Master of Science in Computer Science - University of Camerino
Compilers A. Y. 2018/2019
Written Test of 6th February 2019 (Appello I)
Teacher: Luca Tesei

NOTE: Regular expressions are written and should be written using the usual rules of precedence: the $*$ operator has precedence on concatenation, which has precedence on the $|$ operator. The notation $(r)^+$ can be used with the usual meaning.

EXERCISE 1 (10 points)

Consider a lexical analyser designed for recognising the tokens p_1, p_2 and p_3 of the following regular definition.

$$
\begin{array}{lcl}
p_1 & \rightarrow & ba \\
p_2 & \rightarrow & ab \\
p_3 & \rightarrow & b(a|b)^*a
\end{array}
$$

Assume that the lexical analyser is designed following the classical two rules for matching tokens, i.e. at each step the token with the longest possible lexeme is selected and, if more than one lexemes are the longest ones and have the same length, the token that is defined in a higher position in the regular definition is selected.

1. List the sequence of tokens (and the relative lexemes) that will be emitted by the lexical analyser if the following input string is given: $abbabaa\$$. Justify your answer carefully possibly showing the sequence of steps made by the lexical analyser.

EXERCISE 2 (10 points)

Consider the following language:

$$L = \{a^n\, b^m\, c^n \mid n \geq 0, m > 0\} \cup \{c^{2n}\, ba\, c^n \mid n \geq 0\}$$

1. Is the language $\mathrm{LL}(1)$? Justify your answer and, if the answer is yes, provide the table for a top-down predictive parser for the language.
2. Is the language $\mathrm{LR}(1)$? Justify your answer.

EXERCISE 3 (12 points)

Consider a language of lists defined recursively in the following way:

- $()$ is a list and is the empty list;
- $(x_1, \ldots, x_k)$ is a list where each x_i can be an atom, i.e. a or b, or a list itself.

1. Give a Syntax Directed Translation Scheme for the language that is suitable for being implemented during bottom-up parsing (you don't need to show that the grammar is $\mathrm{LR}(1)$). The SDT must calculate an attribute for each list that gives the number of elements of the longest sub-list in the list, considering the list itself.

For instance, the value for the attribute of the list $(a, (a, b), a, b)$ must be 4, the value for $()$ must be 0, the value for $(())$ must be 1 (the list is not empty, and the empty list that is a sub-list must be considered just like an atom) and the value for the list $(a, (a, b, a))$ must be 3.