



Universidade do Minho  
Mestrado em Engenharia Informática  
Engenharia de Linguagens  
Engenharia Gramatical - Grupo 1  
**Resolução das Fichas 3 e 5**  
Ano Lectivo de 2012/2013

pg22820 - **António Silva**  
pg22781 - **Rui Brito**

16 de Novembro de 2012

## Conteúdo

<b>1</b>	<b>Introdução</b>	<b>3</b>
<b>2</b>	<b>Ficha 3</b>	<b>3</b>
2.1	Composição do Corpo . . . . .	3
2.2	Código Java . . . . .	3
2.3	Exemplo de Input . . . . .	3
<b>3</b>	<b>Ficha 5</b>	<b>4</b>
3.1	Exemplo de frase válida e Árvore de Derivação . . . . .	4
3.2	Lista de Autores . . . . .	4
3.3	Recursividade à esquerda para Recursividade à direita . . . . .	4
<b>4</b>	<b>Gramática Tradutora</b>	<b>5</b>
<b>5</b>	<b>Conclusão</b>	<b>9</b>
<b>A</b>	<b>Imagens</b>	<b>10</b>

## 1 Introdução

Este primeiro trabalho de *Engenharia Gramatical* de avaliação, da Unidade Curricular de Especialização *Engenharia de Linguagens*, consiste na realização da ficha 3 e 5 disponibilizadas no Blackboard.

## 2 Ficha 3

### 2.1 Composição do Corpo

Para escrever uma gramática tradutora, foi necessário completar a informação sobre o corpo. Assim inicialmente o corpo da factura era um conjunto de linhas, em que cada linha era `('codartigo ','designacao ','pvu ','quantidade ')`. Depois para suportar o pedido da alínea c, cada linha passou a ser somente `('codartigo ','quantidade ')`.

### 2.2 Código Java

Para conseguirmos saber os totais dos produtos, com várias facturas (sendo que cada factura possuía um id alfanumérico), foi criado um *hashmap* para associar a cada id de factura uma lista de valores, que era o total de cada linha da factura. No final, é possível apresentar o o total de cada linha em cada factura e ainda o tal de cada factura (que mais não é que a soma dos totais das linhas). Para se obter o Preço Unitário, que a pedido da alínea c) deveria já ter sido indicado no início, foi também criada um *hashmap* com a correspondência entre o código do produto e os seus atributos (guardados numa classe). Assim, por cada linha só era necessário obter o PVU através do código do artigo, e multiplicá-lo pela quantidade.

### 2.3 Exemplo de Input

```
a1 "xpto" 3.6 50
a2 "outro" 1 60.5
a3 "mais um" 4.99 4
---
```

```
f1
"Nome 1" "NIF 1" "Morada 1" "NIB 1"
"Nome 2" "NIF 2" "Morada 2"
(a1,5) (a3,2)
;
```

```
f2
"Nome 3" "NIF 3" "Morada 3" "NIB 3"
"Nome 4" "NIF 4" "Morada 4"
(a2,9.5) (a1,5) (a3,2)
;
```

```
f3
"Nome 5" "NIF 5" "Morada 5" "NIB 5"
```

"Nome 6" "NIF 6" "Morada 5"  
(a2,2.25)  
.

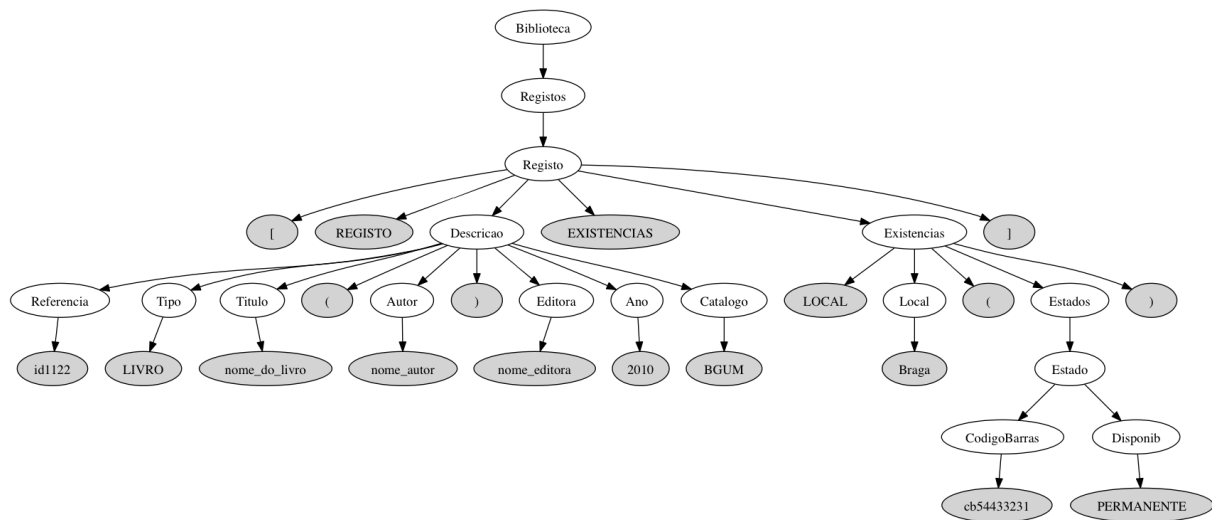
### 3 Ficha 5

#### 3.1 Exemplo de frase válida e Árvore de Derivação

Um exemplo rudimentar de frase válida seria:

```
[ REGISTO id1122 LIVRO "nome_livro" ("nome_autor") "nome_editora" 2010 BGUM
EXISTENCIAS LOCAL "Braga" ( cb54433231 ; PERMANENTE ) ]
```

A árvore de derivação corresponde seria como se segue:



**Figura 1:** Árvore de Derivação

Imagem em maior detalho na figura 4

#### 3.2 Lista de Autores

Para permitir uma lista de autores ao contrário de apenas um, basta extender a produção dez. Ficaria, então, algo como segue abaixo:

```
p10: Autores --> Autor
p11:          | Autores ',' Autor
```

#### 3.3 Recursividade à esquerda para Recursividade à direita

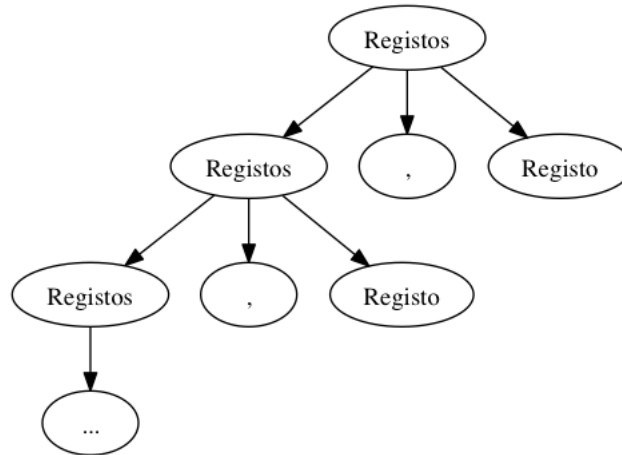
Recursividade à esquerda:

```
p1: Registos --> Registo
p2:          | Registos ',' Registo
```

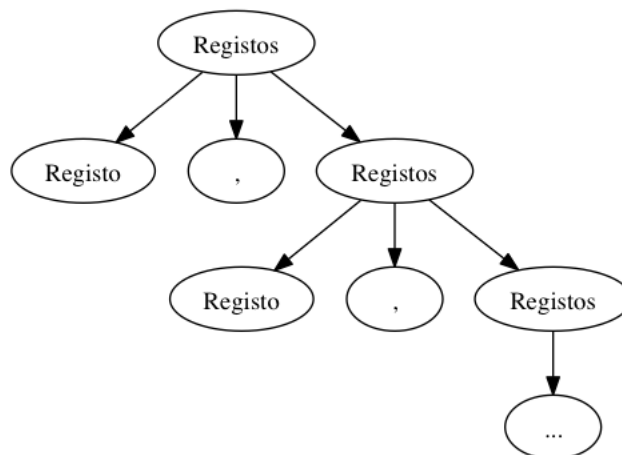
O que transformado para recursividade à direita (assumindo a notação eBNF):

```
p1: Registos --> Registo
p2:          | Registo ',' Registos
```

As diferenças quando à árvore de derivação são apresentadas a seguir:



**Figura 2:** Recursivo à esquerda



**Figura 3:** Recursivo à direita

## 4 Gramática Tradutora

Os pontos pedidos foram os seguintes:

1. calcular e imprimir: o número de registros; e o número de livros existentes para cada registro.
2. o número total de livros com estado RESERVADO/PERMNEENTE/ESTANTE.
3. identificar e listar por ordem alfabética os títulos dos livros.

4. verificar se não existem registos com a mesma referência.

```
grammar ficha5;

options{
k=2;
language=Java;
}

@header{
import java.util.TreeSet;
}

@members{
int numRegistos = 0;
int numLivros = 0;
String refReg = null;

int reservados = 0;
int permanentes = 0;
int estante = 0;

Boolean isBook = false;

TreeSet<String> bookNames = new TreeSet<String>();
TreeSet<String> refs = new TreeSet<String>();
}

biblioteca
@after{
System.out.println ("Livros Reservados: " + reservados);
System.out.println ("Livros Permanentes: " + permanentes);
System.out.println ("Livros em Estante: " + estante);

System.out.println("=== Livros ===");
for (String s : bookNames) {
System.out.println(s);
}

}
: registos
;

registos
: registo{numRegistos++;} (',' registo{numRegistos++;})*
;

registo
@after {
System.out.println(refReg + ": " + numLivros);
numLivros = 0;
refReg = null;
}
: '[ REGISTO ' descricao ' EXISTENCIAS ' existencias ']'
;
```

```

;

descricao
: referencia {refReg = $referencia.text;
               if(!refs.add($referencia.text)){
                   System.out.println("Referencia ja existente: " + $referencia.text);}
               }
               tipo titulo {if(isBook) {
                   bookNames.add($titulo.text);isBook = false;}
                   } '(' autores ')' editora ano catalogo
;

autores
: autor (',' autor)*
;

referencia
: ID
;

tipo: 'LIVRO' {numLivros++; isBook = true;}
| 'CDROM'
| 'OUTRO'
;

titulo
: STRING
;

autor
: STRING
;

editora
: STRING
;

ano : INT
;

catalogo
: 'BGUM'
| 'ALFA'
| 'OUTRO'
;

existencias
: 'LOCAL ' local '(' estados ')'
;

local
: STRING
;

```

```

estados
: estado (',' estado)*
;

estado
: codigoBarras disponib
;

codigoBarras
: ID
;
disponib
: 'ESTANTE' {estante++;}
| 'PERMANENTE' {permanentes++;}
| 'EMPRESTADO' {reservados++;} dataDev
;

dataDev
: ano '-' mes '-' dia
;
mes : INT;
dia : INT;

ID : ('a'..'z'|'A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'0'..'9'|'_')*
;

INT : '0'..'9'+
;

FLOAT
: ('0'..'9')+ '.' ('0'..'9')* EXPONENT?
| '.' ('0'..'9')+ EXPONENT?
| ('0'..'9')+ EXPONENT
;

WS : ( ' '
| '\t'
| '\r'
| '\n'
) {$channel=HIDDEN;}
;

STRING
: '"' ( ESC_SEQ | ~('\\"|'\"') ) * '"'
;

fragment
EXPONENT : ('e'|'E') ('+'|'-')? ('0'..'9')+ ;

fragment
HEX_DIGIT : ('0'..'9'|'a'..'f'|'A'..'F') ;

fragment

```



```

ESC_SEQ
:   '\\' ('b'|'t'|'n'|'f'|'r'|'\"'|'\''|'\\')
|   UNICODE_ESC
|   OCTAL_ESC
;

fragment
OCTAL_ESC
:   '\\' ('0'..'3') ('0'..'7') ('0'..'7')
|   '\\' ('0'..'7') ('0'..'7')
|   '\\' ('0'..'7')
;

fragment
UNICODE_ESC
:   '\\' 'u' HEX_DIGIT HEX_DIGIT HEX_DIGIT HEX_DIGIT
;

```

## 5 Conclusão

...

## A Imagens

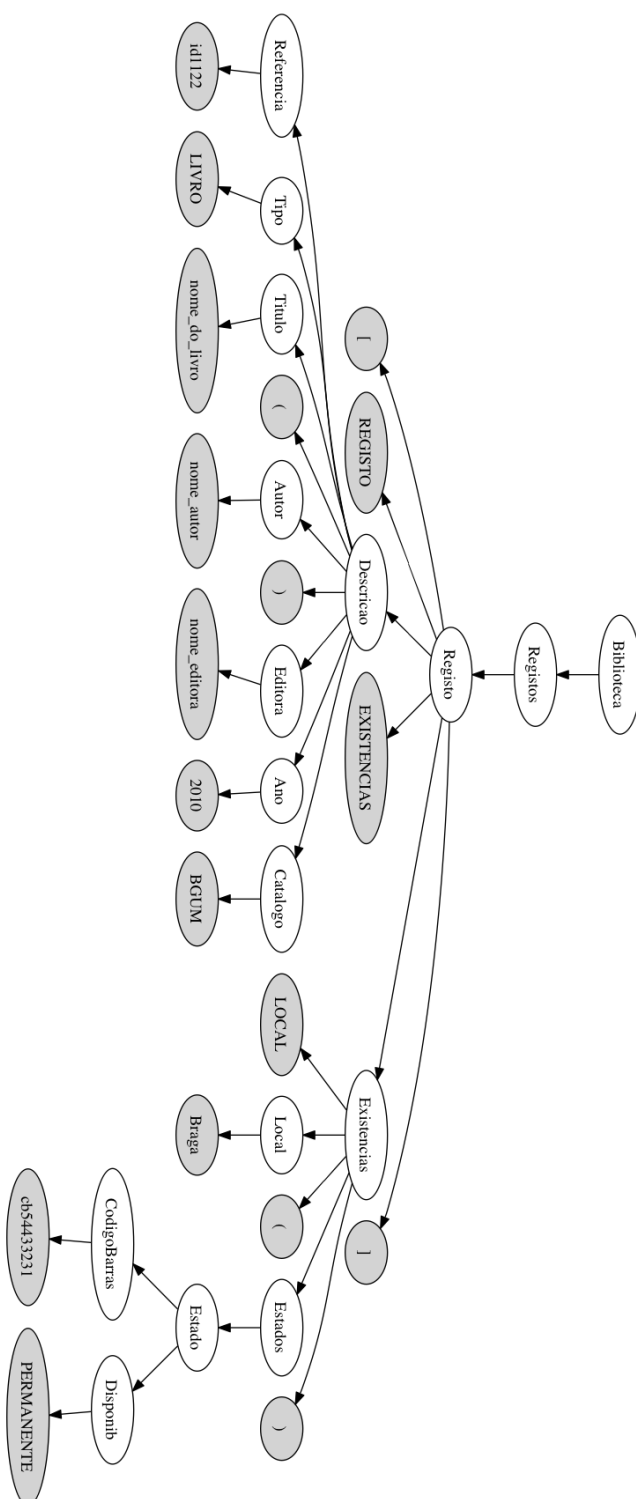


Figura 4: Árvore de Derivação