# Relatório

Bdex - Linguagem para manipulação de tabelas

Linguagens **Formais e Autómatos(LFA)**

Ricardo Antunes, Diogo Fontes, Hugo Domingos, Tiago Marques, Diogo Santos

universidade de aveiro
theoria poiesis praxis

DETI - Universidade de Aveiro

Ricardo Antunes, Diogo Fontes, Hugo Domingos,
Tiago Marques, Diogo Santos
(98275) ricardofantunes@ua.pt, (98403) diogo.fontes@ua.pt,
(98502) h.domingos@ua.pt, (98459) tamarques@ua.pt
(98393) diogoejsantos@ua.pt

20 de junho de 2021

# Conteúdo

# 1 Linguagem

## 1.1 Bdex

A gramática Bdex, é uma linguagem para manipulação de tabelas. Esta contem meios para a incorporação de tabelas definidas pela linguagem secundária.

### 1.1.1 Declaração de variáveis

```
164   VAR: LETTER (LETTER|DIGIT)*;
165   fragment LETTER: [a-zA-Z_];
166   fragment DIGIT: [0-9];
```

A definição das variáveis é feita da seguinte maneira: uma letra seguida ou não de outras letras ou dígitos.

Esta variável é composta por letras minúsculas ou maiúsculas de A a Z e por números compreendidos entre 0 e 9.

Exemplos:

- Table1;
- T12;
- table;

### 1.1.2 Operações sobre tabelas

```
create_table: // Cria uma tabela
    'create_table' '[' columns ']'                  # CreateTableNoLine
  | 'create_table' '[' columns ']' '[' line=expr ']'  # CreateTableLine
  | 'create_table' '[' columns ']' ':' '[' values ']' # CreateTableValues
  ;

add_line: // Adiciona uma linha de uma tabela
     'add_line' tablename=VAR index=expr? '[' values ']' ';' # AddLineValues
  |  'add_line' tablename=VAR index=expr? ';'               # AddLineIndexOmission
  ;

add_column: // Adiciona Coluna a tabela
    'add_column' tablename=VAR where=('after' | 'before') '[' place=column ']' '[' to=column ']' ';'              # AddColumnOmissionPlace
  | 'add_column' tablename=VAR where=('after' | 'before') '[' place=column ']' '[' to=column ']' '[' values ']' ';' # AddColumnValuesPlace
  | 'add_column' tablename=VAR where=('first' | 'last')?  '[' to=column ']' ';'                                    # AddColumnOmissionFL
  | 'add_column' tablename=VAR where=('first' | 'last')?  '[' to=column ']'    '[' values ']' ';'                # AddColumnValuesFL
  ;
```

Create_table: Permite criar tabelas de diferentes formas, isto é, tabelas só com colunas, com colunas e um certo número de linhas e tabelas com colunas e os respetivos valores.

Add_line: Permite adicionar linhas de duas formas. Adicionar uma linha num índice específico com valores por omissão ou com os respetivos valores.

Add_column**:** Permite adicionar colunas de diferentes formas:

- Adicionar uma coluna antes ou depois de uma respetiva coluna com valores por omissão ou com os respetivos valores.

- Adicionar uma coluna no início ou no fim com valores por omissão ou com os valores pretendidos.

Exemplos:
- Table table1 = create_table [“Nome” (Text)] [1+1];
- add_line table1 1*3 [“Cristiano”];
- add_column table1 last [“Nº mec” (Integer)];

```
remove_column: //Remove a coluna/s
    'remove_column' tablename=VAR '[' column ']' ';'
  ;

remove_line: // Remove uma linha de uma tabela
    'remove_line' tablename=VAR index=expr? ';'
  ;

put_value: //puts value, if the a value is already there substitui
    'put_value' tablename=VAR '[' column ',' expr ']' ':' '[' expr ']' ';'
  ;
```

Os métodos remove_column e remove_line servem para remover uma coluna ou uma linha, respetivamente.

O método put_value permite inserir um valor na respetiva linha e coluna de uma tabela.

Exemplos:
- remove_column table1 [“Nº mec” (Integer)];
- remove_line table1 1;
- put_value table1 [“Nome” (Text)], 2] : [“Nani”];

```
extract_table: // Extrai uma linha/coluna de uma tabela existente com critérios
    'extract from' reftable=VAR 'columns.' '[' columns ']'                                          # ExtractTableColumns
  | 'extract from' reftable=VAR 'columns.' '[' column ',' column ']'                                # ExtractTableColumnsInterval
  | 'extract from' reftable=VAR 'line.' '[' values ']'                                              # ExtractTableLines
  | 'extract from' reftable=VAR 'line.' '[' expr ',' expr ']'                                       # ExtractTableLineBreak
  | 'extract from' reftable=VAR 'columns.' '[' columns ']' 'line.' '[' expr ',' expr ']'            # ExtractTableBothColumns
  | 'extract from' reftable=VAR 'columns.' '[' column ',' column ']'  'line.' '[' expr ',' expr ']' # ExtractTableBothInterval
  | 'extract from' reftable=VAR 'contains' '[' values ']'                                           # ExtractTableContains
  ;

join_tables: // Junta tabelas
    'join' table (',' table)+
  ;

print: //Print de uma variavel
    'print' VAR ';'   # PrintVar
  | 'print' table ';' # PrintTable
  | 'print' expr ';'  # PrintExpr
  | 'print' assign    # PrintAssign
  ;

read_table: //Ler tabela do terminal
    'read' tablename=VAR ';'
  ;
```

Extract_table:  Permite extrair linhas ou colunas de uma tabela.

Join_table: Tem como função a junção de tabela.
Print: Permite imprimir uma tabela, uma expressão ou uma variável.

Read_table: Lê uma tabela do terminal.

Exemplos:

- extract from table1 columns. [“Nome” (Text)] line. [0+0, 1+2];
- join table1, table2, table3;
- print table2;
- read table3;

```
89  assign: // Atribui uma tabela ou valor a uma variável
90      declaration '=' table ';' # AssignDeclarationTable
91    | declaration '=' expr  ';' # AssignDeclarationExpr
92    | declaration '=' put_value # AssignDeclarationPut
93    | VAR '=' table ';'         # AssignTable
94    | VAR '=' expr ';'          # AssignExpr
95    | VAR '=' put_value         # AssignPut
96    ;
106 iterate: // Itera sobre uma tabela, completa, por linha ou por coluna???
107     'iterate' tablename=VAR ';'                       # IterateTable
108   | 'iterate' tablename=VAR 'line' index=expr ';'     # IterateLine
109   | 'iterate' tablename=VAR 'column' method=column ';' # IterateColumn
110   ;
111
112 load_table: // Lê uma tabela de um ficheiro CSV
113     'load' tablename=VAR filename=TEXT ';'
114   ;
115
116 save_table: // Guarda uma tabela num ficheiro CSV
117     'save' tablename=VAR filename=TEXT ';'
118   ;
```

Assign: Tem como função associar uma tabela ou uma expressão a uma variável.

Iterate: Permite iterar sobre uma tabela.

Load_table: Tem como função ler uma tabela de um ficheiro CSV.

Save_table: Tem como função guardar uma tabela num ficheiro CSV.

Exemplos:
- iterate table1;
- load table5 ficheiro2;

### 1.2 Table

Table é a gramática secundária. Esta tem como objetivo reconhecer uma tabela de um ficheiro CSV.
A classe TableMain tem um método invocado pela gramática principal (Bdex.g4), capaz desse efeito.

## 2 Compilação

Para a compilação é necessário executar os seguintes comandos:

```
antlr4 -visitor Table.g4
javac Table*.java
antlr4 -visitor Bdex.g4
javac Bdex*.java
```

Também é possível compilar com o seguinte comando: `bash compile`

## 3 Execução

Para a execução pode usar o seguinte comando:

```
antlr4-java -ea BdexMain "$1"
```

Ou então `bash run`

Para executar o ficheiro Output.java deve usar:

```
javac Output.java
Java Output
```

## 4 Programas de teste e gestão de erros

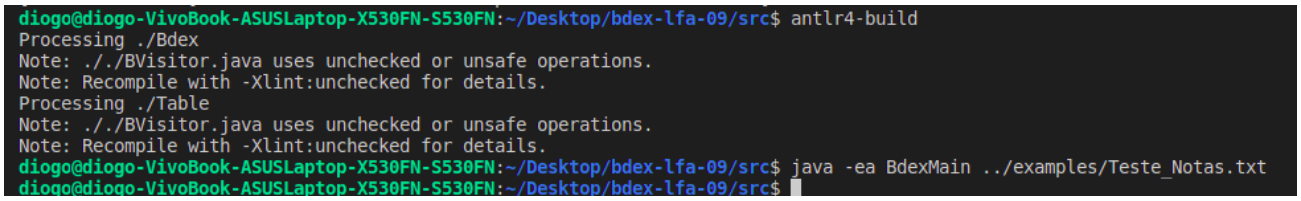

Figura 1: Compilação e execução do examples/Ficheiro.txt

Ao executar os comandos da figura acima, é criado um ficheiro Output.java no diretório src.

```java
public class Output {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      Column[] v8 = new Column[5];
      Column v9 = new Column("Nome", "Text");
      v8[0] = v9;
      Column v10 = new Column("Nmec", "Int");
      v8[1] = v10;
      Column v11 = new Column("Nota1", "Real");
      v8[2] = v11;
      Column v12 = new Column("Nota2", "Real");
      v8[3] = v12;
      Column v13 = new Column("NotaFinal", "Real");
      v8[4] = v13;
      Table v14 = new Table(v8);

      Table table1 = v14;

      table1.printTable();
      String v15 = "-----------------------------------------------------------------------------------------";
      System.out.println(v15);
      Object[] v16 = new Object[5];
      String v17 = "Diogo.F";
      v16[0] = v17;
      int v18 = 98403;
      v16[1] = v18;
      int v19 = 12;
      v16[2] = v19;
      int v20 = 14;
      v16[3] = v20;
      int v21 = 13;
      v16[4] = v21;
      table1.addLineLast(v16);

      Object[] v22 = new Object[5];
      String v23 = "Ricardo";
      v22[0] = v23;
      int v24 = 98275;
      v22[1] = v24;
      int v25 = 14;
      v22[2] = v25;
      int v26 = 19;
      v22[3] = v26;
      int v27 = 16;
      v22[4] = v27;
      table1.addLineLast(v22);
```

Figura 2: Parte do ficheiro Output.java

```
diogo@diogo-VivoBook-ASUSLaptop-X530FN-S530FN:~/Desktop/bdex-lfa-09/src$ java -ea BdexMain ../examples/errossem.txt
[ERROR at line 1] no viable alternative at input 'create table["C0"(integer'
[ERROR at line 4] no viable alternative at input 'add_columntable1after[1'
[ERROR at line 5] mismatched input 'Double' expecting {'Int', 'Real', 'Text'}
[ERROR at line 5] mismatched input ')' expecting {']', '(', '+', '-', '*', '/', '%', '==', '!=', 'and', 'or', '<', '<=', '>', '>='}
[ERROR at line 6] mismatched input 'intruso1' expecting {']', '(', '+', '-', '*', '/', '%', '==', '!=', 'and', 'or', '<', '<=', '>', '>='}
[ERROR at line 7] no viable alternative at input 'extractfromt5columns['
diogo@diogo-VivoBook-ASUSLaptop-X530FN-S530FN:~/Desktop/bdex-lfa-09/src$
```

Figura 3: Mensagens de erro ao compilar o examples/Ficheiro.txt

# 5 Contribuição dos autores

No desenvolvimento do trabalho todos os autores contribuíram igualmente para a realização do projeto em todas as suas componentes.
Autoavaliações:
Ricardo Antunes -> 20%
Diogo Fontes -> 20%
Hugo Domingos -> 20%
Tiago Marques -> 20%
Diogo Santos -> 20%