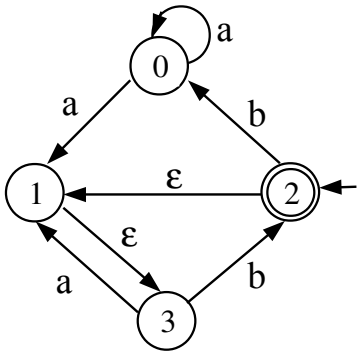
July 9, 2013

Compilers

Surname, Name	
Student identifier	

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



2. Codify the recursive-descent parser of the language defined by the following EBNF, also checking that phrases end with an `EOF` (end-of-file).

program → *stat-list*
stat-list → {*stat* **;**}$^{+}$
stat → *def-stat* | *assign-stat* | *case-stat*
def-stat → **var id** {**, id**} **is** *type*
type → **integer** | **string** | **matrix (intconst** {**, intconst**} **)** **of** *type*
assign-stat → **id** = *const*
const → **intconst** | **strconst** | *matconst*
matconst → **[** *const* {**,** *const*} **]**
case-stat → **case id of** { *const* **:** *stat* **;** }$^{+}$ [**default :** *stat* **;**] **end**

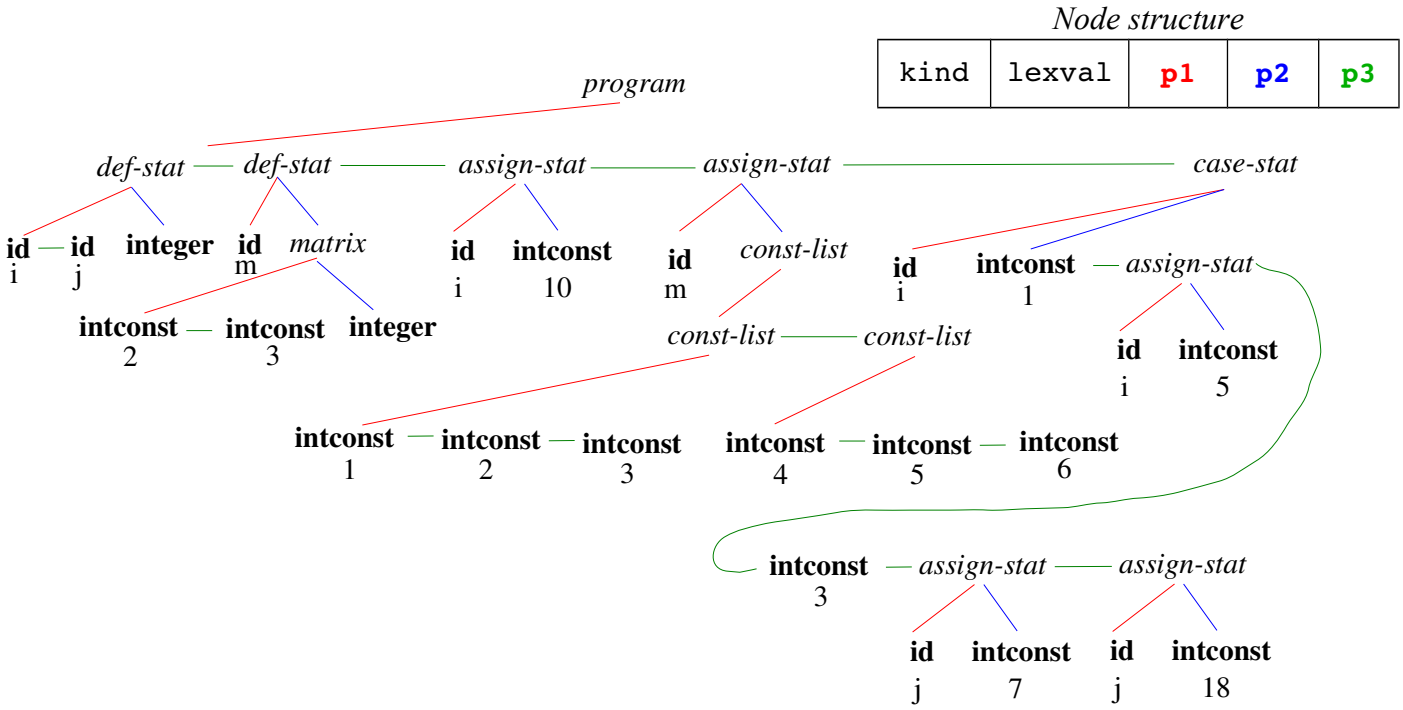
3. After constructing the complete parsing automaton for the following BNF, discuss whether the BNF is LR(1) .

S → *S* **a** *S* | **b** *T*
T → **a** | **ε**

4. Codify in *Yacc* the generator of the ternary abstract trees based on the following BNF and structures:

program → *stat-list*
stat-list → *stat* **;** *stat-list* | *stat* **;**
stat → *def-stat* | *assign-stat* | *case-stat*
def-stat → **var** *id-list* **is** *type*
id-list → **id ,** *id-list* | **id**
type → **integer** | **string** | **matrix (** *intconst-list* **)** **of** *type*
intconst-list → **intconst ,** *intconst-list* | **intconst**
assign-stat → **id** = *const*
const → **intconst** | **strconst** | *matconst*
matconst → **[** *const-list* **]**
const-list → *const* **,** *const-list* | *const*
case-stat → **case id of** *branch-list opt-default* **end**
branch-list → *branch* **,** *branch-list* | *branch*
branch → *const* **:** *stat* **;**
opt-default → **default :** *stat* **;** | **ε**

```
var i, j is integer;
var m is matrix(2,3) of integer;
i = 10;
m = [[1,2,3],[4,5,6]];
case i of
  1: i = 5;
  3: j = 7;
  default: j = 18;
end;
```



Note: Before the Yacc specification, outline the mapping between the concrete syntax tree and the abstract syntax tree for the given example phrase.

5. Specify the (extended) attribute grammar relevant to the BNF defined in point **4**, based on the following semantic constraints only:

- Variable names are unique;
- Referenced variables shall exist;
- Each dimension in matrix definition is greater than zero;
- In case statement, the case variable (**id**) is of simple type (either integer or string);
- In case statement, each case constant has the same type of the case variable;
- Variables are assigned with constants of the same type (in case of matrix, no type-checking of the deep structure of the matrix is required).

Notes:

- Lexical values of terminals are accessed through `lexval`;
- A symbol table is used to catalog variables by means of the following functions:
  `void insert(name, type)`: insert variable `name` with `type`;
  `Type lookup(name)`: returns the type of variable `name` (`INT`, `STR`, `MAT`) if cataloged, otherwise `NULL`;
- In case of semantic error, function `semerror(string msg)` is called, which prints a pertinent error message `msg`, and then terminates the analysis.

6. With reference to the abstract tree relevant to the grammar in point **4**, where the case statement is assumed to be missing, codify a procedure of P-code generation for a virtual machine involving the following set of instructions:

- **`DEF`** *id*: allocate variable named *id*;
- **`LDA`** *id*: load address of variable named *id*;
- **`LDI`** *constant*: load integer *constant*;
- **`LDS`** *constant*: load string *constant*;
- **`HLT`**: halt program (the last instruction of the generated code).

Note: To load a matrix constant in assignment, all involved atomic constants (from left to right) must be loaded before the (unique) final store.