Linguaggi Formali e Compilatori
Prof. Crespi Reghizzi
Soluzione della Prova scritta[1]
29/06/2004

	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			

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:

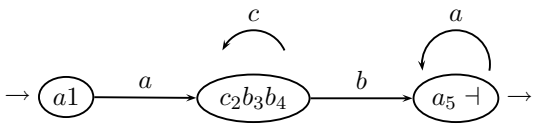
   $$mischia(L_X, L_Y) = d^*a(b \mid d)^*cb^* \mid d^*cab^*$$

[1] 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. È consentito scrivere a matita. Scrivere il proprio nome sugli eventuali fogli aggiuntivi.

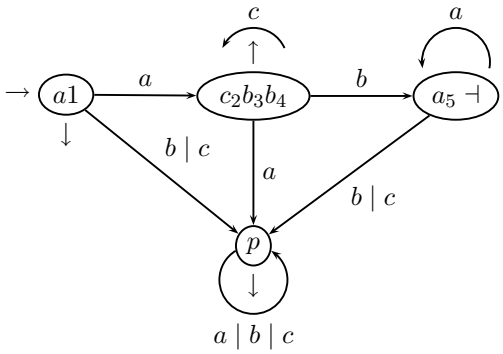
2. Costruire e poi minimizzare l'automa finito che riconosce il linguaggio $L = \neg(ac^*(b \mid ba^+))$. Riportare tutti i passaggi.

Soluzione

Costruzione automa deterministico:



Completamento e negazione:



L'automa è minimo.

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.

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 : :

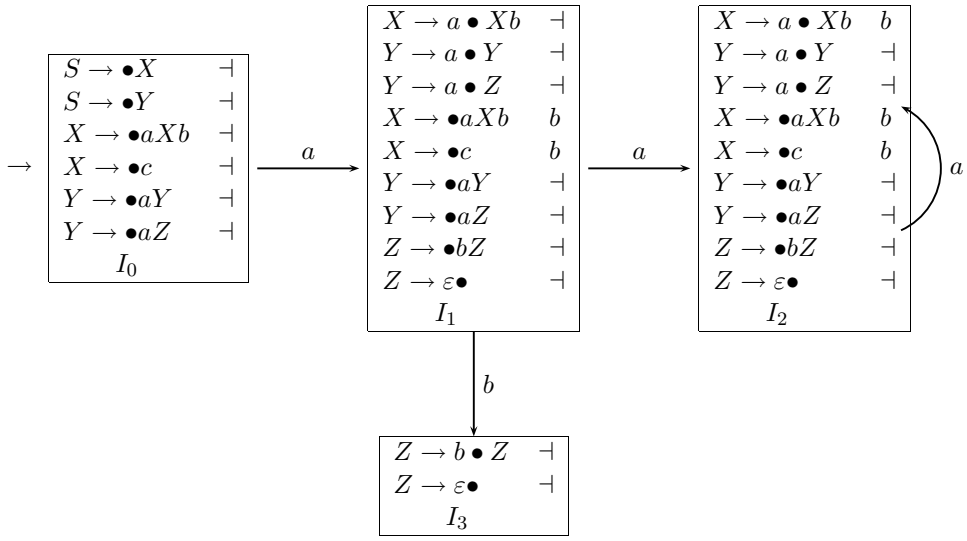
$S \to X$	$S \to Y$
$X \to aXb$	$X \to c$
$Y \to aY$	$Y \to aZ$
$Z \to bZ$	$Z \to \varepsilon$

   (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:



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:

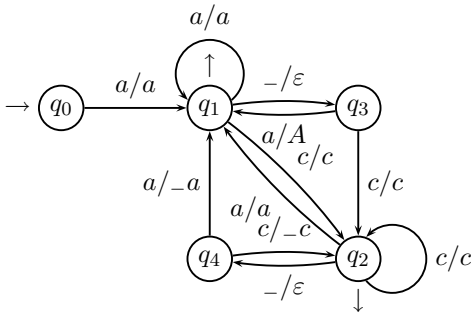
   (a) Con uno schema di traduzione sintattica (senza attributi)

   (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:
   $S' \to S$
   $S \to write(V)S \quad S \to read(V)S \quad S \to \varepsilon$
   $V \to var,\ V \qquad V \to var$
   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.