Formal Languages and Compilers
(Linguaggi Formali e Compilatori)

prof. Luca Breveglieri
(prof. S. Crespi Reghizzi, prof. A. Morzenti)

Written exam - 16 september 2010 - Part I: Theory

NAME:

SURNAME:

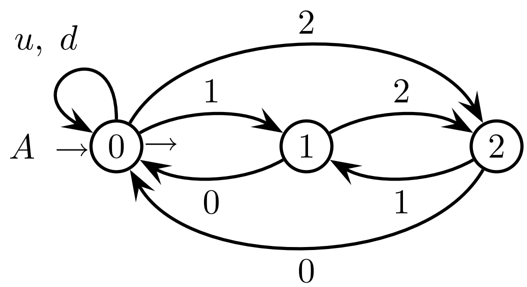
ID: SIGNATURE:

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 answer the mandatory (not optional) questions. Notice the full grade is achieved by answering the optional questions.
- 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. An elevator serves three floors, numbered 0, 1 and 2. The keys 0, 1 and 2 select the floor to go to, according to the state-transition graph of the automaton A below:



Before entering the cabin at floor 0, the passenger must insert a card with a personal id code that enables the access to the floors, namely: code *uno* gives access to the two floors 0 and 1, code *due* gives access to all the floors. After the cabin has gone down to the floor 0, only one passenger may enter the cabin, and so on.

Given the terminal alphabet $\Sigma = \{ d, u, 0, 1, 2 \}$, consider the language C (with $C \subseteq \Sigma^*$) of the sequences that are conforming to the access rights, as it is explained by the following examples and counterexamples.

Examples:

$$d\,1\,0\,d\,2\,0\,d\,1\,2\,0\,1\,0, \quad u\,1\,0\,d\,2\,1\,0, \quad u\,1\,0\,d\,2\,0, \quad u\,1\,0\,u\,d\,2\,0, \quad d\,d\,u\,1\,0$$

Counterexamples:

$$u\,2\,0, \qquad u\,1\,2\,0, \qquad u\,1\,0\,d\,u\,2\,0$$

Answer the following questions:

(a) Design the recognizer of the language C, as a *cartesian product* of two finite state automata, that is:
   - the automaton A drawn above
   - and another automaton M (deterministic or not), still to be designed

(b) (optional) Say if the language $L(A)$ is local and explain why.

2. Consider the following regular expression E:

$$E = (\ b^* \ (\ b \ \mid \ c\ )\ a\ b^+\ )^*$$

Answer the following questions:

(a) Show an ambiguous phrase belonging to the language defined by the regular expression E. Discuss if the ambiguity degree of E is bounded or unlimited.

(b) By means of a systematic technique, design a deterministic automaton A equivalent to the regular expression E.

---

2 Free Grammars and Pushdown Automata 20%

1. Consider the well known arithmetic expressions with the operations of addition '$+$', multiplication '$\times$', the round brackets open '(' and closed ')', and variables symbolized by the character a. As usual, multiplication takes precedence over addition.

   Answer the following questions:

   (a) Write a grammar G_1, not ambiguous and not in the extended form (BNF), of the expressions introduced above, with the additional constraint that they *contain only* an even number of additions (zero included).

   (b) (optional) Write a grammar G_2, not ambiguous and not in the extended form (BNF), of the expressions introduced above, with the additional constraint that they *contain only* an odd number of parenthesized (sub)expressions.

---

2. Consider a fragment of the language *Attempto Controlled English* (or ACE), which models a simplified part of the english grammar, illustrated below:

   (a) Simple Sentence SS:

   $$\overbrace{\overbrace{a\ customer}^{NP}\ \overbrace{inserts}^{VP}\ \overbrace{the\ card}^{NP}}^{SS}\ .$$

   where NP is a NounPhrase and VP is a VerbPhrase. The substantives like *customer* and *card* are denoted by the generic symbol N. They may be preceded by the indeterminative article *a* or determinative *the*, indifferently.

   (b) A NP may contain one or more adjectives, denoted by the generic symbol A:

   $$a\ \overbrace{rich}^{A}\ customer\ inserts\ a\ \overbrace{small}^{A}\ and\ \overbrace{green}^{A}\ card\ .$$

   If there are two adjectives, they have to be separated by the conjunction *and*; if there are three or more, they have to be separated by commas except the last one, which has to be preceded by the conjunction *and*, like *a small, thin and green card* .

   (c) There are two types of verb: transitive TV and intransitive IV

   $$a\ customer\ \overbrace{inserts}^{TV}\ the\ card\ . \qquad\qquad a\ customer\ \overbrace{waits}^{IV}\ .$$

   Yet the object of a transitive verb may be missing: *the customer calls* .

   (d) Two or more simple sentences SS may be coordinated by means of the conjunctions *or* and *and*, and also by means of the conjunctions *, or* and *, and* (please notice the comma !). The conjunction *and* takes precedence over the conjunction *or*, but the conjunctions with a comma give precedence to those without, as it is shown below:

   $$\underbrace{the\ bell\ rings}\ or\ \overbrace{\underbrace{the\ girl\ waits}\ and\ \underbrace{the\ boy\ opens\ the\ door}}\ .$$

   $$\overbrace{\underbrace{the\ bell\ rings}\ or\ \underbrace{the\ girl\ waits}}\ ,and\ \underbrace{the\ boy\ opens\ the\ door}\ .$$

   Yet in the phrase there may be only one conjunction with a comma.

   The terminal alphabet of the language is the following:

   $$\Sigma = \{\ a,\ the,\ N,\ A,\ IV,\ TV,\ and,\ or,\ \text{‘,’},\ \text{‘.’}\ \}$$

   Write a grammar, not ambiguous and in the extended form (EBNF), that generates the fragment of the language ACE illustrated above.

---

3 Syntax Analysis and Parsing 20%

1. Consider the following grammar G, over the alphabet $\{a, b, d, e\}$ (axiom S):

$$G \begin{cases} S \rightarrow a\ B \mid D \mid a\ b\ e \\ B \rightarrow b\ e\ D \\ D \rightarrow a\ b\ d\ S \end{cases}$$

Answer the following questions:

(a) Draw at least the machine of the axiomatic rules and show that the grammar G is neither $LL(1)$ nor $LL(2)$.

(b) Find the minimum k such that the grammar G is $LL(k)$.

(c) (optional) Consider the language generated by the grammar G; say if it has a grammar of type $LL(1)$ and explain why.

---

2. Consider the following grammar G_1, over the alphabet $\{a, b, c, d, e\}$ (axiom S):

$$G_1 \begin{cases} S \rightarrow B\ a\ S \mid e \\ B \rightarrow a\ B\ c \mid a\ B\ d \mid b \end{cases}$$

Answer the following questions:

(a) Consider the rules of the grammar G_1 and discuss if G_1 is $LR(1)$; do not build the driver graph, rather justify the answer adequately.

(b) In the grammar G_1 change the rules of the nonterminal B and consider the following grammar G_2:

$$G_2 \begin{cases} S \rightarrow B\ a\ S \mid e \\ B \rightarrow a\ B\ c \mid a\ B\ a \mid a \end{cases}$$

Discuss if G_2 is $LR(1)$ and justify adequately the answer.

(c) (optional) Now consider the following grammar G_3 (obtained from G_2 by modifying the rules of the axiom S) and discuss what changes as for the possibility of doing the $LR(1)$ analysis of the generated language; justify adequately the answer.

$$G_3 \begin{cases} S \rightarrow B\ f\ S \mid e \\ B \rightarrow a\ B\ c \mid a\ B\ a \mid a \end{cases}$$

---

4 Translation and Semantic Analysis 20%

1. Consider a language of one-level lists, delimited by brackets, not empty, with elements represented by the character 'e' and separated by commas ','. The translation of such lists is exemplified as follows:

source	translation
(e)	single e
(e, e)	former e latter e
(e, e, e)	first e next e last e
(e, e, e, e)	first e next e next e last e
...	...

where single, former, latter, first, next and last are the new delimiters and separators. Here is the grammar G that generates the source language (axiom S):

$$G \begin{cases} S \rightarrow \text{`('}\ L\ \text{`)'} \\ L \rightarrow e\ \text{`,'}\ L \mid e \end{cases}$$

Answer the following questions:

(a) Write a grammar (or a scheme) for the transduction described above, by suitably changing the source grammar G.

(b) (optional) Imagine you enlarge the source language to include, besides the previous ones, also the lists nested at any nesting depth, and extend the transduction to these new lists, as follows:

source	translation
((e))	SINGLE e
((e) , e)	former single e latter e
((e, e,) , (e))	former former e latter e latter single e
(e, e, (e, e))	first e next e last former e latter e
...	...

where SINGLE is a short form to compact the repetition single single, though losing some structural information.

Proceed similarly to translate lists of three or more levels, where single single single ... is shortened into SINGLE (instead the repetitions of former or first must not be not shortened).

Write a grammar (or a scheme) for the extended transduction, by suitably changing the source grammar G.

2. Consider the following fragment of the abstract grammar of a programming language (axiom statL):

$$
\begin{aligned}
\langle \mathsf{statL} \rangle &\rightarrow \langle \mathsf{stat} \rangle \ \langle \mathsf{statL} \rangle \\
\langle \mathsf{statL} \rangle &\rightarrow \langle \mathsf{stat} \rangle \\
\langle \mathsf{stat} \rangle &\rightarrow if \ \langle \mathsf{cond} \rangle \ then \ \langle \mathsf{statL} \rangle \ end \\
\langle \mathsf{stat} \rangle &\rightarrow if \ \langle \mathsf{cond} \rangle \ then \ \langle \mathsf{statL} \rangle \ else \ \langle \mathsf{statL} \rangle \ end \\
\langle \mathsf{stat} \rangle &\rightarrow asg \\
\langle \mathsf{cond} \rangle &\rightarrow c
\end{aligned}
$$

Answer the following questions:

(a) Write an attribute grammar that puts a label 'else' to each assignment statement that is reachable through at least one else branch in the program. To this purpose, use a boolean attribute *in_else* and the function *label_else* to label the assignments. Here is an example:

```
asg                     -> do not label
if c then
    asg                 -> do not label
    asg                 -> do not label
else
    if c then
        asg             -> label_else
    else
        asg             -> label_else
    end
end
```

Say if the grammar is of the one sweep type and explain why.

(b) Since statistically the assignment statements reachable through an else branch have a low execution probability, one wishes one evaluated in the root of the syntax tree of the program, the ratio between the number of assignments reachable via an else branch (or more such branches) and the overall number of assignments. This piece of information may help the compiler optimize the machine code generated. In the example above such a ratio is 2/5.

To this purpose, use the integer attributes *n_else* and *n_tot* to count the two assignment types, and the real attribute r for the ratio. Say if the grammar is of the one sweep type and in particular of type L, and explain why.

syntax	semantic functions (question (a))
1: $\langle\mathsf{statL}\rangle_0 \rightarrow \langle\mathsf{stat}\rangle_1\ \langle\mathsf{statL}\rangle_2$	
2: $\langle\mathsf{statL}\rangle_0 \rightarrow \langle\mathsf{stat}\rangle_1$	
3: $\langle\mathsf{stat}\rangle_0 \rightarrow if\ \langle\mathsf{cond}\rangle_1\ then\ \langle\mathsf{statL}\rangle_2\ end$	
4: $\langle\mathsf{stat}\rangle_0 \rightarrow\ if\ \langle\mathsf{cond}\rangle_1\ then\ \langle\mathsf{statL}\rangle_2$ $else\ \langle\mathsf{statL}\rangle_3\ end$	
5: $\langle\mathsf{stat}\rangle_0 \rightarrow asg$	
6: $\langle\mathsf{cond}\rangle_0 \rightarrow c$	

syntax	semantic functions (question (b))
1: $\langle \mathsf{statL} \rangle_0 \rightarrow \langle \mathsf{stat} \rangle_1 \ \langle \mathsf{statL} \rangle_2$	
2: $\langle \mathsf{statL} \rangle_0 \rightarrow \langle \mathsf{stat} \rangle_1$	
3: $\langle \mathsf{stat} \rangle_0 \rightarrow if \ \langle \mathsf{cond} \rangle_1 \ then \ \langle \mathsf{statL} \rangle_2 \ end$	
4: $\langle \mathsf{stat} \rangle_0 \rightarrow \ if \ \langle \mathsf{cond} \rangle_1 \ then \ \langle \mathsf{statL} \rangle_2$ $else \, \langle \mathsf{statL} \rangle_3 \ end$	
5: $\langle \mathsf{stat} \rangle_0 \rightarrow asg$	
6: $\langle \mathsf{cond} \rangle_0 \rightarrow c$	