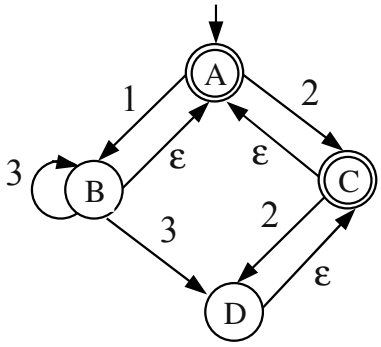
February 6, 2013

Compilers

Surname, Name	
Curriculum	

1. After generating the DFA equivalent to the following NFA, specify the BNF expressing the regular language relevant to the DFA.



2. Given the following grammar **G** in BNF notation,

$A \rightarrow B$ **a** $C \mid C$
$B \rightarrow C$ **b** $\mid$ **b**
$C \rightarrow A$ **b** $\mid$ **a**

we ask to:

- Transform **G** into a non left-recursive grammar **G'** (equivalent to **G**);
- Generate the complete LL(1) parsing table of **G'**.
- Based on the parsing table, establish whether **G'** is LL(1), providing relevant explanation.

3. After constructing the complete parsing automaton for grammar **G** introduced in point 2, determine whether **G** is SLR(1), providing relevant explanation.

4. Given the following BNF, relevant to a language for logical expressions,

expr $\rightarrow$ *expr* **or** *term* $\mid$ *term*
term $\rightarrow$ *term* **and** *factor* $\mid$ *factor*
factor $\rightarrow$ **not** *expr* $\mid$ **(** *expr* **)** $\mid$ **id** $\mid$ **true** $\mid$ **false**

we ask to codify in *Yacc* the code generator of logical expressions for a P-machine. The P-code shall be generated as strings of characters (`define YYSTYPE *char`). The following auxiliary functions are provided:

- `char* code(char* operator, char* argument)`, generating the string containing the single P-code statement "`operator argument`", where `argument` may be empty;
- `char* concode(char* pcode1, char* pcode2 ...)`, generating the string containing the catenation of two or more strings of code (separated by a newline) "`pcode1 \n pcode2 ...`".

The lexical string relevant to **id** is assumed to be assigned by the lexical analyzer (not to be codified) in the *Yacc* variable `char* lexval`. Operator **and** is evaluated in short circuit, while operator **or** is fully evaluated.

The P-machine includes the following set of instructions:

- `LDC` *const*: load boolean constant *const*;
- `LOD` *var*: load value of boolean variable *var*;
- `AND`: logical conjunction;
- `OR`: logical disjunction;
- `NOT`: logical negation;
- `LAB` *label*: create label;
- `GOF` *label*: conditional (to false) jump;
- `GOT` *label*: unconditional jump.

5. Specify the (extended) attribute grammar relevant to the following BNF,

program → *def-relation* *extend-relation*
def-relation → **relation id (** *id-list* **)**
id-list → **id ,** *id-list* | **id**
extend-relation → **extend id by id** = *expr*
expr → *expr* + *term* | *expr* - *term* | *term*
term → **id** | **num**

```
relation R (a, b, c)
extend R by n = a + c - 25
```

based on the following semantic constraints:

- Attributes are implicitly of integer type;
- Names of attributes are unique,
- The operand of the `extend` is the defined relation,
- The new attribute does not belong to the relation,
- Each identifier within the expression is an attribute of the relation,

and the following requirements:

- The set of semantic attributes is { ok, name },
- A symbol table is used to catalog table attributes by means of the following functions:
  ```
  void insert(attr)
  bool lookup(attr)
  ```
- Function `lookup(name)` returns `true` if the attribute is cataloged, otherwise it returns `false`,
- A possible intermediate semantic error does not terminate the semantic analysis.

6. With reference to the BNF introduced in point 5, assuming that the syntax tree of the phrase is semi-abstract (only irrelevant lexical sugar is removed from the concrete tree) and binary (pointers: `child`, `brother`), we ask first to outline the semi-abstract syntax tree relevant to the example phrase, and then to codify a procedure for P-code generation based on the following requirements:

- Within the syntax tree, each lexical value is stored as a string in field `lexval`;
- The translation scheme of the definition of relation *relname* is composed by a first instruction `NEW` *relname* followed by several instructions `ATTR` *attrname*, one for each attribute *attrname* in the relation, terminated by the instruction `END`;
- The translation scheme of the extension of relation *relname* with new attribute *attrname* is composed by a first instruction `EXT` *relname attrname*, followed by the translation of the attribute-value expression, terminated by the instruction `END`;
- The set of instructions for the P-machine also includes `ADD`, `SUB`, `LDC`, and `LOD`, with the usual meaning.
- The auxiliary function `emit(string ...)` , with one or more string operands, is used to print an instruction of the P-machine.