PCS 3566/3866 - Linguagens e Compiladores - Aulas 3 e 4

Rodrigo Perrucci Macharelli - 9348877

Formação de Autômatos Para Dortmund Basic a partir da notação de Wirth

Introdução

Utiliza-se neste este projeto uma técnica baseada em notações de gramáticas (como por exemplo a de Wirth) para traduzir as regras de formação para autômatos finitos e, posteriormente em código.

Para isso, parte-se de uma gramática definida, a qual se altera para um modelo de notação padronizado (após aplicar possíveis otimizações), possibilitando a interpretação como máquina de estados e, posteriormente, utiliza-se um simples algoritmo para transformá-lo para uma estrutura de código que pode ser implementado.

Definindo a Gramática no formato padronizado

A gramática a seguir descrita é bastante semelhante com a original mostrada em aula, apenas com algumas diferenças, em particular a regra de Exp foi quebrada em Term e Eb para promover a ordenação dos operadores e os identificadores podem ter mais de um digito/numero.

```
Program : BStatement BStatement* INTEGER END

BStatement : INTEGER Assign|Read|Data|Print|Goto|If|For|Next|Dim|Def|Gosub|Return|Remark

Assign : LET Var EQUAL Exp

Var : ID
    | ID LPAREN Exp (COMMA Exp)* RPAREN

Exp: Term (PLUS|MINUS Term)*

Term: Eb (MUL|DIV Eb)*

Eb: LPAREN Exp RPAREN| INTEGER |Var|FN ID LPAREN Exp RPAREN
```

```
Read : READ Var (COMMA Var)*

Data : DATA Snum (COMMA Snum)*

Print : PRINT
      | PRINT Pitem (COMMA Pitem)* | (COMMA Pitem)* COMMA

Pitem : Exp | QUOTES CHARACTER(CHARACTER)* QUOTES (Exp)*

Goto : (GOTO | GO TO) Integer

If : IF Exp (GRTEQL | GRT | NOTEQL | LESSEQL | LESS | EQUAL) Exp THEN INTEGER

For : FOR ID EQUAL Exp TO Exp | FOR ID EQUAL Exp TO Exp STEP Exp

Next : NEXT ID

Dim : DIM ID LPAREN INTEGER (COMMA INTEGER)* RPAREN (COMMA ID LPAREN INTEGER (COMMA INTEGER)

Def : DEF FN ID LPAREN ID RPAREN EQUAL Exp

Gosub : GOSUB INTEGER

Return : RETURN

Remark : REM (CHARACTER)*

Snum = PLUS | MINUS INTEGER

INTEGER: digit(digit)*

CHARACTER = letter | digit | special

ID: letter(digit|letter)*
```

Lista de Terminais Identificados

- INTEGER
- PLUS +
- MINUS -
- MUL *
- DIV /
- EQUAL =
- GRTEQL >=

- GRT >
- NOTEQL <>
- LESSEQL <=
- LESS <
- LPAREN (
- RPAREN)
- COMMA ,
- QUOTES "
- LET
- FN
- READ
- DATA
- PRINT
- GOTO
- GO
- TO
- IF
- THEN
- FOR
- TO
- STEP
- NEXT
- DIM
- DEF
- GOSUB
- RETURN
- REM

Descrição do Algoritmo de Transformação do Autômato para Código

A seguir descreve-se um algoritmo simples para transformar a notação de gramática em código implementável para a geração do analisador sintático (Parser)

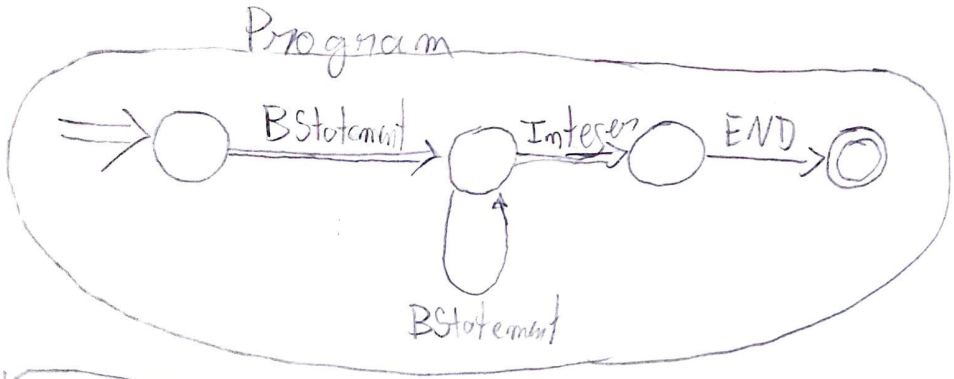
1. Cada regra de produção é representada como um método. As referências a esta regra são as chamadas deste método, que por sua vez segue a estrutura definida pela regra correspondente.
2. Elementos opcionais são transformados em blocos condicionais.
3. Um agrupamento com repetição se transforma em um laço.
4. Cada não terminal é consumido. Isso é feito verificando se o tipo do não terminal sendo consumido é o esperado segundo as normas da gramática e,

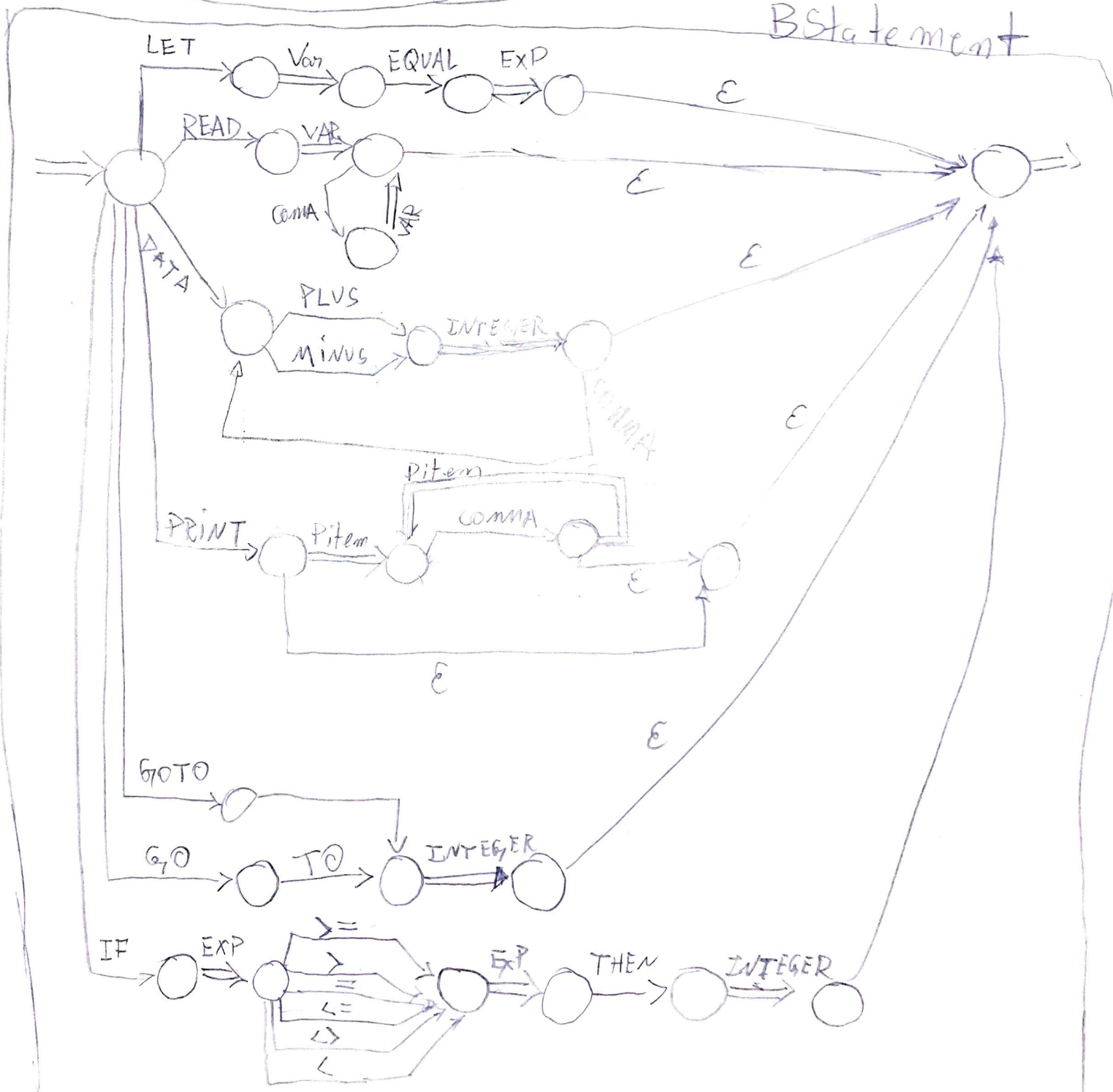
em caso positivo, obtém o próximo token a ser analisado.

Conversão da Gramática para Representação de Autômatos

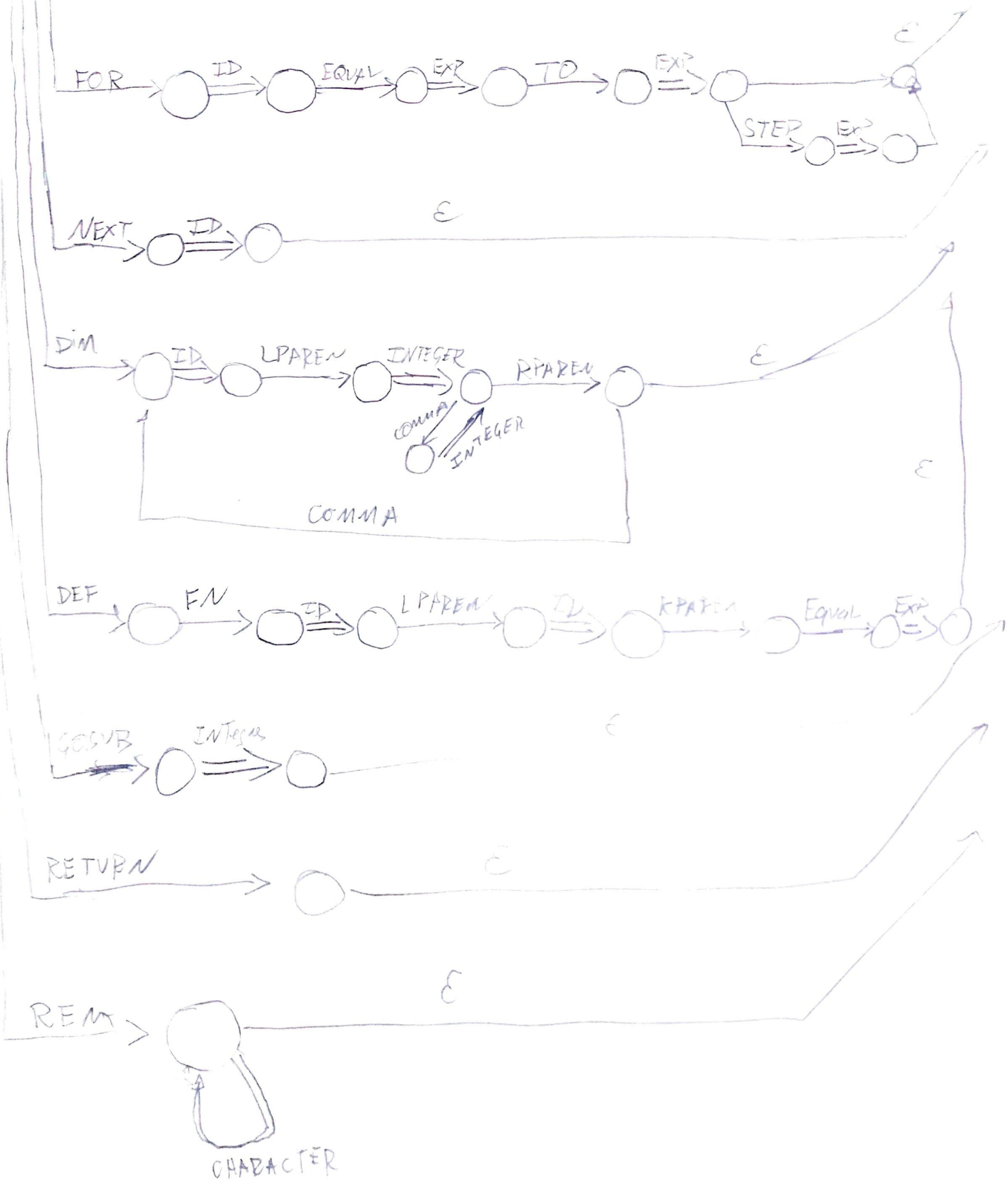
A seguir mostram-se os resultados da representação da gramática do Dortmund Basic por Autômatos

Autômatos P/ Gramática Dortmund Basic



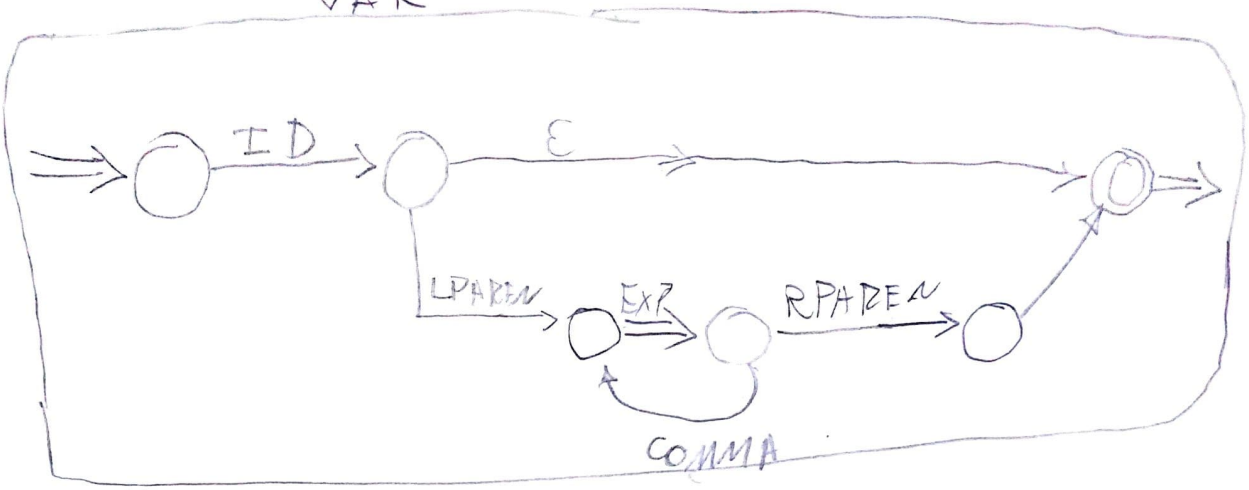


Continua...

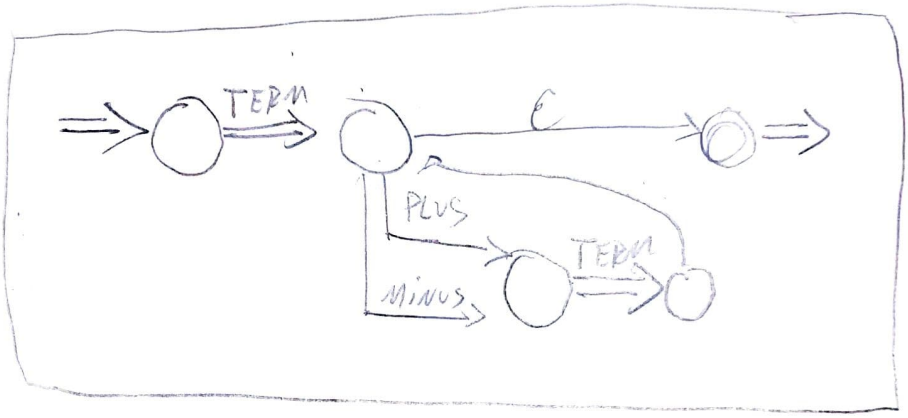

FOR
ID
EQUAL
EXP
TO
EXP
STEP
EXP
ε
NEXT
ID
ε
DIM
ID
LPAREN
INTEGER
RPAREN
ε
COMMA
INTEGER
COMMA
DEF
FN
ID
LPAREN
ID
RPAREN
EQUAL
EXP
GOSUB
INTEGER
ε
RETURN
ε
REM
ε
CHARACTER

Bstatement

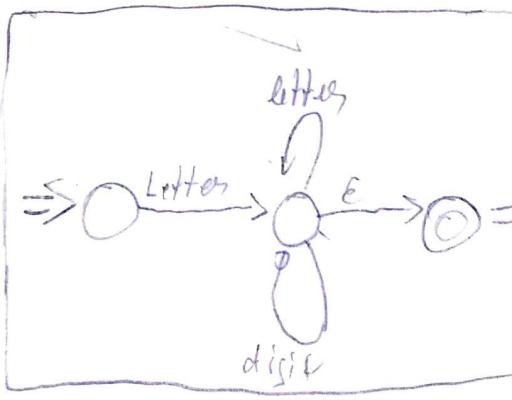
VAR



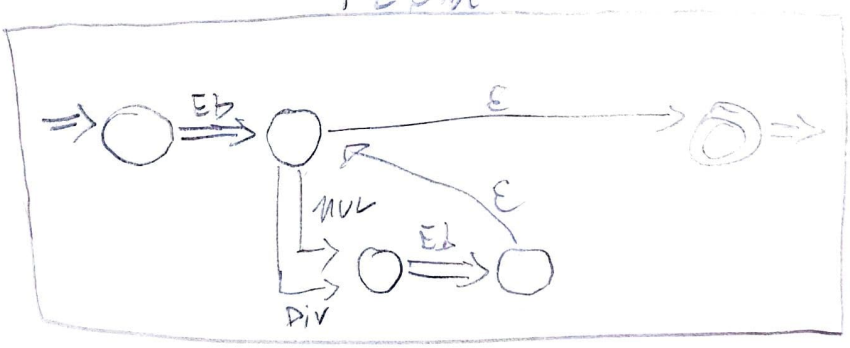
EXP



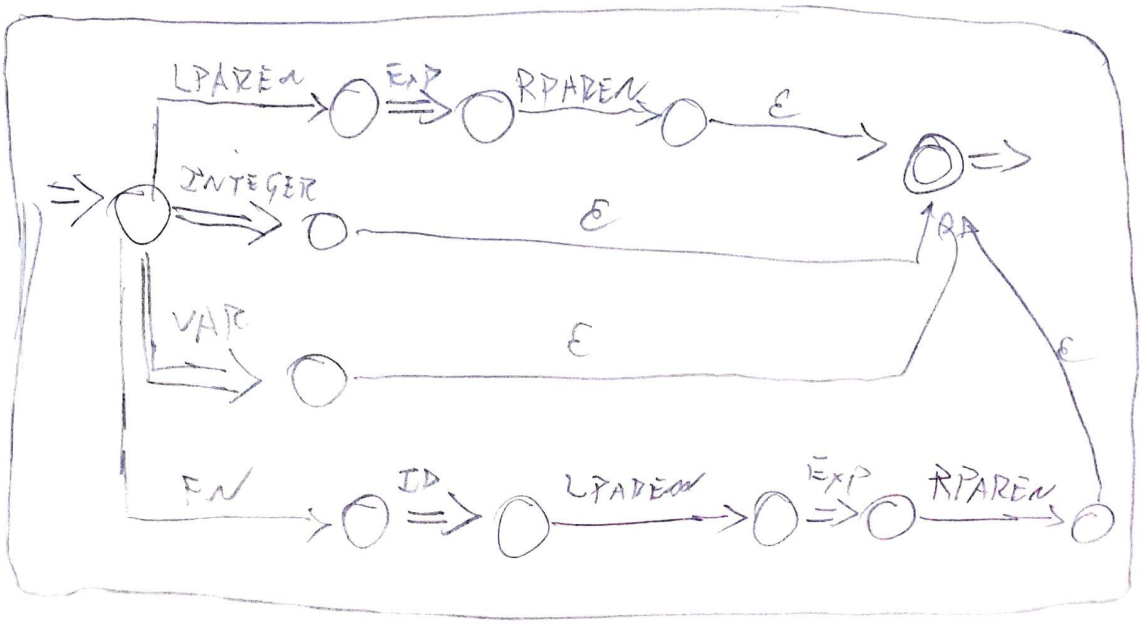
ID



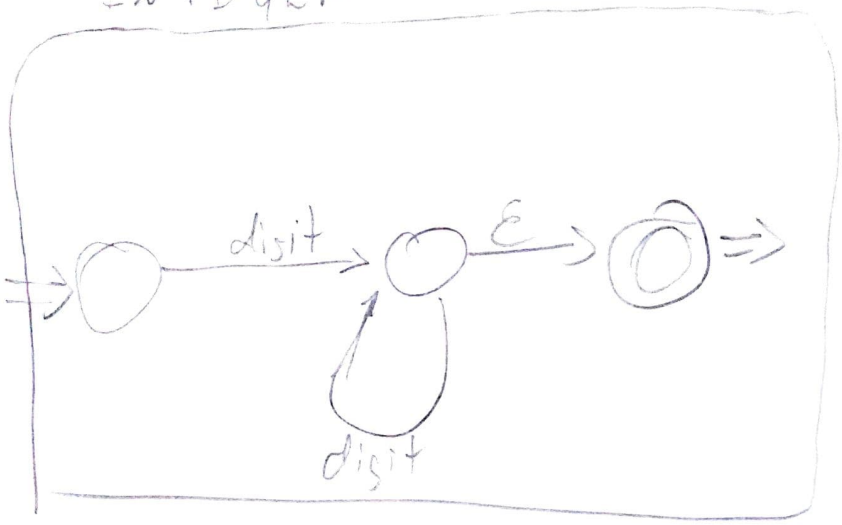
TERM



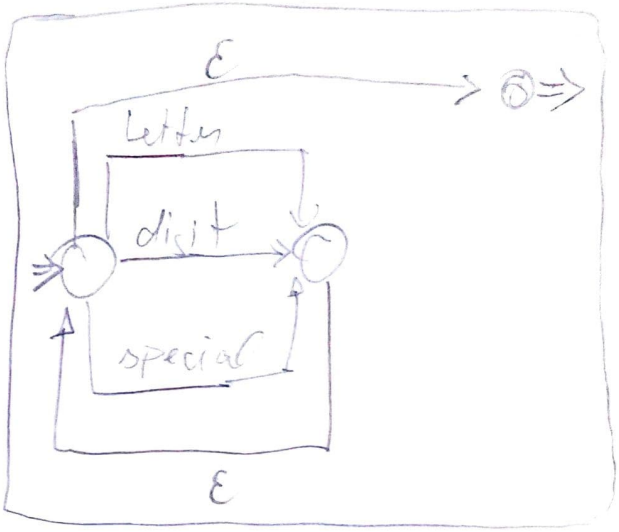
Eb



INTEGER



Character



Pitem