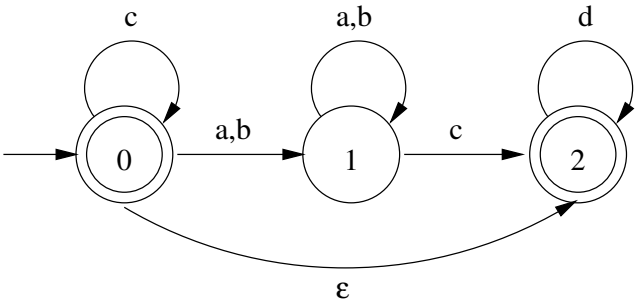
Master of Science in Computer Science - University of Camerino
Formal Languages and Compilers A. Y. 2018/2019
Written Test of 21st February 2019 (Appello II)
Teacher: Luca Tesei

NOTE: Regular expressions 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 the following automaton:



1. Express the language accepted by the automaton using a regular expression
2. Is the automaton deterministic? Justify your answer and if the answer is no, then give an equivalent deterministic automaton.
3. Is the given deterministic automaton minimum? Justify your answer.

EXERCISE 2 (12 points)

Consider the following grammar:

$$\begin{array}{lcl} S & \rightarrow & bSb \mid aAbB \\ A & \rightarrow & cA \mid cb \\ B & \rightarrow & aBc \mid ca \end{array}$$

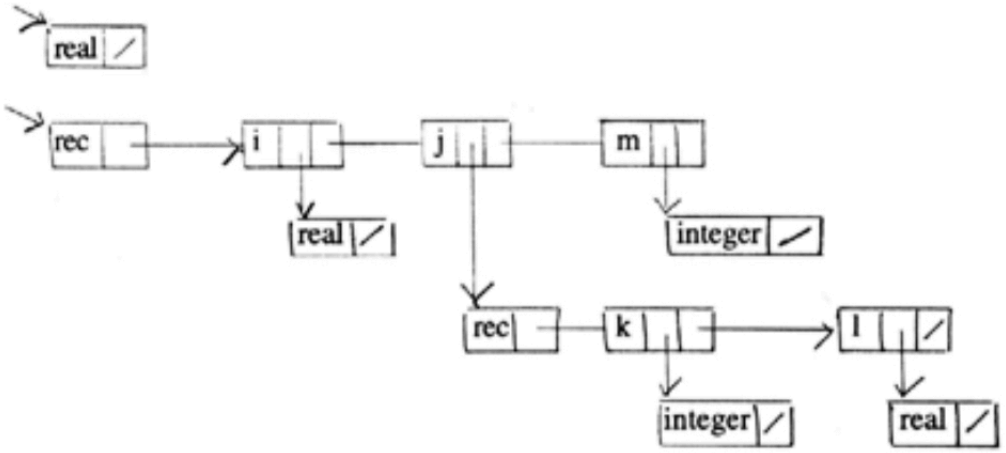
1. Write formally the language generated by the grammar as a set of strings.
2. Is the grammar LR(1)? Justify your answer and, if the answer is yes, give the table of a bottom-up shift-reduce parser for the grammar.

EXERCISE 3 (10 points)

Consider a language of types. A type can be **integer**, **real** or **record**. **record** types contain fields that can have type **integer**, **real** or **record**. As an example consider the following two type expressions of this language: **real** and

```
rec
        i: real,
        j: rec
                k: integer,
                l: real
        endrec,
        m: integer
endrec
```

1. Define a Syntax Directed Translation Scheme suitable to be implemented during top-down parsing for this language. The SDT has to construct, during the parsing, a structure that, for the examples given above, should look like the following figure:



The following operations can be used to construct the structure, whose pointers are called `StructPointer`:

- $\mathbf{newType}$: `String` $\times$ `StructPointer` $\rightarrow$ `StructPointer`, e.g. $\mathbf{newType}(\texttt{real}, \texttt{null})$ creates a structure representing the simple type **real** (the first example given);
- $\mathbf{newField}$: `String` $\times$ `StructPointer` $\times$ `StructPointer` $\rightarrow$ `StructPointer`, e.g. $\mathbf{newField}(\texttt{l}, \mathbf{newType}(\texttt{real}, \texttt{null}), \texttt{null})$ creates the sub-structure corresponding to the field `l` in the bottom-right part of the figure above.

For identifiers, the token **id** can be used and the corresponding attribute **id.name** can be used to obtain the string of the lexeme of the identifier.