Formal Languages and Compilers
(Linguaggi Formali e Compilatori)

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

Written exam - 5 july 2007 - Part I: Theory

NAME:

SURNAME:

ID: SIGNATURE:

INSTRUCTIONS - READ CAREFULLY:

- The exam consists of two parts:
  - I (80%) Theory:
    1. regular expressions and finite automata
    2. free grammars and pushdown automata
    3. syntax analysis and parsing
    4. translation and semantic analysis
  - II (20%) Practice on Flex and Bison
- To pass the exam, the candidate must succeed in both parts (I and II), in one call or more calls separately, but within one year.
- To pass part I (theory) one must be sufficient in all the four sections (1-4).
- The exam is open book (texts and personal notes are admitted).
- Please write in the free space left and if necessary continue on the back side of the sheet; do not attach new sheets nor replace the existing ones.
- Time: Part I (theory): 2h.30m - Part II (practice): 45m

1 Regular Expressions and Finite Automata 20%

1. The language L over the alphabet $\{a, b, c\}$ is defined by means of the following three *conditions* (every string of L must satisfy all of them):

   (a) a phrase of L starts with the letter a

   and

   (b) a phrase of L ends with the letter c

   and

   (c) a phrase of L may not contain any of the following four two-letter factors:

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

   For example, the following string:

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

   is a phrase of L, while the string:

   $$a\ \underbrace{b\,a}_{\text{forbidden}}\ b\,c$$

   is invalid (and is not a phrase of L), because the highligthed factor is not admitted.

   Do the following points:

   (a) Write a regular expression of the language L by using only the operators union, concatenation and star (or cross).

   (b) Define the language $\overline{L}$, the complement of L, by modifying suitably the three conditions (a, b, c) above.

---

2. The following regular expression R is given (in generalised form, as it contains the intersection operator), over the alphabet $\{a, b, c\}$:

$$R = a\ (\ b\ c^*\ )^* \quad \cap \quad (\ a\ \mid\ b\ )\ \left(\ a^*\ b^*\ c^+\ \right)^*$$

Do the following points:

(a) Design in a systematic way a recognizer automaton equivalent to R (that is, design it as the intersection of automata).

(b) If necessary, put the designed recognizer in deterministic form.

---

2 Free Grammars and Pushdown Automata 20%

1. A list x contains any number of elements e, separated by the character g; this means that x is modelled as $(e^+\, g)^*\, e^+$. For example:

$$x = \underbrace{e\ e\ e\ e}_{\text{group } G_1}\ g\ \underbrace{e}_{\text{group } G_2}\ g\ \underbrace{e\ e}_{\text{group } G_3}\ g\ \underbrace{e\ e\ e}_{\text{group } G_4}\ g\ \underbrace{e\ e\ e}_{\text{group } G_5}$$

   Name "group G_i" ($i \geq 1$) the i^{th} sublist of elements e included between two consecutive g characters. A group may not be empty (see above). As the sample string x shows, the prefix G_1 and the suffix G_5 are considered groups as well.

   Such a list is a phrase of L if and only of it contains two groups G_i and G_j ($1 \leq i < j$), not necessarily consecutive ($j - i \geq 1$), such that the group G_i is shorter than or equal to G_j, that is the inequality $|G_i| \leq |G_j|$ holds.

   The sample string x is valid and is a phrase of L, because if one poses for instance $i = 2$ and $j = 4$ then the group G_2 is shorter than G_4, which suffices to validate x. On the other side, the following string y:

$$y = \underbrace{e\ e\ e}_{\text{group } G_1}\ g\ \underbrace{e\ e}_{\text{group } G_2}\ g\ \underbrace{e}_{\text{group } G_3}$$

   is invalid and is not a phrase of L.

   Do the following points:

   (a) Write a grammar G, not in extended form, that generates the language L, and draw the syntax tree of the sample phrase x shown above.

   (b) Examine whether the designed grammar G is ambiguous or not.

---

2. Consider a text document format in a style similar to LaTeX, slightly simplified. Such a format has the following structure:

   - The document is a sequence (possibly empty) of structures, each of which is a dotted or numbered list of text items (text_item).
   - The two types of structure are denoted as follows:

Dotted list:	Numbered list:
`%begin_itemize`	`%begin_enumerate`
`%item <text_item>`	`%item <text_item>`
`...`	`...`
`%end_itemize`	`%end_enumerate`

   - Dotted and numbered lists may be nested into one another (even if they are of different type), at an arbitrary nesting depth.
   - Each dotted or numbered list contains at least one text item (whether a simple text string or a nested substructure), or even two or more.
   - The symbol ⟨text_item⟩ denotes a simple text string, denoted by the pure terminal c (even if it consists of some characters), or a substructure, that is a dotted or numbered list.

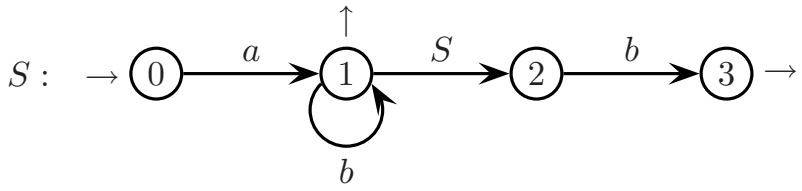
   Example:

```
%begin_itemize

%item c

%item %begin_enumerate

%item c

%item c

%end_enumerate

%item c

%end_itemize

%begin_enumerate

...
```

   Write a grammar in extended form (EBNF), not ambiguous, that generates the above described document format.

3 Syntax Analysis and Parsing 20%

1. The following grammar is given, presented as an automaton:



Do the following points:

(a) Compute the lookahead sets on all the arcs of the graph and say whether the grammar G is $LL(k)$ for some $k \geq 1$.

(b) If necessary, examine whether it is possible to find a grammar equivalent to G with the $LL(1)$ property.

---

2. There are given the following grammar G (with axiom S):

$$
\begin{aligned}
S &\rightarrow a\ B\ S\ b \\
S &\rightarrow a\ B \\
B &\rightarrow b\ B \\
B &\rightarrow \varepsilon
\end{aligned}
$$

and the following sample string, which is a phrase of $L(G)$:

$$a\ b\ a\ b$$

Do the following points:

(a) Simulate the recognition process of the sample string by means of the Earley algorithm; please fill the table prepared on the next page.

(b) Draw the syntax tree of the recognised sample string.

Draft table for the simulation of the Earley algorithm								
state 0	pos. a	state 1	pos. b	state 2	pos. a	state 3	pos. b	state 4

4 Translation and Semantic Analysis 20%

1. A text in natural language is given, consisting of words separated by comma "," or blank ⟨blank⟩, or even by both. The word is roughly modelled as a string of identical characters c, without more precision. The text may contain parentheses "(" and ")" including a subtext. Parentheses may not be nested. The following (source) grammar G_s defines such a text format (with axiom S):

$$
\begin{aligned}
S &\rightarrow (\ W \mid P\ )\ (\ \text{“,”} \mid \text{“⟨blank⟩”} \mid \text{“, ⟨blank⟩”}\ )\ S \\
S &\rightarrow (\ W \mid P\ ) \\
W &\rightarrow c^{+} \\
P &\rightarrow \text{“(”}\ W\ (\ (\ \text{“,”} \mid \text{“⟨blank⟩”} \mid \text{“, ⟨blank⟩”}\ )\ W\ )^{*}\ \text{“)”}
\end{aligned}
$$

   Do the following points:

   (a) Design a syntax transduction scheme (that is, a transduction grammar), by modifying suitably the source grammar G_s, that generates a translated text which is identical to the source one, with the exception that the words included in the parentheses are separated exactly by one blank, in whatever way they appear to be separated in the source text. Here is an example:

   source string:
   ccc ⟨blank⟩ (*cc*, *cc*, ⟨blank⟩ *cccc* ⟨blank⟩ *cc*) *cc*, ⟨blank⟩ *ccc*

   translation:
   ccc ⟨blank⟩ (*cc* ⟨blank⟩ *cc* ⟨blank⟩ *cccc* ⟨blank⟩ *cc*) *cc*, ⟨blank⟩ *ccc*

   (b) Design a transducer automaton that computes the above described transduction (spend a short time and think about what transducer model should be used).

---

2. A (simplified) text in natural language, consisting of words separated by one blank $\langle$blank$\rangle$, has to be centred in a rectangular terminal window of width equal to $W \geq 1$ columns, so that each row is centred in the W columns. The word is modelled roughly as a string of characters c, without more precision.

   Take as a reference the attribute grammar presented in the course textbook at pp. 300, which alignes the text on the left (but does not centre the rows); its syntax support is shown below (with axiom S):

$$
\begin{aligned}
S &\rightarrow T \\
T &\rightarrow T \ \langle\text{blank}\rangle \ T \\
T &\rightarrow V \\
V &\rightarrow c \ V \\
V &\rightarrow c
\end{aligned}
$$

   One wishes one modified such an attribute grammar by adding new semantic attributes, if necessary, and / or by modifying the already known ones, in order to compute the column of the end character of the rightmost word of each centred row. Here follows an example (the same as in the textbook, but with centring rather than left alignment): the phrase "la torta ha gusto ma la grappa ha calore" is centred in a terminal window of width $W = 13$ columns:

1	2	3	4	5	6	7	8	9	10	11	12	13
	l	a		t	o	r	t	a		h	a	
	g	u	s	t	o		m	a		l	a	
		g	r	a	p	p	a		h	a		
			c	a	l	o	r	e				

   The attribute *ultimo* (*last*) has value 3 for the word "la", 9 for "torta", 12 for "ha", ..., and 9 for "calore". Odd blanks, if any, are left on the right side of the rows.

   Do the following points:

   (a) Write the above described attribute grammar, and give its attributes and semantic rules (fill the tables prepared in the next pages).

   (b) Check whether the designed grammar is of type one-sweep or even L.

---

attributes to be used for the grammar

type	name	(non)terminals	domain	meaning
already given in the textbook and / or to be modified / added				

syntax	semantic functions
$S_0 \rightarrow T_1$	
$T_0 \rightarrow T_1 \langle \text{blank} \rangle T_2$	
$T_0 \rightarrow V_1$	
$V_0 \rightarrow c\, V_1$	
$V_0 \rightarrow c$	