Linguaggi Formali e Compilatori

Prof. L. Breveglieri e S. Crespi Reghizzi

Prova scritta 09/09/2005 - Parte I: Teoria

COGNOME E NOME:.....

MATRICOLA:.....FIRMA:.....

ISTRUZIONI:

- L'esame si compone di due parti:
  - I (80%) Teoria:
    1. espressioni regolari e automi finiti
    2. grammatiche e automi a pila
    3. analisi sintattica
    4. traduzione e semantica
  - II (20%) Esercitazioni Flex e Bison
- Per superare l'esame l'allievo deve superare entrambe le parti (I e II) nello stesso appello oppure in appelli diversi della stessa sessione d'esame.
- Per superare la parte I (teoria) occorre dimostrare sufficiente conoscenza di tutte le quattro sottoparti (1-4).
- Tempo: Parte I (esercitazioni): 30 min - Parte II (teoria): 2.30 ore.
- È permesso consultare libri e appunti personali.

1 Espressioni regolari e automi finiti 20%

1. Data l'espressione regolare di alfabeto $\Sigma = \{a, b, c, d\}$ seguente:

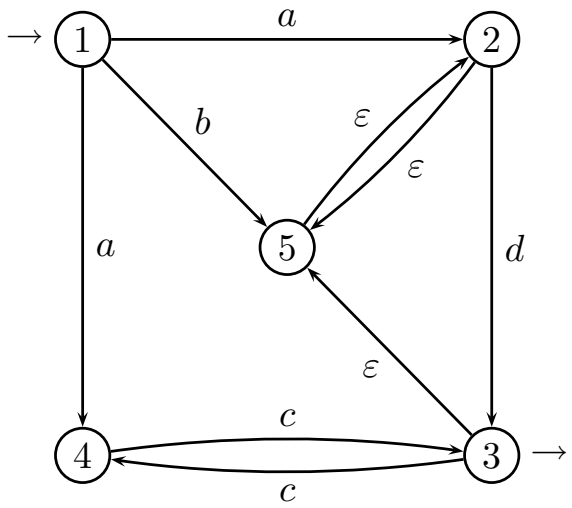
$$R = a\left(a^*bb \mid ab^*c\right)^*\left(d \mid a\right)$$

   (a) Si dica se l'espressione R sia ambigua oppure no, motivando sinteticamente la risposta.

   (b) Si trovi l'automa deterministico (non necessariamente in forma minima) che riconosce il linguaggio $L\left(R\right)$.

   (c) (facoltativo) Si ricavi la forma minima dell'automa deterministico che riconosce il linguaggio $L\left(R\right)$.

2. È dato l'automa M a stati finiti seguente:



(a) Si metta l'automa M in forma deterministica (non necessariamente minima).

(b) Si ricavi un'espressione regolare (non necessariamente non ambigua) equivalente all'automa M.

(c) (facoltativo) Si dica se l'automa deterministico sia minimo oppure no (non si chiede di trovarne la forma minima), motivando sinteticamente la risposta.

2 Grammatiche 20%

1. Si progetti una grammatica in forma non estesa, non ambigua, per il linguaggio delle espressioni aritmetiche polacche prefisse sotto specificato:

   - alfabeto $\Sigma = \{+, \times, a\}$
   - gli operatori di addizione e moltiplicazione hanno sempre due argomenti
   - vi è un vincolo sul numero di moltiplicazioni presenti nell'espressione: non vi può essere più d'una moltiplicazione, a meno che l'espr. contenga soltanto moltiplicazioni, nel qual caso non vi sono limiti al loro numero

   Esempi: $++aaa$ $\times\times aa \times aa$ $+\times a + aaa$

   Controesempi: $+ \times aa \times aa$

   (a) Si scriva la grammatica accertandosi che non sia ambigua.

   (b) Si disegni l'albero sintattico della frase: $+ \times a + aaa$

2. Si progetti la grammatica EBNF non ambigua di un frammento di linguaggio Java, così specificato. Sono presenti i concetti seguenti:

   - costante intera
   - variabile semplice (schematizzata dal carattere terminale i) e array con uno o più indici (racchiusi tra parentesi quadre)
   - operatore aritmetico $+$, $-$ (anche unario) e $*$, con la precedenza consueta della moltiplicazione su addizione e sottrazione; non vi sono sottoespressioni parentetizzate
   - invocazione di metodo con zero o più argomenti (racchiusi tra parentesi tonde); anche gli identificatori dei metodi e delle classi sono schematizzati con i
   - assegnamento
   - ogni istruzione è seguita da punto e virgola

   Esempio:

   $i = i + i\,[3, i] * i.i\,(i, i\,[i])\,; \quad i = i\,(i) +;$

   (a) Si scriva la grammatica nella forma BNF estesa.

   (b) Si disegni l'albero sintattico della frase: $i = i + i\,[3, i] * i.i\,(i, i\,[i])\,;$

3 Grammatiche e analisi sintattica 20%

1. Data la gramm. G seguente:

	Insieme guida
$S \to ASc$	
$S \to d$	
$A \to Ab$	
$A \to aAb$	
$A \to \varepsilon$	

   (a) Si calcolino gli insiemi guida $LL(1)$ delle produzioni di G.

   (b) Si spieghino i motivi del non essere $LL(1)$ di G e si scriva una grammatica $LL(1)$ equivalente a quella data. Si calcolino gli insiemi guida delle regole di tale nuova grammatica.

   (c) (facoltativo) Si verifichi se la grammatica G originale sia $LL(k)$ per $k > 1$.

2. Il linguaggio L seguente, di alfabeto $\{a, b, c\}$:

$$L = a^+b^+c \cup \{a^n b^n \mid \ n \geq 1\}$$

è definito dalla grammatica:

$$S \to Xc \mid Y$$

$$X \to Xb \mid Zb$$

$$Z \to Za \mid a$$

$$Y \to aYb \mid ab$$

(a) Si stabilisca se la grammatica sia $LR(0)$, $LALR(1)$ o $LR(1)$.

(b) Se la gramm. non fosse $LR(1)$, si individui il motivo del conflitto e si scriva una gramm. equivalente di tipo $LR(1)$.

4 Traduzione e semantica 20%

1. Si consideri il linguaggio delle espressioni, formate da variabili a, b, c, ..., z (lettere minuscole), dall'operatore binario + infisso, e dai due operatori funzionali prefissi $\max 2\,(e_1, e_2)$ e $\max 3\,(e_1, e_2, e_3)$ binario e ternario, rispettivamente (i simboli e_i schematizzano variabili o sotto-espressioni); non si usano parentesi per isolare sotto-espressioni.

   (a) Si scriva la schema di traduzione sintattico che traduce le espressioni descritte sopra in espressioni dove l'operatore + è di tipo polacco postfisso e l'operatore max è <u>solo</u> di tipo binario ma anch'esso di tipo polacco postfisso.
   Esempi:

sorgente	*pozzo*
$\max 3\,(a, b, c)$	$a\ b \max c \max$
$\max 2\,(a, b) + \max 3\,(c, d, e)$	$a\ b \max c\ d \max e \max +$
$a + b + \max 3\,(c, d, e)$	$a\ b + c\ d \max e \max +$

   (b) Dire poi se si possa mettere lo schema in forma deterministica $LL\,(1)$, motivando la risposta.

2. Si vuole tradurre in codice macchina un'istruzione d'assegnamento, data in forma polacca prefissa; un es. è la stringa sorgente seguente:

$$:= a_1 + a_2 * a_3 \; a_4$$

il cui significato è: $a_1 \leftarrow a_2 + a_3 * a_4$. La sintassi sorgente è la seguente:

$S \rightarrow := \; a \; E$
$E \rightarrow + \; E \; E$
$E \rightarrow * \; E \; E$
$E \rightarrow a$

dove a è il nome di una variabile.

Il codice macchina ha tre sole istruzioni:

istruzione	significato
move $a_1 \; a_2$	$a_2 \leftarrow a_1$
add $a_1 \; a_2 \; a_3$	$a_3 \leftarrow a_1 + a_2$
mult $a_1 \; a_2 \; a_3$	$a_3 \leftarrow a_1 * a_2$

Esempi:

sorgente	pozzo
$:= \; a_1 + a_2 * a_3 \; a_4$	*mult* $a_3 \; a_4 \; i_1$
	add $a_2 \; i_1 \; a_1$
$:= \; a_5 \; a_6$	*move* $a_5 \; a_6$

dove i_1 è una cella di memoria il cui nome deve essere creato dalle regole semantiche.

(a) Si progettino le funzioni semantiche di una gramm. ad attributi per calcolare la traduzione.

Si suggerisce l'uso degli attributi seguenti:

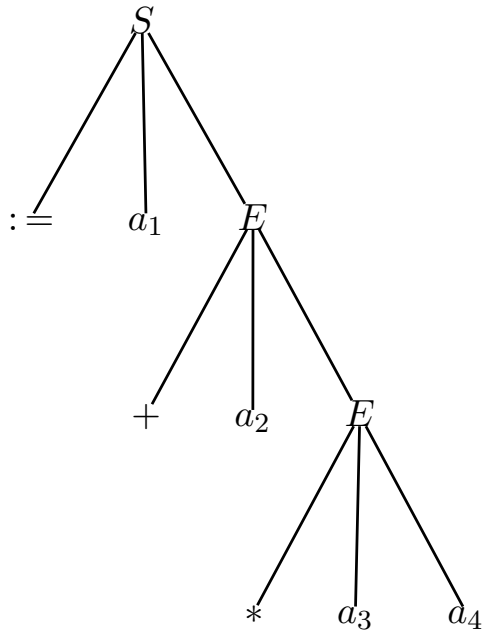
il nome delle variabili: n of a è un attributo lessicale
il nome delle celle di memoria: i of E
il codice prodotto: t of E, t of S

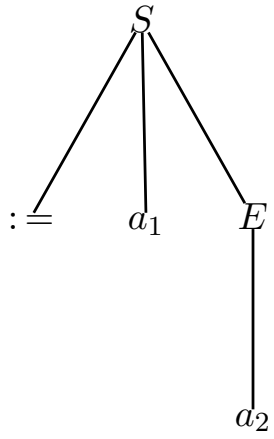
(b) Si decorino i due alberi con i valori degli attributi e le frecce delle dipendenze funzionali.

(c) Si verifichi se la valutazione degli attributi possa essere svolta con il metodo L e si scriva, almeno in parte, lo pseudocodice del

valutatore semantico, supponendo che l'albero sintattico sia stato già costruito dal parsificatore.

Sintassi	Funzioni semantiche
$S \rightarrow := \ a \ E$	
$E \rightarrow + \ E \ E$	
$E \rightarrow * \ E \ E$	
$E \rightarrow a$	


S
:=
a1
E
+
a2
E
*
a3
a4


S
:=
a1
E
a2