \mid$	$Ini_2(bc) = \{bc\}$
	ε	$Seg(S) = \dashv$
$D \rightarrow$	$bDa \mid$	$Ini_2(bDa) = \{bb\}$
	ba	$Ini_2(ba) = \{ba\}$

   - La condizione LL(2) è soddisfatta.

   Volendo è facile ottenere una grammatica equivalente LL(1), prima espandendo D:

$S \to bDaS \mid$
$\quad bcS \mid$
$\quad \varepsilon$
$D \to bDa \mid$
$\quad \varepsilon$

poi fattorizzando a sin.:

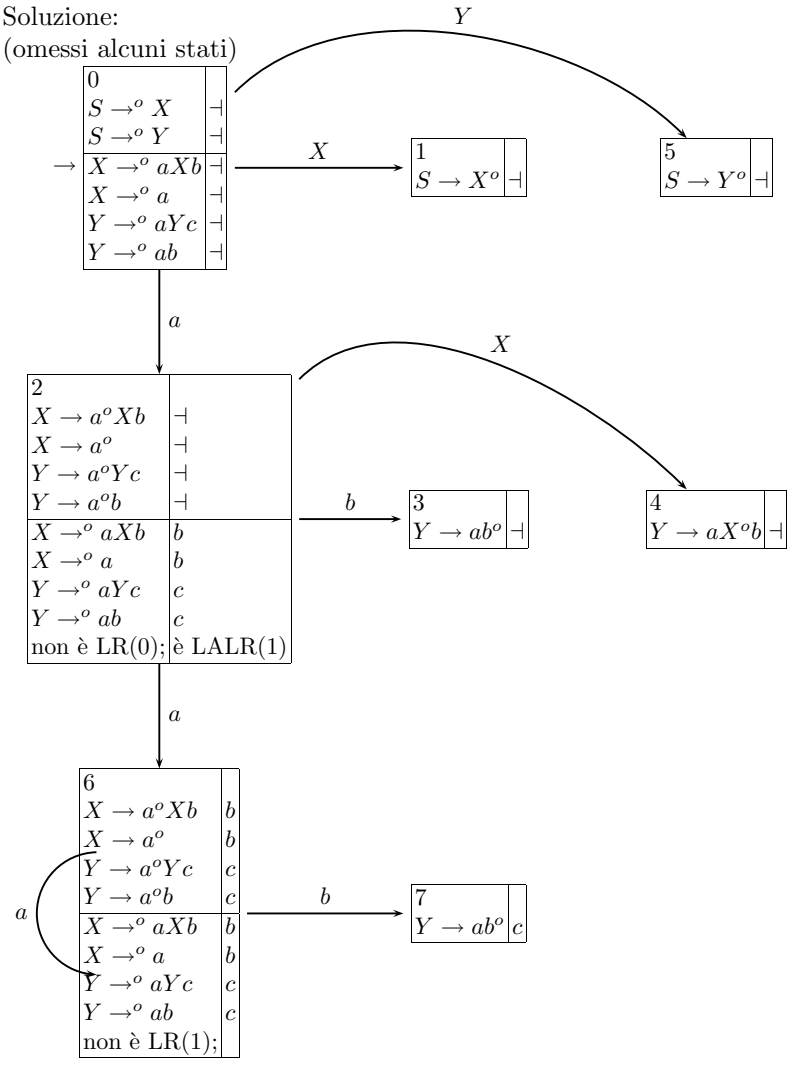
	Condizione LL(1)
$S \to bXS \mid$	b
$\quad \varepsilon$	$\dashv$
$X \to Da \mid$	a, b
$\quad c$	c
$D \to bDa \mid$	b
$\quad \varepsilon$	a

2. Grammatica LR(1)
   Per la seguente grammatica:

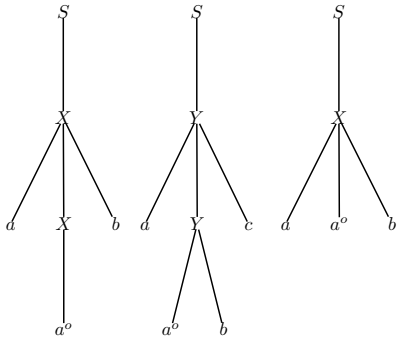
$$G_1 : \begin{array}{|l} S \to X \\ S \to Y \\ X \to aXb \\ X \to a \\ Y \to aYc \\ Y \to ab \end{array}$$

   a) Si costruisca il riconoscitore dei prefissi ascendenti di G_1
   b) Si verifichi se essa è LR(0), LALR(1), LR(1)
   c) Se necessario si trasformi la grammatica per ottenere una grammatica LR(1)

Soluzione:
(omessi alcuni stati)

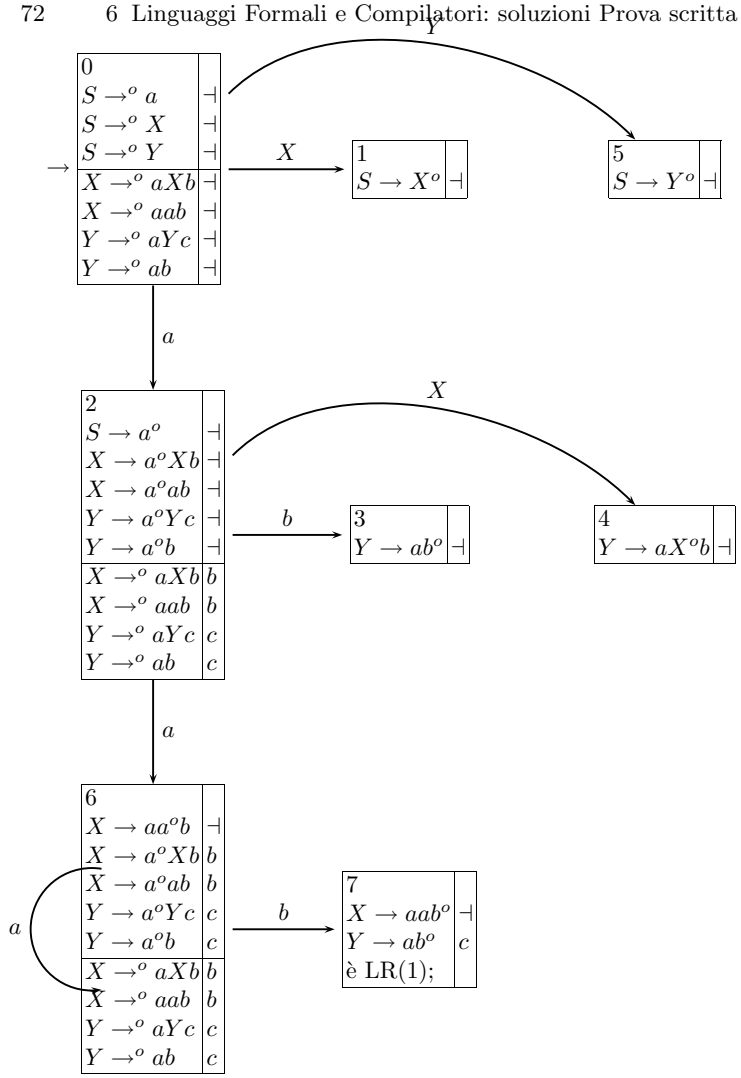


Trasformazione della grammatica. Il conflitto riduzione- spostamento in 6 si può eliminare ritardando la riduzione $X \to a^o$. Infatti provocano conflitto i primi due alberi:



Il terzo albero, equivalente al primo, rimuove il conflitto.

$$G_2 : \begin{vmatrix} S \to a \\ S \to X \\ S \to Y \\ X \to aXb \\ X \to aab \\ Y \to aYc \\ Y \to ab \end{vmatrix}$$


0
$S \to^o a$ ⊣
$S \to^o X$ ⊣
$S \to^o Y$ ⊣
$X \to^o aXb$ ⊣
$X \to^o aab$ ⊣
$Y \to^o aYc$ ⊣
$Y \to^o ab$ ⊣
Y
X
1
$S \to X^o$ ⊣
5
$S \to Y^o$ ⊣
a
2
$S \to a^o$ ⊣
$X \to a^oXb$ ⊣
$X \to a^oab$ ⊣
$Y \to a^oYc$ ⊣
$Y \to a^ob$ ⊣
$X \to^o aXb$ b
$X \to^o aab$ b
$Y \to^o aYc$ c
$Y \to^o ab$ c
X
b
3
$Y \to ab^o$ ⊣
4
$Y \to aX^ob$ ⊣
a
6
$X \to aa^ob$ ⊣
$X \to a^oXb$ b
$X \to a^oab$ b
$Y \to a^oYc$ c
$Y \to a^ob$ c
$X \to^o aXb$ b
$X \to^o aab$ b
$Y \to^o aYc$ c
$Y \to^o ab$ c
a
b
7
$X \to aab^o$ ⊣
$Y \to ab^o$ c
è LR(1);

6.5 Traduzione e semantica 20%

1. Il ling. sorgente L_1 (in forma astratta) contiene una lista di sequenze di numeri di una cifra, con l'eventuale segno meno:
   $S \rightarrow seq\ (\sharp seq\ )^*$
   $seq\ \rightarrow num\ (, num\ )^*$
   $num\ \rightarrow (-|\varepsilon)cifra$

   Esso va tradotto in un ling. pozzo L_2 di eguale alfabeto secondo le seguenti regole:

   a) Numerate $1, 2, \ldots, i, \ldots$ le sequenze presenti nella stringa sorgente. Se una sequenza ha posto i dispari, i numeri sono ricopiati. Se una sequenza ha posto pari, i numeri sono cambiati di segno.

   b) Se però la stringa sorgente contiene soltanto numeri negativi, essi sono tutti trasformati in 0

   Esempi:

sorgente	pozzo
$3, -2, -1\sharp 3, -5$	$3, -2, -1\sharp -3, 5$
$-3, -2, -1\sharp -3, -5$	$0, 0, 0\sharp 0, 0$

   È lasciata libertà di scegliere, in base alla convenienza, uno dei seguenti modi di progettare il traduttore:

   - Con uno schema di traduzione sintattica (senza attributi)
   - Con una grammatica a attributi

   a) Si progetti il traduttore nel modo prescelto

   b) Si disegnino gli alberi che calcolano la traduzione per i due esempi precedenti

   c) Si verifichi se la traduzione può essere integrata con l'analisi sintattica.

   Soluzione

   - Mediante schema di traduzione puramente sintattico

     Occorre cambiare la sintassi in modo che traduzioni diverse siano associate a nonterminali diversi. Questa strada è più complicata della successiva.

sorgente G_1	pozzo G_2
$S \to N$	$S \to N$
	- - tutti negativi
$N \to seq_N(\sharp\ seq_N)^*$	$N \to seq_N(\sharp\ seq_N)^*$
$seq_N \to num_N(,\ num_N)^*$	$seq_N \to num_N(,\ num_N)^*$
$num_N \to -cifra$	$num_N \to 0$
$S \to M$	$S \to M$
	- - non tutti negativi
$M \to (seq_1\sharp\ seq_2\sharp)^* seq_{1+}(\sharp seq_2\sharp\ seq_1)^*$	$M \to (seq_1\sharp\ seq_2\sharp)^* seq_{1+}(\sharp seq_2\sharp\ seq_1)^*$
	- - seq_1 dispari, seq_2 pari
	- - seq_{1+} dispari e non tutta neg.
$M \to \ldots$	$\ldots$
	altre regole simili
$seq_1 \to num_1(,\ num_1)^*$	$seq_1 \to num_1(,\ num_1)^*$
$num_1 \to cifra$	$num_N \to cifra$
$num_1 \to -cifra$	$num_N \to -cifra$
$seq_{1+} \to (num_1\ ,)^* num_{1+}(,\ num_1)^*$	$seq_{1+} \to (num_1\ ,)^* num_{1+}(,\ num_1)^*$
$num_{1+} \to cifra$	$num_{1+} \to cifra$
$\ldots$	$\ldots$
$seq_2 \to num_2(,\ num_2)^*$	$seq_2 \to num_2(,\ num_2)^*$
$num_2 \to -cifra$	$num_N \to cifra$
$num_2 \to cifra$	$num_N \to -cifra$

- Mediante una grammatica a attributi.

Attr.	Commento
neg	sint.; è vero se tutti i num. sono negativi
eneg	come *neg* ma ereditato
pari	ered.; vero se una seq. è di posto pari
t	sint.; la traduzione
val	il valore di una cifra, attr. lessicale

La seguente grammatica opera su alberi astratti e trascura i separatori.

1. $\overline{S}_0 \rightarrow S_1$		
	$eneg_1 \leftarrow neg_1$	$pari_1 \leftarrow false$
	$t_0 \leftarrow t_1$	
2. $S_0 \rightarrow seq_1 S_2$		
	$eneg_1 \leftarrow eneg_0$	$eneg_2 \leftarrow eneg_0$
	$pari_1 \leftarrow pari_0$	$pari_2 \leftarrow \neg pari_0$
	$neg_0 \leftarrow neg_1 \wedge neg_2$	$t_0 \leftarrow t_1 CAT t_2$
3. $S_0 \rightarrow seq_1$		
	$eneg_1 \leftarrow eneg_0$	
	$pari_1 \leftarrow pari_0$	
	$neg_0 \leftarrow neg_1$	$t_0 \leftarrow t_1$
4. $seq_0 \rightarrow cifra_1 seq_2$		
	$pari_2 \leftarrow pari_0$	
	$neg_0 \leftarrow false$	$t_0 \leftarrow f(eneg_0, pari_0, val(cifra_1))CAT t_2$
5. $seq_0 \rightarrow -cifra_1 seq_2$		
	$pari_2 \leftarrow pari_0$	
	$neg_0 \leftarrow neg_2$	$t_0 \leftarrow f(eneg_0, pari_0, val(-cifra_1))CAT t_2$
6. $seq_0 \rightarrow cifra_1$		
	$neg_0 \leftarrow false$	$t_0 \leftarrow f(eneg_0, pari_0, val(cifra_1))$
7. $seq_0 \rightarrow -cifra_1$		
	$neg_0 \leftarrow true$	$t_0 \leftarrow f(eneg_0, pari_0, val(-cifra_1))$

La funzione f è così definita:

$$f(negativita, parita, valore) \leftarrow if(negativita = true)$$
$$0$$
$$else\ if(parita = false) valore\ else - valore$$

2. Data la grammatica a attributi

1. $S_0 \to C_1 C_2$	
$\varphi_1 \leftarrow 1$	$\varphi_2 \leftarrow f_1(\lambda_2)$
$\alpha_0 \leftarrow f_2(\alpha_1, \alpha_2)$	
2. $C_0 \to C_1 B_2$	
$\varphi_1 \leftarrow f_3(\varphi_0)$	$\varphi_2 \leftarrow \varphi_0$
$\lambda_0 \leftarrow f_4(\lambda_1)$	
$\alpha_0 \leftarrow f_5(\alpha_1, \alpha_2)$	
3. $C_0 \to B_1$	
$\varphi_1 \leftarrow \varphi_0$	
$\lambda_0 \leftarrow 8$	
$\alpha_0 \leftarrow f_6(\alpha_1)$	
4. $B \to a$	
$\alpha_0 \leftarrow 5$	
5. $B \to b$	
$\alpha_0 \leftarrow f_7(\varphi_0)$	

a) Si indichi quali attributi sono ereditati e quali sintetizzati

b) Si verifichi se la grammatica è priva di errori

c) Si disegnino i grafi delle dipendenze funzionali

d) Si verifichi, riportando le spiegazioni, se la grammatica è

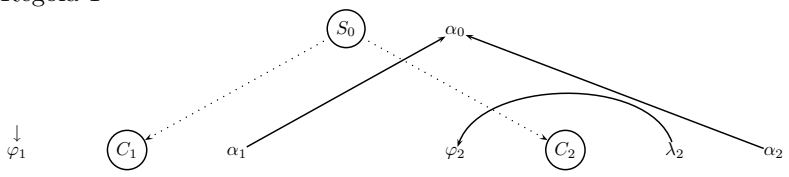
   i. del tipo L

   ii. valutabile a una scansione

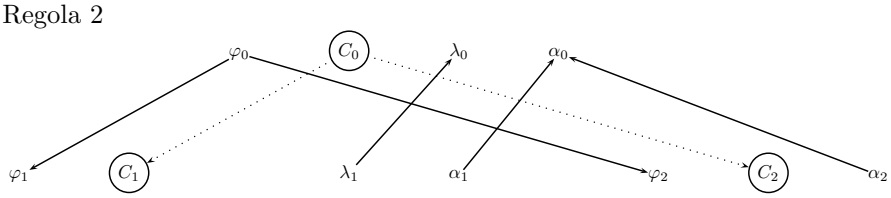
   iii. Facoltativo: si scriva una procedura del valutatore semantico.

Soluzione:
I grafi delle dipendenze funzionali delle regole sono:

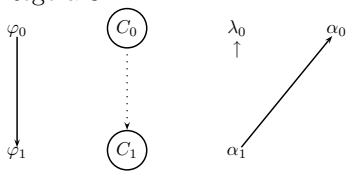
Regola 1



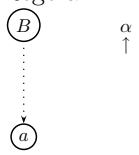
Regola 2



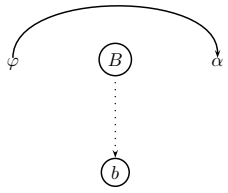
Regola 3



Regola 4



Regola 5



Il grafo 1 viola la condizione per la valutazione a 1 scansione, perché φ_2 dipende da λ_2 un attributo sintetizzato dello stesso nodo. Ciò impedirebbe

al valutatore di conoscere gli attributi ereditati della radice C_2 prima di visitare ricorsivamente il sottoalbero di C_2 per calcolare gli attributi sintetizzati di C_2.
A maggiore ragione è violata la condizione L per la valutabilità da sinistra a destra.
Tuttavia è facile vedere che la valutazione può essere fatta con due scansioni, nel modo seguente.

a) Si valuta l'attributo sint. λ che dipende soltanto da se stesso e è inizializzato nella regola 3.
b) Calcolato λ, la dipendenza $\varphi_2 \leftarrow \lambda_2$ scompare in quanto λ_2 è un valore noto.
   Poiché i rimanenti attributi (α, φ) soddisfano la condizione L, si possono calcolare con una semplice passata.

7

Linguaggi Formali e Compilatori: soluzioni Prova scritta 27/07/2004

Tempo 2 ore 30' Per superare la prova l'allievo deve dimostrare la conoscenza di tutte e 5 le parti.

7.1 Espressioni regolari e automi finiti 20%

1. Progetto di espr. regolare. Alfabeti $\Sigma_1 = \{a, b, c\}, \Sigma_2 = \{a, b\}$. Dati i linguaggi
   $R_1 = \{x \in (\Sigma_1)^* \mid x \text{ non contiene la sottostringa } aa\}$
   $R_2 = \{x \in (\Sigma_2)^* \mid x \text{ non contiene la sottostringa } aa\}$
   considerate i linguaggi

   $$L' = R_1 R_2 \qquad L'' = R_2 R_1$$

   a) Scrivete una frase di:

L'	L''	
$L' \setminus L''$	$L'' \setminus L'$	$L' \cap L''$

   b) Scrivete l'espr. reg. di L' con i soli operatori di base $\{\cup, *, .\}$

   c) Scrivete l'espr. reg. di L' usando anche l'operatore $\neg$.

   Soluzione

   Risulta

   $$R_1 = \neg((a \mid b \mid c)^* aa (a \mid b \mid c)^*) = \Big((b \mid c) \mid a(b \mid c)\Big)^* (\varepsilon \mid a)$$

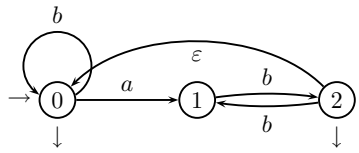
   e analogamente

$$R_2 = \neg((a \mid b)^* aa(a \mid b)^*) = \Big(b \mid ab\Big)^* (\varepsilon \mid a)$$

da cui $L' = R_1 R_2$ e $L' = R_1 R_2$, dove per R_1 e R_2 si può usare una delle due formule precedenti.

L'	ε	L''	ε	
$L' \setminus L''$	caa	$L'' \setminus L'$	aac	$L' \cap L''$ ac

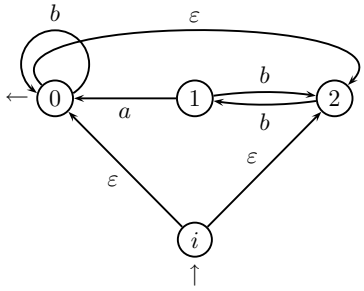
2. Dato l'automa A



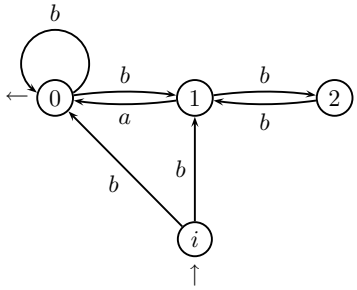
costruite, mostrando i passaggi, il riconoscitore deterministico minimo del ling. speculare $(L(A))^R$.

Soluzione

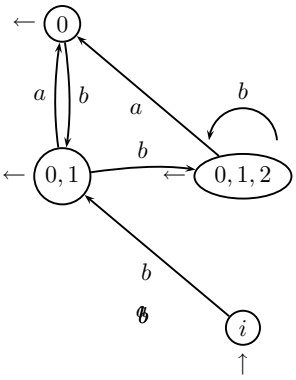
Riconoscitore del ling. riflesso:



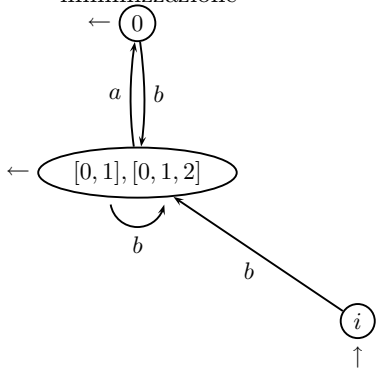
Eliminazione ε-archi :



e determinizzazione:



minimizzazione



7.2 Grammatiche 20%

1. Progettate una grammatica (consentita la forma EBNF) per il ling. di alfabeto $\{[,], a, b\}$ le cui frasi sono ben parentesizzate (b.p.) e inoltre contengono almeno una sottostringa b.p. in cui sono presenti lettere a ma non b.
   Esempi $[a], [[aaa][b]], [[[bb][a][a[aab]]]$
   Controesempi $[], [b], [bba]$
   Disegnare l'albero sint. di una frase rappresentativa.

   Soluzione

   Consideriamo i seguenti ling.:
   - L_1, stringhe b.p. prive di a e di b, ad es. $[[][]]$
   - L_2, stringhe b.p. prive di b con almeno una sosstostringa b.p contenente delle a
   - L_3, stringhe b.p. contenenti a o b o anche nulla.

   $A_1 \rightarrow {A_1}^+ \mid [A_1] \mid [\,]$
   $A_2 \rightarrow [a^+] \mid [(A_1 \mid A_2)^* A_2 (A_1 \mid A_2)^*]$
   $A_3 \rightarrow [(a \mid b \mid A_3)^*]$
   Il ling. richiesto è definito da:
   $S \rightarrow [{A_3}^* S {A_3}^*] \mid A_2$

2. Progettate la grammatica di un minilinguaggio di programmazione contenente
   - dichiarazioni di variabili.
     Le var. possono essere del tipo *int* o *record*.
     Le var. possono essere inizializzate con una costante o con un record di costanti.
   - istruzioni di assegnamento, con nella parte destra espr. aritmetiche semplici, senza parentesi
   - le dichiarazioni precedono gli assegnamenti
   - il separatore è il punto e virgola
   - nella grammatica un identificatore sarà denotato dal terminale v, una cost. dal terminale c
   - per quanto non precisato siete liberi di scegliere.

   Esempio:
   $Z : int := 8; A : int; B : record(C, D : int) := (3, 7); E : record(F : int, G : record(H, I : int)); A := B.C + 2; E.G.H := 1$

   a) Scrivere la grammatica (consentita la forma EBNF) non ambigua
   b) Disegnate il diagrammia sint. di una regola della grammatica
   c) Disegnate un albero sintattico sufficientemente rappresentativo.

Soluzione

$S \to D(;D)^* I(;I)^*$
$D \to v(,v)^* :$ int $[:=$ const $(,$ const $)^*]$
$D \to v(,v)^* : R[:= C]$
$R \to$ record $'('F(,F)^*')'$
$F \to v :$ int
$F \to R$
$C \to' ('(C \mid$ const $)(,(C \mid$ const $))^*')'$
$I \to N := [-]A((+ \mid -)A)^*$
$N \to v(\bullet v)^*)$
$A \to N \mid$ const

Domanda relativa alle esercitazioni 20%

7.3 Grammatiche e analisi sintattica 20%

Data la grammatica:
$S \to a(Bc)^*d \;\; B \to \varepsilon \mid (aS)^+$

1. Verificate se essa è ELL(1) (o ELL(k))
2. Scrivete, se possibile, una procedura del parsificatore
3. Costruite una grammatica BNF, equivalente, verificando che sia adatta all'analisi LR(1).

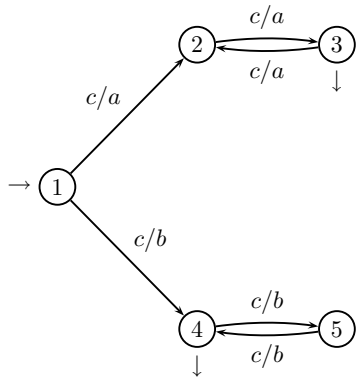
Soluzione

1. Verificate se essa è ELL(1) (o ELL(k))
   I due insiemi $ini(Bc) = \{a, c\}$ e $seg((Bc)^*) = \{d\}$ sono disgiunti.
   Inoltre per le alternative di B sono disgiunti i due insiemi $seg(B) = \{c\}$ e $ini((aS)^+) = \{a\}$.
   Infine per l'iterazione $(aS)^+$ sono disgiunti gli insiemi $ini(aS)$ e $seg(aS)^+) = seg(B) = \{c\}$.
   Pertanto la grammatica è ELL(1)
2. Scrivete, se possibile, una procedura del parsificatore
3. Costruite una grammatica BNF, equivalente, verificando che sia adatta all'analisi LR(1).
   La gramm. data equivale alla seguente
   $S \to a((aS)^*c)^*d$
   la quale equivale alla gramm. BNF G_2:
   $S \to aXd$
   $X \to \varepsilon$
   $X \to AcX$
   $A \to aSA$
   $A \to \varepsilon$

   E' facile verificare che G_2 è LL(1), quindi, a fortiori è anche LR(1). L'affermazione si può verificare costruendo il riconoscitore dei prefissi ascendenti.

7.4 Traduzione e semantica 20%

1. Dato il trasduttore finito:



Progettare il traduttore:

a) Definite con un predicato la relazione di traduzione $\tau \subseteq (\{c\}^* \times \{a,b\}^*)$ calcolata dal trasduttore:

$$\tau = \{(x,y) \mid x \ldots \wedge y \ldots\}$$

b) Scrivete uno schema di traduzione sintattica (senza attributi) per la stessa traduzione

c) Progettate un automa trasduttore deterministico a pila per la stessa traduzione.

Soluzione

a) Definite con un predicato la relazione di traduzione

$$\tau = \{(x,y) \mid x = c^n \wedge (y = a^n \text{ per } n \text{ pari}, y = b^n \text{ per } n \text{ dispari})\}$$

b) Schema di traduzione sintattica

$S_1 \to cS_2$	$S_1 \to aS_2$
$S_1 \to cS_4$	$S_1 \to bS_4$
ecc.	ecc.

c) Automa trasduttore deterministico a pila:
esso spinge nella pila le c lette, alternando tra i due stati q_0 e q_1. Incontrando il terminatore $\dashv$, a seconda che si trovi nello stato q_0 o q_1, svuota la pila emettendo una a o una b per ogni simbolo della pila.

2. Dato il ling. di Dyck L_D definito dalla sintassi G:

1	$S \to (D)$
2	$D \to (D)D$
3	$D \to \varepsilon$

Per ogni frase $x \in L(G)$ è definito il predicato:
$(\alpha(x) = true) \quad \Leftrightarrow$
(ogni coppia di parentesi (escluse le più interne) contiene lo stesso numero di coppie di parentesi)

Esempi: $x_1 = (\overbrace{(\underbrace{()()}_{2\,coppie}())}^{2\,coppie}), \qquad \alpha(x_1) = True$

$x_2 = (\overbrace{(\underbrace{()}_{1\,coppia}())}^{2\,coppie}), \qquad \alpha(x_2) = False$

a) Progettate una gramm. a attributi per calcolare il predicato α
b) Disegnate un albero decorato con gli attributi
c) Disegnate i grafi delle dipendenze funzionali e indicate queli algoritmi di valutazione si possono applicare.

Soluzione

a) Progettate una gramm. a attributi per calcolare il predicato α
Usiamo tre attributi:
- n, sint, il numero di sottoespr. b. p., in via di calcolo
- m, ered., il numero definitivo di sottoespr. b. p.
- α, sint, il predicato

Funzioni per il calcolo di n:

1	$n_0 \leftarrow n_1$
2	$n_0 \leftarrow 1 + n_2$
3	$n_0 \leftarrow 0$

Funzioni per il calcolo di m:

1	$m_1 \leftarrow m_0$	$\alpha_0 \leftarrow \alpha_1$
2	$m_1 \leftarrow m_0$ $m_2 \leftarrow m_0$	$\alpha_0 \leftarrow \alpha_1 \wedge \alpha_2 \wedge ((m_1 = n_1) \vee (n_1 = 0))$
3		$\alpha_0 \leftarrow T$

b) Disegnate un albero decorato con gli attributi
c) Disegnate i grafi delle dipendenze funzionali e indicate queli algoritmi di valutazione si possono applicare.
La grammatica non è a 1 scansione, quindi a fortiori neanche del tipo L, poiché nella regola 1 m_1, ereditato, dipende da n_1, sintetizzato.
Ma il calcolo di n può essere eseguito con una passata asc. (o disc.). I

restanti attributi, m, α possono poi essere calcolati con una visita di tipo L.

8

Linguaggi Formali e Compilatori: soluzione della Prova scritta 29/06/2004

Tempo 2 ore 30'. Per superare la prova l'allievo deve dimostrare la conoscenza di tutte e 5 le parti.

	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			

8.1 Espressioni regolari e automi finiti 20%

1. Progetto di espr. regolare. Alfabeto terminale $\{a, b, c, d\}$. Dati i linguaggi $L_X = ab^*, \qquad L_Y = d^*c$
   scrivere l'e.r. del linguaggio $mischia(L_X, L_Y)$, le cui frasi sono ottenute mescolando le frasi di L_X e di L_Y, così definito. Prese due stringhe $x \in L_X, y \in L_Y$, si segmentano in $k \geq 1$ sottostringhe, anche nulle, $x = x_1 x_2 \dots x_k,\ y = y_1 y_2 \dots y_k$. La mischia è
   $mischia(L_X, L_Y) = x_1 y_1 x_2 y_2 \dots x_k y_k$
   Es.: $mischia(ab, dc) = \{abdc, adcb, dcab, dabc, dacb\}$, $bdca \notin mischia(ab, dc)$

   Soluzione

   Nelle frasi della mischia è obbligatoria la presenza di una a e di una c, ossia

   $$mischia(L_X, L_Y) = \dots a \dots c \dots \mid \dots c \dots a \dots$$

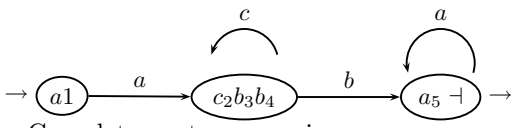
Analizzando i due linguaggi possiamo precisare le parti indicate dalle ellissi:

$$mischia(L_X, L_Y) = d^*a(b \mid d)^*cb^* \mid d^*cab^*$$

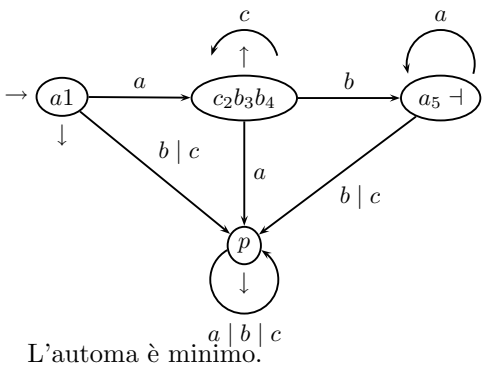
2. Costruire e poi minimizzare l'automa finito che riconosce il linguaggio $L = \neg(ac^*(b \mid ba^+))$. Riportare tutti i passaggi.

Soluzione

Costruzione automa deterministico:



Completamento e negazione:



L'automa è minimo.

8.2 Grammatiche 20%

1. Progettare una grammatica per il ling. definito dalla formula
   $L = \{a^{n_1}b^{n_2}c^{n_3} \mid (n_1 \geq n_2 \wedge n_3 \geq 1) \vee (n_1 \neq n_3)\}$
   - Per il primo predicato e per il secondo, scrivere due stringhe corte che lo soddisfano.
   - Scrivere la gramm. di L
   - Disegnare l'albero sintattico di una frase di L in cui n_1, n_2, n_3 non sono nulli.

   Soluzione

   La def. si divide in tre casi: il primo caso, e i due sottocasi $n_1 > n_3$ e $n_1 < n_3$

$S \to S_1 \mid S_2 \mid S_3$		
$S_1 \to aS_1 \mid aWC$	$W \to aWb \mid \varepsilon$	$C \to cC \mid c$
$S_2 \to aS_2c \mid aV_2V_3$	$V_2 \to av_2 \mid \varepsilon$	$V_3 \to bV_3 \mid \varepsilon$
$S_3 \to aS_3c \mid V_3V_4c$	$V_4 \to cV_4 \mid \varepsilon$	

2. Istruzioni di lettura, scrittura e frasi *for* annidabili. Un es. è la frase read(b, c); for i from b to $b+c$ do write$(i, b+(a+3+c))$; read(a, c); write(c) end

   - gli argomenti delle read sono una lista di variabili
   - gli argomenti delle write sono una lista di espr.
   - le espr. possono contenere somme, parentesi, cost. e var.
   - una istruzione `for` contiene una lista di istr. dei tipi read, write e for, separate da puntoevirgola e chiuse da `end`
   - nella grammatica una variabile sarà denotata dal terminale v, una cost. dal terminale c.
   - per quanto non precisato siete liberi di scegliere.

   a) Scrivere una grammatica (consentita la forma EBNF) non ambigua
   b) Disegnare, almeno in parte, i diagrammi sintattici delle regole della grammatica
   c) Disegnare un albero sintattico sufficientemente rappresentativo.

Soluzione

$progr \rightarrow istr(;istr)^*$
$istr \rightarrow lett \mid scri \mid ciclo$
$lett \rightarrow read'('v(,v)^*\,')'$
$scri \rightarrow write'('espr(,espr)^*\,')'$
$espr \rightarrow term(+term)^*$
$term \rightarrow c \mid v \mid'('espr')'$
$ciclo \rightarrow for\ v\ from\ espr\ to\ espr\ do\ progr\ end$

Domanda relativa alle esercitazioni 20%

Per la risoluzione dei seguenti punti si deve utilizzare l'implementazione del compilatore `Simple` che viene fornita insieme al compito.

1. Modificare la specifica dell'analizzatore lessicale da fornire a `flex`, quella dell'analizzatore sintattico da fornire a `bison` ed i file sorgenti per cui si ritengono necessarie delle modifiche in modo da estendere il linguaggio `Simple` con la possibilità di avere un costrutto `goto` simile a quello del linguaggio `C`:

   ```
   label <etichetta>;

   <codice vario>

   goto <etichetta>;
   ```

   Le modifiche devono mettere il compilatore `Simple` in condizione di analizzare la correttezza sintattica del costrutto sopra descritto e di generare una traduzione corretta per tale costrutto nel linguaggio assembler della macchina `SimpleVM`. Il compilatore deve inoltre verificare la definizione di una etichetta quando essa viene impiegata in una istruzione di goto (i.e., deve verificare che ad ogni goto corrisponda una definizione di label). Nel caso in cui i controlli semantici sopra descritti falliscano, il compilatore non si deve bloccare ma deve segnalare all'utente un messaggio di warning al termine della compilazione.

   Soluzione

   Sarà pubblicata.

8.3 Grammatiche e analisi sintattica 20%

1. Data la grammatica G_1:

	Insieme Guida
$S \to ASb$	
$S \to c$	
$A \to Aa$	
$A \to \varepsilon$	

- Calcolare gli insiemi guida di G_1, verificando se essa risulta LL(1) o LL(k)
- Se necessario, costruire una grammatica equivalente che goda della proprietà LL(1) e ricalcolare per essa gli insiemi guida.

Soluzione

	Insieme Guida	
$S \to ASb$	a, c	conflitto
$S \to c$	c	
$A \to Aa$	a	conflitto causato da ricorsione sinistra
$A \to \varepsilon$	a, c	

Osservando che il linguaggio generato $L(G_1) = (cb^* \mid a^+cb^+)$ è regolare, è facile scrivere l'automa finito deterministico ossia la grammatica lineare a destra
$q_0 \to aq_1 \mid cq_4$
$q_1 \to aq_1 \mid cq_2$
$q_2 \to bq_3$
$q_3 \to bq_3 \mid \varepsilon$
$q_4 \to bq_4 \mid \varepsilon$
la quale è per costruzione LL(1).

2. Grammatica LR(1)
   Per la seguente grammatica G_1 : :

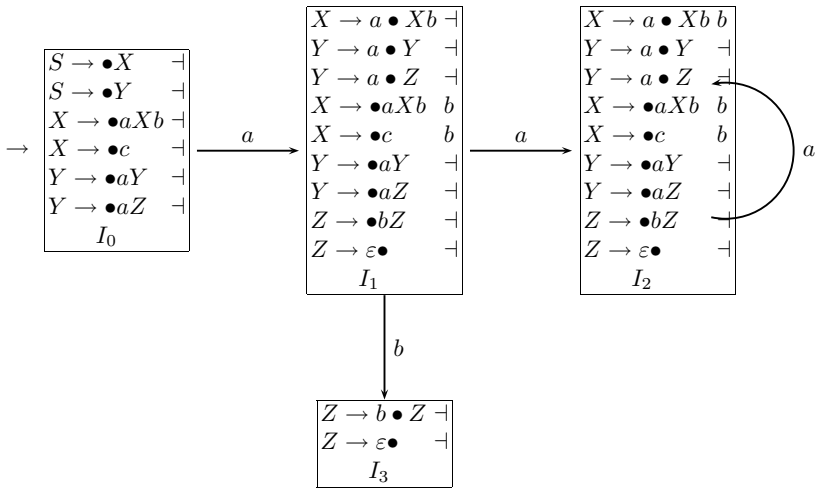
$$\begin{array}{|ll} S \to X & S \to Y \\ X \to aXb & X \to c \\ Y \to aY & Y \to aZ \\ Z \to bZ & Z \to \varepsilon \end{array}$$

   a) Costruire il riconoscitore dei prefissi ascendenti di G_1
   b) Verificare se la grammatica è LR(1)
   c) Se necessario, trasformare la grammatica per ottenere una grammatica LR(1)
   d) Scrivere un predicato caratteristico per definire il linguaggio

$$L(G_1) = \{x \in \{a, b, c\}^* \mid \text{ predicato caratteristico}\}$$

Soluzione

L'ultima domanada ha lo scopo di far osservare che le frasi sono di due tipi: $a^n c b^n, n \geq 0$ e $a^+ b^*$. Intuitivamente, l'analizzatore a spostamento e riduzione può decidere se la frase appartiene al primo o al secondo tipo, quando, dopo aver impilato tutte le a, incontra c o b. Questo ragionamento porta a ritenere che la grammatica sia LR(1), come viene confermato dalla costruzione del riconoscitore dei prefissi ascendenti, di cui mostriamo soltanto la parte interessante:



8.4 Traduzione e semantica 20%

1. Tradurre gli identificatori di alfabeto $\Sigma = \{a \dots z\} \cup \{0 \dots 9\} \cup \{\_\}$ come sotto indicato:

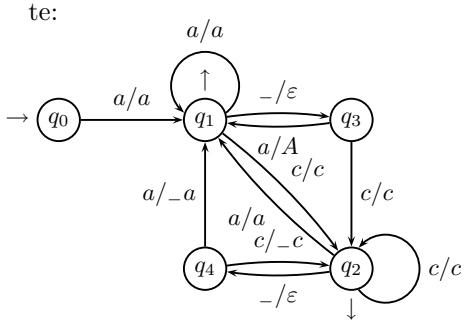
sorgente	⇒ pozzo	nota
`fine_ciclo`	`fineCiclo`	il tratto tra due lettere cade e la 2^a lettera diviene maiuscola
`fine_del_mod`	`fineDelMod`	
`ciclo_2_33_a`	`ciclo2_33a`	il tratto tra una lettera e una cifra cade; resta tra due cifre
`ciclo23`	`ciclo23`	nessuna modifica

Progettare il traduttore:

a) Con uno schema di traduzione sintattica (senza attributi)
b) Con un automa trasduttore finito
c) Verificare sul vostro progetto che la stringa `a_bb_3_4_c` si traduce in `aBb3_4c`

Soluzione

La risposta alle due prime domande è sostanzialmente la stessa: l'automa trasduttore o la grammatica di traduzione (ossia lo schema) corrispondente:



G. sorgente	*G. pozzo*
$q_0 \to aq_1$	$q_1 \to aq_1$
$q_1 \to aq_1$	$q_0 \to aq_1$
$q_1 \to \varepsilon$	$q_0 \to \varepsilon$
$q_1 \to_{\_} q_3$	$q_1 \to q_3$
ecc.	
$q_4 \to_{\_} q_2$	$q_4 \to q_2$

2. Progettare una gramm. a attributi che esegue dei controlli su un programma. Il programma è una lista di istruzioni di lettura e scrittura, ad es.:
`read(a,b,e) read(c) read(a) write(c,a) read(a) write(a,d)`
I controlli sono:
   a) una variabile non può essere scritta se prima non è stata letta, es. la d in `write(a,d)`
   b) una variabile non può essere riletta se non è stata scritta, es. la a in `read(a)`
   c) una variabile letta deve essere scritta, es. la e in `read(a,b,e)`

   La sintassi suggerita è:
   $S' \to S$
   $S \to write(V)S \; S \to read(V)S \; S \to \varepsilon$
   $V \to var, \; V \qquad V \to var$
   dove var ha l'attributo lessicale $id\ of\ var$ che è il nome della variabile.

   a) Definire gli attributi necessari per effettuare almeno i controlli a,b,c e scrivere le corrispondenti funzioni semantiche
   b) Disegnare i grafi delle dipendenze funzionali
   c) Studiare quale algoritmo di valutazione semantica è applicabile
   d) Scrivere almeno una procedura semantica

Soluzione

Useremo gli attributi seguenti:
s, le variabili scrivibili, ereditato;
a, gli argomenti di una operasione, sintetizzato
α, predicato di validità, sintetizzato.
Grammatica a attributi:

$S'_0 \to S_1$	$\alpha_0 := \alpha_1$	
$S_0 \to write(V_1)S_2$	$\alpha_0 := (a_1 \subseteq s_0) \wedge \alpha_1$	$s_1 := s_0 \setminus a_1$
$S_0 \to read(V_1)S_2$	$\alpha_0 := (a_1 \cap s_0 = \emptyset) \wedge \alpha_1$	$s_1 := s_0 \cup a_1$
$S \to \varepsilon$	$\alpha_0 := true$	
$V_0 \to var_1, \; V_2$	$a_0 := \{id_1\} \cup a_2$	
$V_0 \to var_1$	$a_0 := \{id_1\}$	

Le dipendenze funzionali sono del tipo L.

9

Linguaggi Formali e Compilatori - Prof. Breveglieri e Crespi Reghizzi - Soluzione Prova scritta 07/02/2005

Tempo 2 ore per chi è esonerato dalle prime due parti, 3 ore per gli altri.
AVVERTENZA: L'esame è diviso in 5 parti

1. Espr. reg. e aut. finiti: *Esonerato chi ha superato il compitino*
2. Grammatiche *Esonerato chi ha superato il compitino*
3. Esercitazioni Flex Bison (fascicolo separato) *Obbligatorio per tutti*
4. Grammatiche e analisi sintattica *Obbligatorio per tutti*
5. Traduzione e semantica *Obbligatorio per tutti*

9.1 Espressioni regolari e automi finiti 20%

1. Trovare la (o le) stringa più breve che appartiene al linguaggio di alfabeto $\{a, b\}$ generato dall'espressione regolare R_1:

$$R_1 = (ab^+)\left(\neg(ab \mid b^*)\left((b^*ab^*a)^+ \cap (aaa)^*\right)\right)$$

   Soluzione

$$\text{min di } (ab^+) = ab$$

$$\text{min di } (\neg(ab \mid b^*)) = a$$

$$\text{min di } \left((b^*ab^*a)^+ \cap (aaa)^*\right) = aaaaaa$$

   La frase più breve è pertanto aba^7
2. Trovare la (o le) stringa più breve che appartiene al linguaggio di alfabeto $\{a, b\}$ generato dall'espressione regolare R_2:

$$R_2 = \neg\left(a^* \mid (a^*b^+)^+\right)^* \mid b(b \mid a)^+bb^+$$

Soluzione

essendo il primo termine

$$\neg \left(a^* \mid (a^* b^+)^+\right)^*$$

il complemento del linguaggio universale, esso è vuoto. Dal secondo termine

$$b(b \mid a)^+ bb^+$$

si ha:

$$\text{min di } b \ = b$$

$$\text{min di } (b \mid a)^+ \ = a, b$$

$$\text{min di } bb^+ \ = bb$$

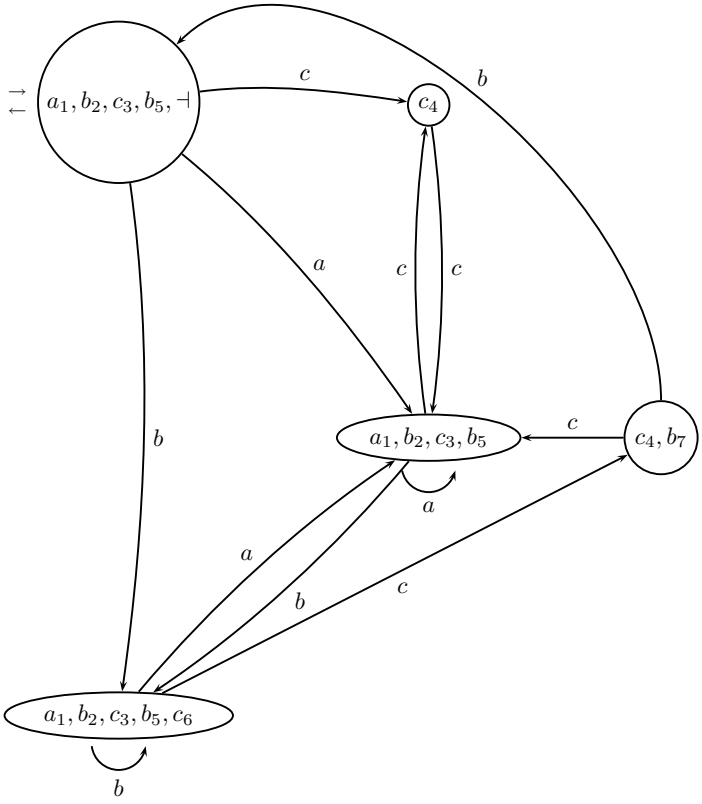
La risposta è pertanto *babb* e *bbbb*

3. È data l'espressione regolare seguente:

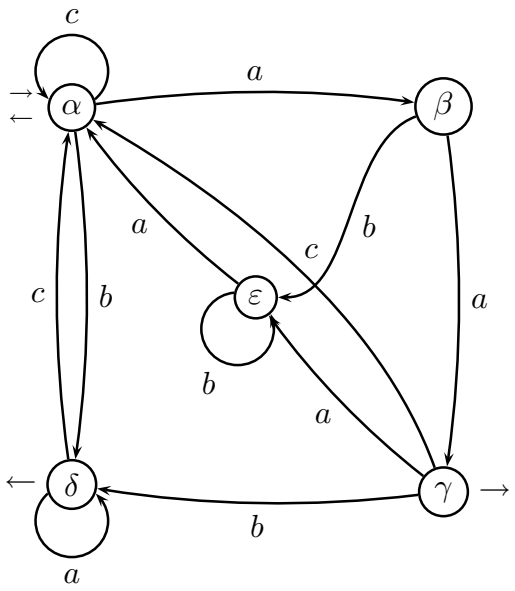
$$R = ((a \mid b^* \mid cc)^* bcb)^*$$

Tramite l'algoritmo di McNaughton-Yamada costruire l'automa deterministico che riconosce il linguaggio regolare $L(R)$.

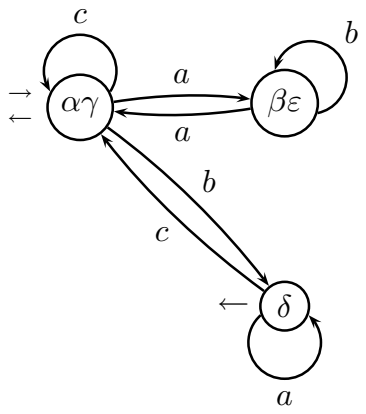
Soluzione



4. Minimizzare l'automa seguente.



Soluzione



9.2 Grammatiche 20%

1. Si consideri il ling. di Dyck di alfabeto $\{ '(','),'[',']' \}$.
   a) Trovare la grammatica BNF che genera tutte e sole le stringhe di Dyck in cui la parentesi tonda chiusa deve essere immediatamente seguita dalla parentesi quadra aperta.
   b) Generalizzando il caso precedente, trovare la grammatica BNF che genera tutte e sole le stringhe di Dyck in cui la parentesi tonda chiusa deve essere immediatamente seguita dalla parentesi quadra aperta oppure dalla parentesi tonda chiusa.

   Soluzione

   a)
   $$S \to \{S\}S \mid [S]S \mid (S)Q \mid \varepsilon$$
   $$Q \to [S]S$$

   b)
   $$S \to \{S\}S \mid [S]S \mid (T)Q \mid \varepsilon$$
   $$Q \to [S]S \qquad T \to (T) \mid S \mid \varepsilon$$

2. Costruire la grammatica EBNF della dichiarazione della `struct` del ling. C, con le regole seguenti:
   a) l'alfabeto comprende le lettere minuscole, le cifre, le due parentesi '{' e '}' e punto-e-virgola
   b) i tipi semplici per i campi delle struct sono: `int`, `char`, `float`
   c) le `struct` possono essere annidate
   d) la grammatica non deve essere ambigua
   e) esempio

```
struct {
  int campo1;
  char campo2;
  struct {
    float elema;
    char elemb;
  } alfa;
  float campo3;
} beta;
```

Soluzione

$S \rightarrow \mathsf{struct}\ \{F\}I;$
$F \rightarrow ((\mathsf{int}\ \mid \mathsf{char}\ \mid \mathsf{float}\ )I; \mid S)^+$
$I' \rightarrow [a \ldots z, A \ldots Z][a \ldots z, A \ldots Z, 0 \ldots 9]^*$

9.3 Domanda relativa alle esercitazioni

Vedi fascicolo separato.

9.4 Grammatiche e analisi sintattica 20%

1. Calcolare gli insiemi guida e verificare se le regole di produzione soddisfano la condizione LL(1).

Regola	Insieme Guida
$S \to ASB$	
$S \to c$	
$A \to c$	
$A \to aA$	
$A \to B$	
$B \to bBc$	
$B \to \varepsilon$	

Soluzione

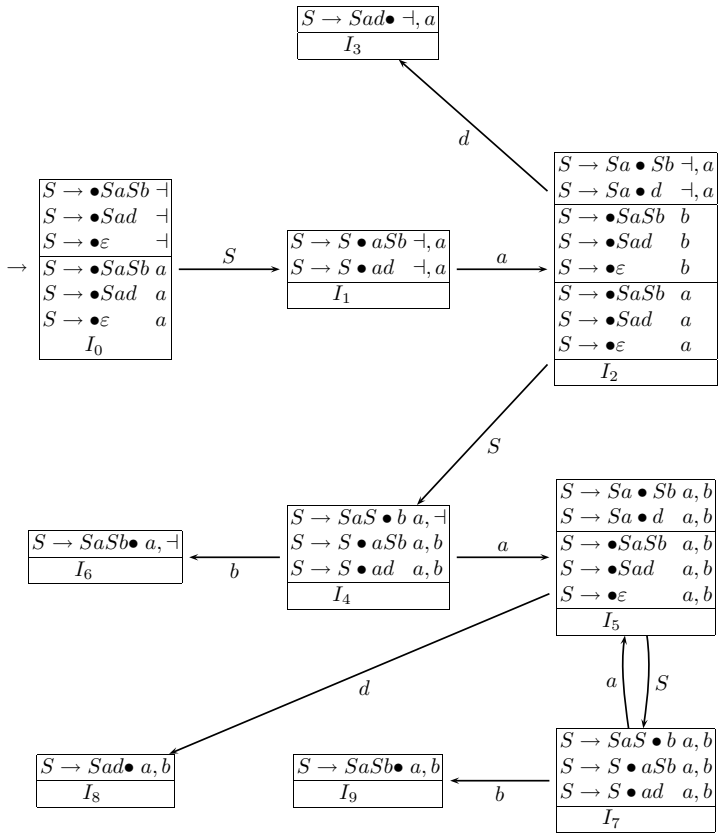
Regola	Insieme Guida
$S \to ASB$	$\text{Ini}(ASB) = \{a, b, c\}$
$S \to c$	c
$A \to c$	c
$A \to aA$	a
$A \to B$	$\text{Ini}(B) \cup \text{Seg}(A) = \{b\} \cup \{a, b, c\} = \{a, b, c\}$
$B \to bBc$	b
$B \to \varepsilon$	$\text{Seg}(B) \cup \text{Seg}(A) \cup \text{Seg}(S) = \{c, \dashv\} \cup \{a, b, c\} \cup \{b, \dashv\} = \{a, b, c, \dashv\}$

Tutti i nonterminali violano la condizione LL(1).
Nota: la grammatica è ambigua.

2. Costruire il riconoscitore deterministico dei prefissi ascendenti e indicare in quali stati sono violate le condizioni LR(1), LALR(1) e LR(0).

$$S \to SaSb \qquad S \to Sad \qquad S \to \varepsilon$$

Soluzione



Tutti e soli gli stati con la candidata $S \to \bullet\varepsilon$ non sono LR(0).
Tutti gli stati sono LR(1): ad es. in I_5 il conflitto LR(0) tra la riduzione $S \to \bullet\varepsilon$ e lo spostamento, scompare prospettando il successivo carattere d, l'unica etichetta terminale di un arco uscente.
La grammatica è anche LALR(1). Infatti fondendo insieme gli stati aventi lo stesso insieme di regole marcate:

$$I_2, I_5 \qquad I_3, I_8 \qquad I_4, I_7 \qquad I_4, I_9$$

e unendo gli insiemi di prospezione a parità di regla marcata, si ottiene il riconoscitore LALR(1), in cui nessuno stato ha conflitto spostamento-riduzione. Si può anche osservare che il riconoscitore LR(1) è privo di

stati contenenti due riduzioni, condizione che permette di affermare che la grammatica è LALR(1)

3. Domanda facoltativa. Dato il linguaggio delle espressioni in forma polacca postfissa con un operatore '+' e una variabile v, trovare una grammatica LL(1) che lo generi.
Esempio di frase del linguaggio: $vv + v+$

Soluzione

La grammatica più immediata del linguaggio

$$E \to EE+ \qquad E \to v$$

è ricorsiva a sx, quindi inadatta all'analisi LL(1). Trasformando la trasformando la ricorsione sx in dx, si ottiene la grammatica LL(1) seguente:

	Insieme Guida
$E \to vE'$	v
$E' \to E + E'$	v
$E' \to \varepsilon$	$\dashv, +$

9.5 Traduzione e semantica 20%

1. Dati gli alfabeti sorgente e pozzo

$$\Sigma = \{a, b, c\} \qquad \Delta = \{a, b\}$$

considerare la traduzione

$$\tau(ucv) = vu^R \qquad \text{dove } u, v \in \{a, b\}^+$$

a) Scrivere uno schema di traduzione (oppure una gramm. di traduzione) puramente sintattico per definire la traduzione τ e disegnare l'albero sintattico della traduzione

$$\tau(aabcba) = babaa$$

b) Verificare se è possibile calcolare la traduzione con un trasduttore a pila deterministico, ovvero con un parsificatore LL(1) o LR(1).

Soluzione

a) La grammatica di traduzione (con i caratteri dell'alfabeto pozzo sottolineati) è:

	Insieme Guida LL(2)
$S \to aY\underline{a}$	a
$S \to bY\underline{b}$	b
$Y \to aY\underline{a}$	a
$Y \to bY\underline{b}$	b
$Y \to cX$	c
$X \to a\underline{a}X$	$\{aa, ab\}$
$X \to b\underline{b}X$	$\{ba, bb\}$
$X \to a\underline{a}$	$a \dashv$
$X \to b\underline{b}$	$b \dashv$

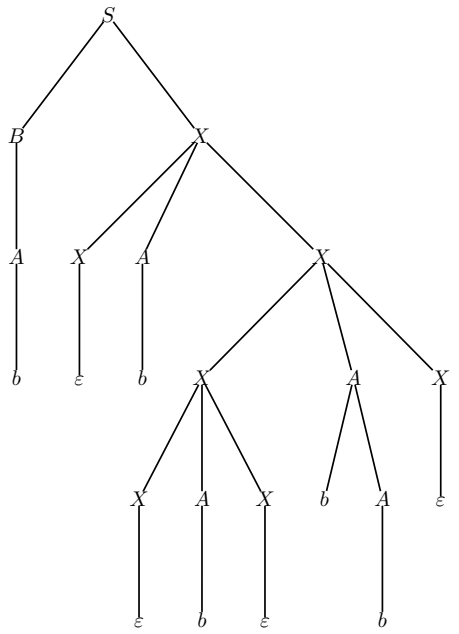
b) Essendo la sintassi deterministica LL(2), la traduzione può essere calcolata da un parsificatore a discesa ricorsiva, e quindi anche da un trasduttore a pila deterministico.

2. È data la Gramm. ad Attributi:

$S_0 \to B_1 X_2$	$\delta_0 \leftarrow \delta_2 \qquad \gamma_2 \leftarrow \beta_1$
$X_0 \to X_1 A_2 X_3$	$\delta_0 \leftarrow \delta_1 + \delta_3 + (\text{if } (\gamma_0 = \alpha_2) \text{ then 1 else 0})$
	$\gamma_1 \leftarrow \gamma_0 \qquad \gamma_3 \leftarrow \gamma_0$
$X_0 \to \varepsilon$	$\delta_0 \leftarrow 0$
$B_0 \to A_1$	$\beta_0 \leftarrow \alpha_1$
$A_0 \to bA_1$	$\alpha_0 \leftarrow \alpha_1 + 1$
$A_0 \to b$	$\alpha_0 \leftarrow 1$

a) Indicare gli attributi ereditati e sintetizzati

b) Disegnare sull'albero gli attributi e le frecce delle dipendenze funzionali tra di essi.



c) Quanto vale l'attributo δ nella radice?

d) Verificare se la gramm. a attributi è del tipo L e scrivere almeno in parte le procedure del valutatore ricorsivo.

e) Domanda facoltativa. Spiegare che significato ha il calcolo sugli alberi astratti, definito dalla gramm. a attributi, e che cosa il valore di δ rappresenta.
Scrivere una gramm. ad attributi puramente sintetizzata che calcola lo stesso valore di δ.

Soluzione

a) Indicare gli attributi ereditati e sintetizzati:

sintetizzati	ereditati
δ per S, X	γ per X
β per B	
α per A	

b) Disegnare sull'albero gli attributi e le frecce delle dipendenze funzionali tra di essi.

c) Quanto vale l'attributo δ nella radice? 2

d)
- La gramm. a attributi è del tipo L:
  l'attrib. ered. dipende nella 2 dall'attrib ered. del padre, nella 1 dall'attrib. sint. del fratello precedente;
  l'attrib. sint. δ (nella 2) dipende dall'attributo ered. del padre e da attrib dei figli;
  gli altri attrib. sint. dipendono solo da attrib. sintet.

- scrivere almeno in parte le procedure del valutatore ricorsivo:
  Poiché la sintassi è astratta e ambigua, non si può costruire un parsificatore integrato con il valutatore. Il valutatore opera dunque sull'albero sintattico già costruito. Il suo codice contien una procedura per simbolo nonterminale, avente i seguenti argomenti:
  un puntatore al nodo dell'albero;
  un argomento d'ingresso per l'attributo ereditato γ dove è rilevante;
  un argomento d'uscita per l'attributo sintetizzato.
  Il corpo delle procedure A e X contiene un condizionale guidato dalla produzione sintattica applicata nel nodo.

e) Domanda facoltativa.
- Spiegare che significato ha il calcolo sugli alberi astratti, definito dalla gramm. a attributi, e che cosa il valore di δ rappresenta.
  Il sottoalbero B è il numero n (rappresentato dalla stringa a^n) da cercare nel resto dell'albero, che non è altro che una albero binario con un numero m (la stringa a^m) associato a ogni nodo interno.
  Il valore di δ è il numero di comparse di n nell'albero.

- Scrivere una gramm. ad attributi puramente sintetizzata che calcola lo stesso valore di δ:
  Gli attributi α e β si conservano, e si eliminano gli altri. Si usa un attributo sintet. μ, un multi-insieme di interi, che conterrà il numero di comparse delle stringhe a^m presenti. Nella produzione

$$X_0 \to X_1 A_2 X_3$$

  l'attributo μ_0 è calcolato come unione dei multi-insiemi μ_1, μ_3 e dell'insieme $\{\alpha_2\}$
  Nella radice, si assegna a δ_0 la molteplicità di β_1 in μ_2.

Nota: l'eliminazione degli attributi ereditati ha richiesto l'uso di un attributo sintetizzato di dominio complesso (multi-insieme).

10

Linguaggi Formali e Compilatori: soluzioni della Prova scritta 25/02/2005

AVVERTENZA: L'esame è diviso in 5 parti:
1 Espr. regolari e automi finiti
2 Grammatiche
3 Esercitazioni Flex Bison (fascicolo separato)
4 Grammatiche e analisi sintattica
5 Traduzione e semantica
Per superare la prova, l'allievo deve dimostrare la conoscenza di tutte e cinque le parti.

10.1 Espressioni regolari e automi finiti 20%

1. Dato il linguaggio di alfabeto $\{a, b, c\}$

$$L = \Big((b \mid c)(ab^*ab^*)^*\Big)^+ \; - \; \Big(c(a \mid b \mid c)^* \mid (a \mid b \mid c)^*aa(a \mid b \mid c)^*\Big)$$

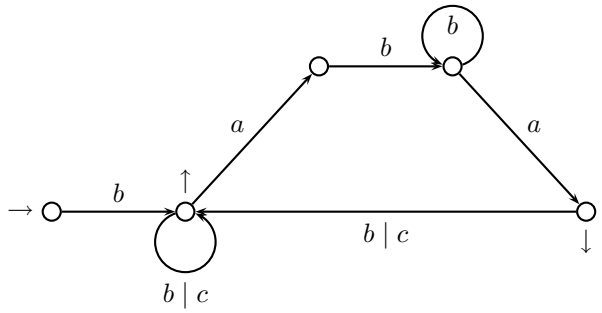
   a) Trovare la (o le) stringa più breve che appartiene al linguaggio L.
   b) Scrivere una espr. reg. di L con i soli operatori $\{. \mid {}^* {}^+\}$
   c) Costruire, descrivendo il procedimento applicato, l'automa riconoscitore deterministico di L.

Soluzione

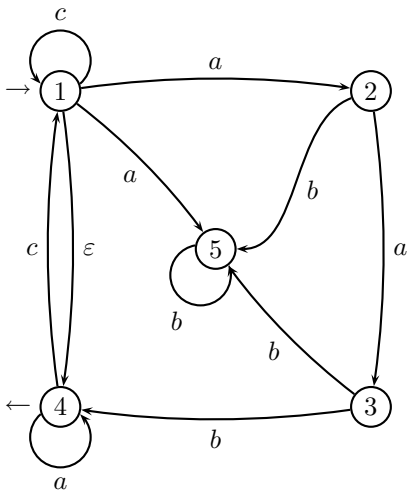
a)
$$L = L_1 - L_2 = L_1 \cap \neg L_3$$
$$L_3 = \{x \mid x \text{ inizia con } c \vee x \text{ contiene la sottostringa } aa\}$$
Ne segue che la stringa più breve di L è b.

b) Seguendo la precedente descrizione di L si ha
$$L = b\left(\left(ab^+a \mid \varepsilon\right)(b \mid c)\right)^*\left(ab^+a \mid \varepsilon\right)$$

c) Costruzione semiintuitiva del riconoscitore deterministico di L, seguendo la struttura dell'espressione regolare

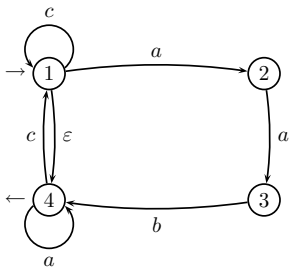


2. Determinizzare e poi minimizzare l'automa seguente.

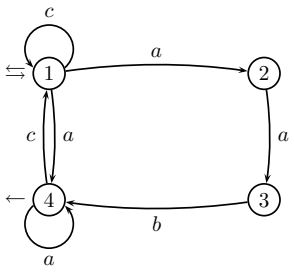


Soluzione

L'automa non è pulito, lo stato 5 si può eliminare:

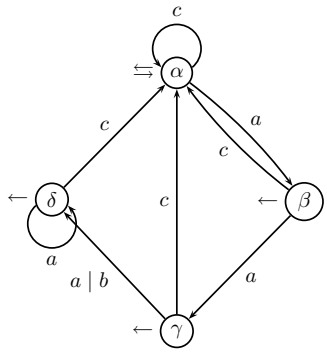


Eliminando la mossa spontanea si ottiene un automa indeterministico nello stato 1



Si calcola la seguente tabella delle transizioni:

	a	b	c	
1	$2,4$	$-$	1	α
$2,4$	$3,4$	$-$	1	β
$3,4$	4	4	1	γ
4	4	$-$	1	δ
$-$	$-$	$-$	$-$	



L'automa è minimo. Infatti, dalla colonna b si vede che γ non è equivalente a niente, dunque dalla colonna a si vede che β non è equivalente a niente e infine sempre dalla colonna a si vede che α non è equivalente a niente.

10.2 Grammatiche 20%

1. Progettare la grammatica G_1 del sottolinguaggio di Dyck di alfabeto $\Sigma =$ { '[', ']', '(', ')' } tale che ogni coppia di parentesi quadre [ ] contenga un numero pari di coppie di parentesi (qualsiasi).
   Esempi: [([])], ([()()])
   Controesempio: [(())()]

   Soluzione

   $$S \to P \mid D$$
   $$P \to [P]D \mid (P)D \mid (D)P \mid \varepsilon$$
   $$D \to [P]P \mid (P)P \mid (D)D$$

   P genera solo nidi pari, D solo nidi dispari. Siccome si ha solo $[P]$, ma non $[D]$, il vincolo è rispettato.

2. Il ling. L da definire sono le formule del calcolo dei predicati del primo ordine (CPPO). I simboli che possono comparire in una formula sono:
   connettivi logici: $\wedge, \vee, \Rightarrow$, elencati in ordine di precedenza;
   quantificatori: $\forall, \exists$;
   parentesi tonde;
   virgola;
   predicati: denotati da $p1, p2, \ldots$, cioè da p seguito da un intero;
   variabili individuali: denotate da $x1, x2, \ldots$, cioè da x seguito da un intero.

   Esempi:

$$\forall x1 \forall x2 \Big(p5(x1, x2) \Rightarrow (p1(x1) \wedge p3(x2)) \Big)$$

$$\forall x9 \Big(p7(x9) \wedge p2(x9) \wedge \exists x10 \big(p4(x10) \vee p5(x9, x10) \big) \Big)$$

   Fate riferimento alla vostra conoscenza del CPPO, per individuare le formule da definire con la grammatica.
   a) Progettare una grammatica G EBNF non ambigua per il ling. L
   b) (Facoltativo) Discutere se le frasi di $L(G)$ soddisfano le condizioni per essere delle formule ben formate del CPPO.

   Soluzione

   a) Grammatica:

$$\begin{aligned}
&S \to Q^{*}\,'('I')' \\
&Q \to (\forall \mid \exists)V \\
&I \to O(\Rightarrow O)^{*} \\
&O \to A(\vee A)^{*} \\
&A \to T(\wedge T)^{*} \\
&T \to P \mid S \\
&V \to x[1..9][0..9]^{*} \\
&P \to p[1..9][0..9]^{*}\,'('V(','V)^{*}\,')'
\end{aligned}$$

   b) Variabili quantificate ma non usate; variabili quantificate più volte nello stesso campo; formule aperte (cioè dove non tutte le variabili sono quantificate); predicati con grado variabile.

3. (facoltativo) Per la grammatica G_2 seguente :

$$S \to SA \mid Bb \mid a$$
$$A \to aS \mid \varepsilon$$
$$B \to bB \mid b$$

a) Dimostrare che la grammatica G_2 è ambigua.
b) Trovare una grammatica G_3 non ambigua tale che $L(G_3) = L(G_2)$.

Soluzione

a) Basta osservare le derivazioni:

$$S \Rightarrow a \quad S \Rightarrow SA \Rightarrow aA \Rightarrow a\varepsilon = a$$

Inoltre la grammatica è ricorsiva bilaterale:

$$S \Rightarrow SA \Rightarrow SaS$$

e circolare:

$$S \Rightarrow SA \Rightarrow S\varepsilon = S$$

b) L_2 è regolare! Infatti

$$L(B) = b^+$$

e sostituendo B ed A nelle regole si ottiene

$$S \to SaS \mid S \mid b^+b \mid a$$

Eliminata la regola circolare, si vede che il ling. è una lista avente come separatore a e come elemento una stringa di $(a \mid bb^+)$

$$L_2 = (a \mid bb^+)(a(a \mid bb^+))^*$$

ed è facile trovare una gramm. lineare a destra non ambigua.

Si ricorda che in generale il problema se un linguaggio libero sia regolare è indecidibile; in questo caso però si riesce a deciderlo facilmente.

10.3 Domanda relativa alle esercitazioni

Vedi fascicolo separato.

10.4 Grammatiche e analisi sintattica 20%

1. È data la seguente grammatica:

$$S \to CBA \qquad \mathcal{G} =$$

$$S \to ABC \qquad \mathcal{G} =$$

$$A \to aA \qquad \mathcal{G} =$$

$$A \to c \qquad \mathcal{G} =$$

$$B \to BS \qquad \mathcal{G} =$$

$$B \to b \qquad \mathcal{G} =$$

$$C \to AS \qquad \mathcal{G} =$$

$$C \to \varepsilon \qquad \mathcal{G} =$$

$$C \to B \qquad \mathcal{G} =$$

   Calcolarne gli insiemi guida (scrivere a lato).

   Soluzione

$S \to CBA$	$\mathcal{G} = a, b, c$
$S \to ABC$	$\mathcal{G} = a, c$
$A \to aA$	$\mathcal{G} = a$
$A \to c$	$\mathcal{G} = c$
$B \to BS$	$\mathcal{G} = b$
$B \to b$	$\mathcal{G} = b$
$C \to AS$	$\mathcal{G} = a, c$
$C \to \varepsilon$	$\mathcal{G} = a, b, c, \dashv$
$C \to B$	$\mathcal{G} = b$

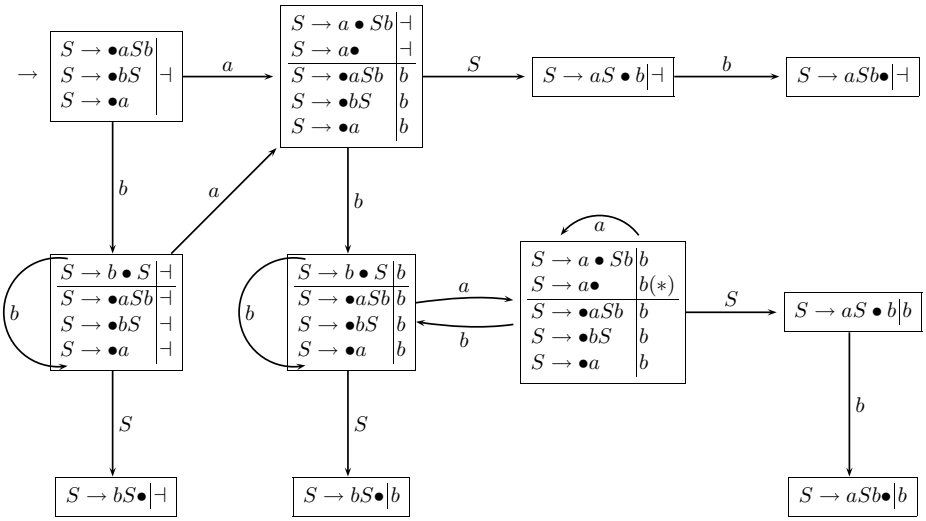
G è ricorsiva a sin., quindi non è LL(k).

2. È data la seguente grammatica:

$$S \to aSb \qquad S \to bS \qquad S \to a$$

Costruire il riconoscitore dei prefissi ascendenti LR(1) e stabilire in quali stati la gramamtica è LR(0), LALR, LR(1).

Soluzione



Conflitto LR(1) nello stato (*). C'è una candidata di riduzione con prospezione b ma lo stato presenta anche un arco uscente con etichetta b: conflitto riduzione-spostamento.

10.5 Traduzione e semantica 20%

1. Data la traduzione seguente, dove $u \in \{a, b\}^*$:

$$\tau(u) = u^R \qquad \text{se } |u| \text{ è pari}$$

$$\tau(u) = u \qquad \text{se } |u| \text{ è dispari}$$

   a) Scrivere lo schema di traduzione puramente sintattico, ossia la grammatica di traduzione, che realizza la traduzione.
   b) Esiste un trasduttore a pila deterministico che realizza la trasduzione τ? (motivare la risposta)
   c) Definire la traduzione inversa di τ, sempre attraverso uno schema di traduzione sintattico.

Soluzione

a) Schema di traduzione puramente sintattico:

Sorgente	Pozzo
$S \to P$	$S \to P$
$S \to D$	$S \to D$
$P \to aP_1$	$P \to P_1a$
$P_1 \to aP$	$P_1 \to Pa$
$P \to bP_1$	$P \to P_1b$
$P_1 \to bP$	$P_1 \to Pb$
$P \to \varepsilon$	$P \to \varepsilon$
$D \to aD_1$	$D \to aD_1$
$D_1 \to aD$	$D_1 \to aD$
$D \to bD_1$	$D \to bD_1$
$D_1 \to bD$	$D_1 \to bD$
$D \to a$	$D \to a$
$D \to b$	$D \to b$

b) Non esiste un trasduttore a pila deterministico che realizza la trasduzione. Infatti l'automa soltanto alla fine della lettura di u può sapere se deve emettere u stessa o la riflessa; ma tale momento è troppo tardi.
c) La traduzione inversa coincide con quella diretta!

2. Considerate un quesito o *query* in un ling. simile a SQL, esemplificato da:

$$\mathsf{select}\ '*'\ \mathsf{where}\ (a_2 = 3)\ \mathsf{from}\ \underbrace{(1,3,5)(2,2,5)(2,3,2)(8,9,2)}_{\text{relazione contenente 4 tuple}}$$

Il comando seleziona le tuple che soddisfano il predicato $a_2 = 3$, ossia che hanno il valore 3 nel 2ndo campo. Il risultato è la relazione:

$$\text{ris of } S = \{(1,3,5)(2,3,2)\}$$

La sintassi del ling. è data:
$S \rightarrow \mathsf{select}\ '*'\ \mathsf{where}\ (\ \mathsf{name}\ =\ \mathsf{value}\ )\ \mathsf{from}\ R$
$R \rightarrow (T)R$
$R \rightarrow (T)$
$T \rightarrow \mathsf{value}\ , T$
$T \rightarrow \mathsf{value}$

a) Completare il progetto della gramm. ad attributi, che assegna all'attributo ris of S il risultato di un quesito. Per ipotesi tutte le tuple della relazione hanno lo stesso grado. Gli attributi sono così specificati:

ris of S	risultato del quesito: un insieme di tuple;
sel of R	risultato del quesito sulla parte della relazione avente radice R
ques of R	il quesito è un record con 2 info.: ordinale dell'attributo su cui si fa la selezione, valore di esso; nell'es. $\mathsf{record}(2,3)$
ord of name	numero ordinale dell'attributo presente nel predicato: nell'es. vale 2;
num of value	valore presente nel predicato; nell'es. vale 3
tupla of T	vettore contenente gli n interi della tupla; ad es.: $\langle 1,3,5\rangle$

Gramm. da completare, specificando in pseudocodice le funzioni semantiche necessarie:

$S \to \mathsf{select}\ '*'\ \mathsf{where}\ (\ \mathsf{name}\ =\ \mathsf{value}\ )\ \mathsf{from}\ R$

ris of $S \leftarrow$ sel of R
ques of $R \leftarrow$ record(ord of name, num of value)

$R_0 \to (T) R_2$

ques of $R_2 \leftarrow \ldots$

sel of $R_0 \leftarrow \ldots$

$R \to (T)$

sel of $R \leftarrow \ldots$

$T_0 \to\ \mathsf{value}\ , T_1$

tupla of $T_0 \leftarrow \ldots$

$T \to\ \mathsf{value}$

tupla of $T \leftarrow \langle$num of $\mathsf{value}\rangle$

b) Esaminare se la condizione L è soddisfatta
c) Scrivere almeno una procedura semantica
d) (Facoltativo) Estendere il progetto della sintassi e della semantica in modo di poter scegliere su quale relazione del data-base si deve fare la selezione. Il data-base sarà fatto da più relazioni identificate dal loro nome. La clausola from conterrà anche il nome della relazione su cui operare.

Soluzione

a) Completare il progetto della gramm. ad attributi
ques of R è ereditato; tutti gli altri attributi sono sintetizzati.
Grammatica ad attributi:

$S \rightarrow$ select $'*'$ where (name = value) from R

ris of $S \leftarrow$ sel of R
ques of $R \leftarrow$ record(ord of name, num of value)

$R_0 \rightarrow (T) R_2$

ques of $R_2 \leftarrow$ ques of R_0

sel of R_0 $\leftarrow$ if (tupla of T[ques of R_0.ord] == ques of R_0.num) then tupla of T $\cup$
sel of R_2 else sel of R_2

(notazione C simile, supponendo che tupla of T sia un vettore di interi e ques of R una struct con
campi ord e num, di tipo intero)

$R \rightarrow (T)$

sel of $R \leftarrow$ if (tupla of T[ques of R_0.ord] == ques of R.num) then tupla of T else $\emptyset$

$T_0 \rightarrow$ value $, T_1$

tupla of $T_0 \leftarrow$ cat(num of value, tupla of T_1)

$T \rightarrow$ value

tupla of $T \leftarrow \langle$num of value$\rangle$

b) La condizione L è soddisfatta (verifica tu regola per regola)
c) Piuttosto ovvio, per es. per la regola $R \to (T)R$; prova tu a scrivere la procedura
d) Ritoccare la sintassi in modo opportuno: mettere l'identificatore della relazione nella clausola select e dotare di identificatore anche la relazione. Aggiungere un attributo ID of R, e aggiungere a ques anche l'identificatore della relazione. Poi ritoccare le regole semantiche.