Formal Languages and Compilers (Linguaggi Formali e Compilatori)

prof. L. Breveglieri
(prof. S. Crespi Reghizzi, L. Sbattella)

Exam - 7 march 2007 - Part I: Theory

NAME:

SURNAME:

ID: SIGNATURE:

INSTRUCTIONS - PLEASE READ CAREFULLY:

- The exam consists of two parts:
  - I (80%) Theory:
    1. regular expressions and finite state automata
    2. context-free grammars and pushdown automata
    3. syntax analysis and parsers
    4. transduction and semantic analysis
  - II (20%) Practice on Flex and Bison
- To pass the complete exam the candidate is required to pass both parts (I and II), in whatever order in a single call or in different calls, but to conclude in one year.
- To pass part I (theory) the candidate is required to demonstrate a sufficient knowledge of the topics of each one the four sections (1-4).
- The exam is open-book: textbooks and personal notes are permitted.
- Please write clearly in the free space left on the sheets; do not attach or replace sheets.
- Time: Part I (theory): 2h.30m - Part II (practice): 45m

1 Regular expressions and finite automata 20%

1. The two following regular expressions are given:

$$R_1 = ((a\, b)^* \; c)^* \qquad\qquad R_2 = (c^* \; (a\, b)^*)^*$$

   Answer the following questions:

   (a) Check intuitively whether the two expressions R_1 e R_2 are equivalent (if they are not show a string belonging to one of the two regular expressions but not to the other one).

   (b) Check in an algorithmic way whether the two expressions R_1 e R_2 are equivalent.

---

2. The language L over the alphabet $\{a, b\}$ is defined by the following conditions:

   (a) the phrases start with the string $a\,b$

   and

   (b) the phrases end with the character b

   and

   (c) the phrases do not contain the factor $a\,b\,b$

   Answer the following questions:

   (a) Design in a systematic way a deterministic automaton that recognizes L.

   (b) If necessary, minimize the previously designed recognizer.

---

2 Context-free grammars and pushdown automata 20%

1. Consider the language L_1 over the alphabet $\Sigma = \{a, b, c\}$:

$$L_1 = (\, x\, c\,)^* \quad \text{where } x \in \Sigma^* \text{ and } |x|_a = |x|_b \geq 0 \text{ and } |x|_c = 0$$

which is exemplified by the following strings:

$$a\,b\,b\,a\,c\,b\,a\,c\,a\,a\,a\,b\,b\,b\,c \qquad\qquad a\,b\,a\,b\,b\,a\,c$$

and consider the regular language R_2 (again over the alphabet Σ):

$$R_2 = \big(\, a^+\, b^+\, c\, \big)^*$$

Now, consider the language L_3 defined by the following intersection:

$$L_3 = L_1 \cap R_2$$

Answer the following questions:

(a) Write all the phrases belonging to L_3 with length less than or equal to 6.

(b) Design a grammar G, not ambiguous and not in extended form, that generates the language L_3, and draw the syntax tree of a phrase of length exactly 6.

---

2. Consider the language L of the programs consisting of (at least) one or more assignment statements, separated and terminated by semicolon ';', with expressions containing only addition and without parentheses. Here follows a sample program:

$$\begin{aligned} i &= i+i+i+i; \\ i &= i; \\ i &= i+i+i+i+i; \end{aligned}$$

With reference to this language L, add the possibility for the program to contain one (and no more than one) syntax error. There are two error types to be considered:

omission of the assignment symbol $=$
omission of the addition symbol $+$

Each of the two following sample programs contains exactly one such syntax error (of either type):

$$\begin{aligned} i &= i+i+i+i; \\ i &= i; \\ i &\quad i+i+i+i+i+i; \end{aligned}$$

$$i = i\ i+i+i;$$

Now, define the language L_F to be the set of all the correct programs and of those that contain exactly one error of either type (but not of both types):

$$L_F = L \cup \text{ programs that contain exactly one error}$$

The following sample program *does not* belong to the language L_F, as it contains more than one error:

$$\begin{aligned} i &= i+i+i\ i; \\ i &= i; \\ i &\quad i+i+i+i+i+i; \end{aligned}$$

Answer the following questions:

(a) Write an EBNF (extended) grammar, not ambiguous, that generates the language L.
(b) Write an EBNF (extended) grammar, not ambiguous, that generates the language L_F.
(c) Explain why the written grammar of L_F ensures that the generated string contains at most one syntax error.

3 Syntax analysis and parsers 20%

1. One wishes one modified the given grammar G so as to obtain an equivalent grammar G' suited to recursive descent deterministic syntax analysis (that is suited to $LL(k)$ analysis, for some $k \geq 1$).

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

Answer the following questions:

   (a) Design the requested grammar G' and explain why it is equivalent to the original grammar G.

   (b) Determine the lookahead sets of the grammar G' and check whether it is $LL(1)$ or $LL(k)$ for some $k > 1$.

---

2. The following grammar G is given:

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

Answer the following questions:

(a) Draw the $LR(1)$ driver graph (the bottom-up recognizer) of G.

(b) Check whether the drawn driver graph satisfies the conditions $LR(1)$, $LALR(1)$ or $LR(0)$, and explain why.

---

4 Transduction and semantic analysis 20%

1. A source language consists of arithmetic expressions with the usual infix addition operator and parentheses. Moreover, on the left of each parenthesized group one finds a marker field that specifies whether the translation of the group is prefix of postfix (see the example below). The source language is generated by the following (source) grammar:

$$G \begin{cases} S \rightarrow (\text{‘prefix’} \mid \text{‘postfix’})\ \text{‘(’}\ E\ \text{‘)’} \\ E \rightarrow T\ (\text{‘+’}\ T)^{*} \\ T \rightarrow a \mid (\text{‘prefix’} \mid \text{‘postfix’})\ \text{‘(’}\ E\ \text{‘)’} \end{cases}$$

A sample translation is the following (for better clarity here the variable names are differentiated and the operators are numbered, but in the source string the variables are all called a and the operators do not have any numbering):

source:	postfix ($a\ +_1\ b\ +_2$ prefix ($c\ +_3$ prefix (d)))
destination:	$a\ b\ +_1\ +_3\ c\ d\ +_2$

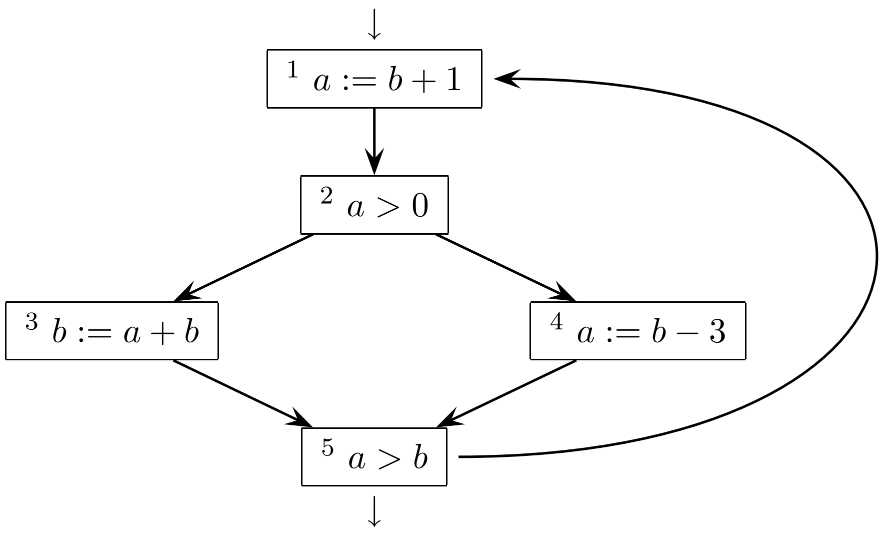
Notice that the marker field is *not* preserved in the destination string.

Answer the following questions:

(a) Modify if necessary the source grammar and design a syntax transduction scheme (or grammar), without semantic attributes, that translates the source language and outputs locally the prefix or postfix form, in agreement with the prescription given by the marker field.

(b) Discuss whether the designed transduction scheme is deterministic.

2. Consider the following control flow graph of a program:



At the end of the program, no variable is live.

Answer the following questions:

(a) Write the flow equations to compute the liveness intervals of the variables.

(b) Compute and write in the following table the sets of live variables at every point of the program (see the table prepared in the next page - the number of columns is not significant).

---

			step 1		**step** 2		**step** 3		**step** 4		**step** 5		**step** 6	
	state 0		**state** 1		**state** 2		**state** 3		**state** 4		**state** 5		**state** 6	
#	*in*	*out*	*in*	*out*	*in*	*out*	*in*	*out*	*in*	*out*	*in*	*out*	*in*	*out*
1	$\emptyset$	$\emptyset$												
2	$\emptyset$	$\emptyset$												
3	$\emptyset$	$\emptyset$												
4	$\emptyset$	$\emptyset$												
5	$\emptyset$	$\emptyset$												