Formal Languages and Compilers (Linguaggi Formali e Compilatori)

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

Written exam - 5 march 2008 - 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 three following regular expressions E_1, E_2 and E are given, extended with the complement and intersection operators, over alphabet $\Sigma = \{ a, +, - \}$:

$$\begin{aligned} E_1 &= a \; ((+ \mid -) \; a)^* \\ E_2 &= \neg \; (\; \Sigma^* \; - \; \Sigma^* \; + \; \Sigma^* \;) \\ E &= E_1 \; \cap \; E_2 \end{aligned}$$

Metasymbol "$\neg$" indicates set complement. Pay attention to that here character "$-$" is an element of alphabet Σ (not the set difference operator).

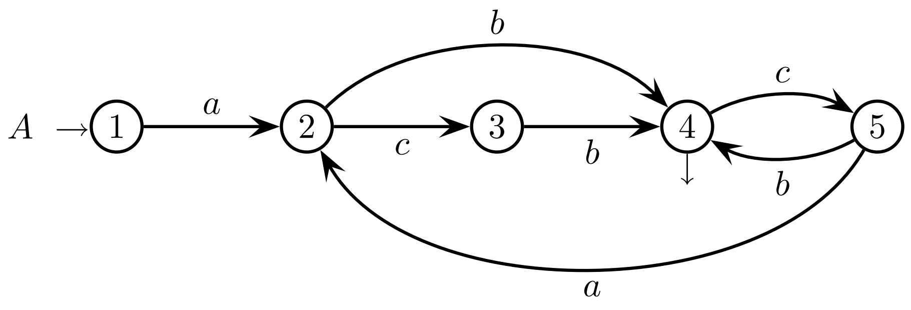
Answer the following questions:

(a) Write in the table below the strings of length equal to exactly 5 characters, which are generated by regular expressions E_1 and E (the number of rows in the table is not significant).

(b) Write a regular expressions equivalent to E but not in extended form (it must contain only concatenation union and star or cross operators). Choose freely the method for constructing the expression and explain briefly why it works.

#	E_1	E

2. The following finite state automaton A is given, over the input alphabet $\Sigma = \{a, b, c\}$:



Apply to automaton A the following alphabetic transformation h (homomorphism):

$$h(a) = a \qquad h(b) = b \qquad h(c) = \varepsilon$$

Answer the following questions:

(a) Draw the state-transition graph of the deterministic automaton A' that recognises language $h(L(A))$ (homomorphic image of language $L(A)$).

(b) Say whether the two following languages:

$$L(A) \qquad \text{e} \qquad h(L(A))$$

which are recognised by automata A and A', respectively, are of local type or not, and explain briefly why.

2 Free Grammars and Pushdown Automata 20%

1. Consider the Dyck language L_D with open and closed round and square brackets, that is over alphabet $\Sigma = \{ \text{‘('}, \text{‘)'}, \text{‘['}, \text{‘]'} \}$. Define language L to be a subset of language L_D with the following constraint: the strings of L do not contain any three or more consecutive open round brackets.

   Here are two invalid strings, not belonging to language L:

   $$[((( ) [ ]))] \qquad\qquad (((( ) [ ])))$$

   Answer the following questions:

   (a) Write a grammar G, not in extended form (BNF), that generates language L (no matter whether G is ambiguous).

   (b) Suppose to express language L as intersection of the Dyck language L_D and of a regular language generated by a regular expression E, as follows:

   $$L = L_D \cap L(E)$$

   It is required to write regular expression E.

2. Consider a language to model the data declaration section of a program, similar to the data declaration syntactic style of a generic programming language. This language must include the following syntactic features:

   - The data declaration section consists of a list of variable declarations, separated by ";" (semicolon). The last declaration has a ";", too.
   - There are two scalar variable types: integer and real. For instance:

     ```
     age : integer ;
     weight, height : real ;
     ```

   - There are variables of record type (or structure type). A record contains one or more fields, of any type: scalar, vector (array - see below) and record. For instance

     ```
     city : record
         position : record
             x, y : real ;
         end record ;
         population : integer ;
     end record ;
     ```

   - There are variables of array type, one or more dimensional. The array elements may be of scalar or record type, but not of array type. For instance:

     ```
     tabella : array [1 ... 10, 1 ... 100, -10 ... 10] of
     complex : record
         re, im : real ;
     end record ;
     ```

   Answer the following questions:

   (a) Write a grammar G, not ambiguous and in extended form (EBNF), that generates the language described above.

   (b) (optional) Represent the rules of grammar G in the form of syntactic diagrams.

3 Syntax Analysis and Parsing 20%

1. The following grammar G is given, in extended form (EBNF) (axiom S):

$$G \begin{cases} S \rightarrow A\, b \mid a\, [\, c\, S\,] \\ A \rightarrow a\, (\, b\, A\,)^{*}\, S \end{cases}$$

   The terminal alphabet of grammar G is $\{a, b, c\}$.

   Answer the following questions:

   (a) Represent grammar G in the form of a recursive network of finite state automata over the total alphabet (terminals and nonterminals).

   (b) Determine all the lookahead sets of grammar G (use the automaton network) with $k = 1$, and say whether G is of type $LL(1)$.

   (c) Say whether grammar G is of type $LL(2)$ by determining the lookahead sets with $k = 2$ that are necessary.

---

2. Consider the following grammar G (axiom S), not in extended form (BNF), over alphabet $\{ s, e \}$:

$$G \begin{cases} S \rightarrow S\, s\, E \mid E \\ E \rightarrow e\, E \mid s\, E \mid \varepsilon \end{cases}$$

Answer the following questions:

(a) Use the Earley method and show the simulation of the analysis of string $s\, s\, e\, e \in L(G)$. Write in the table prepared on the next page.

(b) Draw a syntax tree of string $s\, s\, e\, e$ and show the correspondence between the subtrees and the reduction candidates found during the simulation.

(c) (optional) Say whether grammar G is ambiguous or not, and in particular whether string $s\, s\, e\, e$ is ambiguous and how large its ambiguity degree is; if the string is ambiguous draw all its syntax trees.

Table for writing the simulation of the Earley algorithm

state 0	pos. s	state 1	pos. s	state 2	pos. e	state 3	pos. e	state 4

4 Translation and Semantic Analysis 20%

1. Consider a source language L to model an assembler program. Machine instructions are the following ones (on the right the interpretation is shown as a comment):

```
Instruction:                Interpretation (comment):
add s, d                    d <- s + d
add 1, d                    d <- d + 1
add -1, d                   d <- d - 1
nop                         no operation
```

   Arguments "s" and "d" refer to two general purpose registers of the processor, source and destination respectively; arguments "1" and "-1" are constant.

   The program is simply a list of instructions of the above described type, separated by the special character "⟨NL⟩" (newline).

   The program may contain errors of the following type: instruction "add" has a constant as destination argument (for instance "add s, 1").

   One wishes one designed a syntactic transducer T to translate the program by replacing current machine instructions with new optimised instructions (if possible), as follows (on the right there is an explicative comment)

```
Instruction:          Replacement:          Explanation
add s, d              add s, d              does not change
add 1, d              inc d                 is optimised
add -1, d             dec d                 is optimised
nop                                         is removed
add s, 1              error                 error warning
add s, -1             error                 error warning
```

   Answer the following questions:

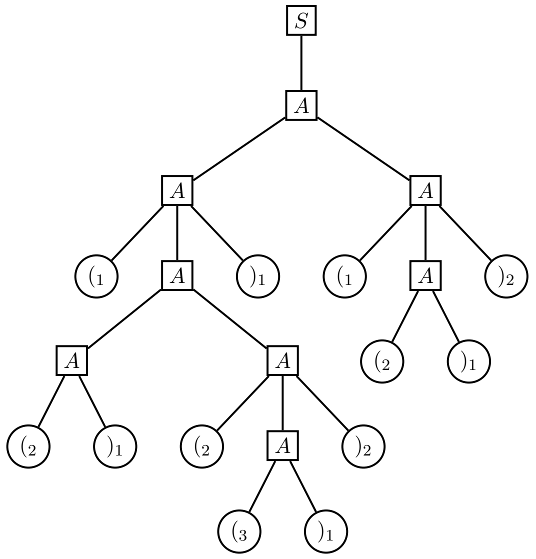
   (a) Say what the most convenient translator model could be (finite state automaton, pushdown automaton, syntax transduction grammar or scheme etc) to realise the transducer T described above.

   (b) After choosing the model, realise the transducer T described above.

   (c) (optional) Say whether transducer T is deterministic or not, and in the case it is indeterministic try to find a deterministic realisation (possibly by changing the transducer model).

2. Consider the Dyck language L with open and closed round brackets, that is over alphabet $\Sigma = \{\, \text{‘(’}, \text{‘)’} \,\}$; language L does not contain the empty string ε. Here is a syntactic support G that generates L (axiom S):

$$G \begin{cases} S \to A \\ A \to A\; A \\ A \to \text{‘(’}\; A\; \text{‘)’} \\ A \to \text{‘(’}\; \text{‘)’} \end{cases}$$



One wishes one associated to the open round bracket “(” a nesting depth index, and to the close round bracket “)” a position index in the list of the brother brackets (both indices are integer numbers ≥ 1). Here is a sample indexed string:

$$(_1\ (_2\ )_1\ (_2\ (_3\ )_1\ )_2\ )_1\ (_1\ (_2\ )_1\ )_2$$

The syntax tree with indices of the sample string is shown above on the right of the syntactic support G. Furthermore, one wishes one computed also the index of the total height of the string, which coincides with the nesting depth index of the most internal bracket pair.

The following attributes are assigned:

- α associated with S: total height of the string
- π associated with A: nesting depth of the current bracket pair
- λ associated with A: position of the current bracket pair

If it is necessary or advisable, the already assigned attributes may be extended and new attributes may be added.

Answer the following questions:

(a) If necessary, complete the list of attributes and say which attributes are of left or right type (synthesised or inherited). Use the table prepared on the next page.

(b) Write the semantic functions associated with the rules of grammar G. Use the table prepared on next page.

(c) Draw the dependence graphs of the attributes, for each rule of grammar G.

(d) Draw the brother graphs of the rules and check whether grammar G is of type one-sweep and possibly of type L.

attributes to be used in the grammar

type	name	(non)terminal	domain	meaning
already assigned in the exercise				
	α	S	integer.	total height of the string
	π	A	integer	nesting depth of the pair
	λ	A	integer.	position of the pair
existing attributes to be extended and new attributes				

syntax	*semantic functions*
$S_0 \rightarrow A_1$	
$A_0 \rightarrow A_1\ A_2$	
$A_0 \rightarrow$ ‘(’ A_1 ‘)’	
$A_0 \rightarrow$ ‘(’ ‘)’	