

Universidade do Minho

Engenharia Gramatical
Ficha de Avaliação N^o1

Engenharia de Linguagens 13/14

André Santos pg25329
Daniel Carvalho a61008
Ricardo Branco pg25339

Conteúdo

1	Introdução	3
2	Respostas	5
2.1	Alínea A	5
2.2	Alínea B	5
2.3	Alínea C	6
2.4	Alínea D	6
2.5	Alínea E	7
2.6	Alínea F	7
2.7	Alínea G	8
A	Gramática Independente de Contexto - AnTLR	9
B	Gramática de Atributos - AnTLR	11
C	Árvores de Derivação	14
D	Parser Recursivo-Descendente	16

1 Introdução

A gramática independente de contexto, GIC, abaixo apresentada, define uma linguagem específica para descrever os livros e CDs disponíveis numa Biblioteca e o estado associado a cada um (livre ou emprestado), à semelhança do que acontece nas bibliotecas da UM. O Símbolo Inicial é Biblioteca, os Símbolos Terminais são escritos em maiúsculas (pseudo-terminais) ou em maiúscula entre apostrofes (palavras-reservadas e sinais de pontuação), e a string nula é denotada por &; os restantes (sempre em minúsculas) serão os Símbolos Não-Terminais. Neste contexto e após analisar a GIC dada, torna-se necessário responder a algumas questões.

```
p0: Biblioteca --> Registos
p1: Registos --> Registo
p2:      | Registos ',' Registo
p3: Registo --> '[' REGISTO Descricao EXISTENCIAS Existencias ']'
p4: Descricao --> Referencia ':' Catalogo '-' Tipo '-' Titulo '(' Autor ')'
Editora '-' Ano
p5: Referencia --> id
p6: Tipo --> LIVRO
p7:      | CDROM
p8:      | OUTRO
p9: Titulo --> string
p10: Autor --> string
p11: Editora --> string
p12: Ano --> num
p13: Catalogo --> BGUM
p14:      | BAUM
p15:      | OUTRO
p19: Existencias --> Estado
p20:      | Existencias ',' Estado
p21: Estado --> CodigoBarras Disponib
p22: CodigoBarras --> id
p23: Disponib --> ESTANTE
p24:      | PERMANENTE
p25:      | EMPRESTADO DataDev
p26: DataDev --> Ano '-' Mes '-' Dia
p27: Mes --> num
p28: Dia --> num
```

- Escreva uma frase válida da linguagem gerada pela GIC dada, mostrando a respectiva Árvore de Derivação.
- Altere a gramática de modo a permitir que cada livro tenha mais de um Autor.
- O par de produções p1/p2 define uma lista com recursividade à esquerda. Altere esse par para usar recursividade à direita e mostre, através das árvores de derivação, a diferença entre ambos os esquemas iterativos.
- Escreva as funções de um parser RD-puro (recursivo-descendente) para reconhecer o Símbolo Existencias e seus derivados, isto é, os símbolos que apareçam no lado direito das produções.
- Construa o estado inicial do autómato LR(0) para gramática dada e os estados que dele saem.
- No sentido de começar a avaliar a qualidade desta gramática calcule algumas métricas de tamanho sobre ela: número de símbolos de cada tipo; número de produções; número de produções unitárias; comprimento médio e máximo dos lados direitos das produções; ...
- Transforme a GIC dada numa gramática de atributos, GA, reconhecível pelo AnTLR, para:
 - calcular e imprimir: o número de registos; e o número de livros existentes para cada registo.

- calcular o número total de livros com estado PERMANENTE.
- identificar e listar por ordem alfabética os títulos dos livros.
- verificar se não existem registos com a mesma referência.

2 Respostas

2.1 Alínea A

Uma frase válida da linguagem gerada pela GIC dada no enunciado do problema é:

```
[
REGISTO r1 :
BAUM - LIVRO - "Game Of Thrones"
("George R.R. Martin") "Bang" - 2007
EXISTENCIAS
baum_gt_28 PERMANENTE ,
bgum_gt_62 EMPRESTADO 2014-5-10
],
[
REGISTO r2 :
BGUM - CDROM - "Abbey Road"
("Beatles") "Apple Records" - 1969
EXISTENCIAS
bgum_arbeat_85 PERMANENTE
]
```

Na figura 1 pode-se ver como é a árvore de derivação da frase apresentada.

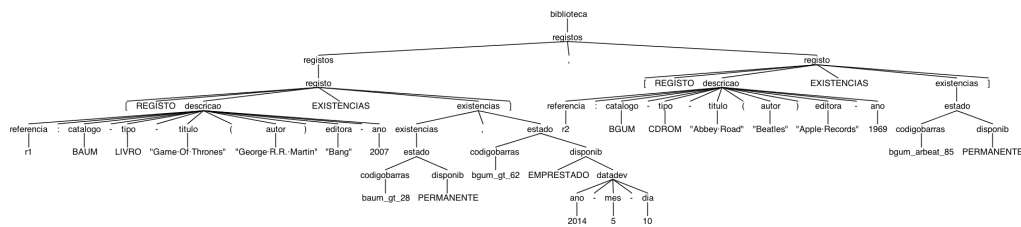


Figura 1: Árvore de Derivação

2.2 Alínea B

De forma a permitir que um livro tenha mais de que um autor, foram feitas as seguintes alterações:

Antes

Descricao --> Referencia ':' Catalogo '-' Tipo '-' Titulo '(' Autores ')' Editora '-' Ano

Depois

Descricao --> Referencia ':' Catalogo '-' Tipo '-' Titulo '(' Autor ')' Editora '-' Ano

Autores --> Autor
| Autores ',' Autor

Com esta nova gramática, frases como a que seguem tornam-se válidas.

```
[
REGISTO r1 :
BAUM - LIVRO - "Uma Aventura Na Serra da Estrela"
("Isabel Alçada","Ana Maria Magalhães") "Caminho" - 2010
```

EXISTENCIAS

```
baum_avent_28 Estante ,
bgum_avent_62 EMPRESTADO 2014-5-10
]
```

Quanto a árvore de derivação, está é representada da seguinte maneira:

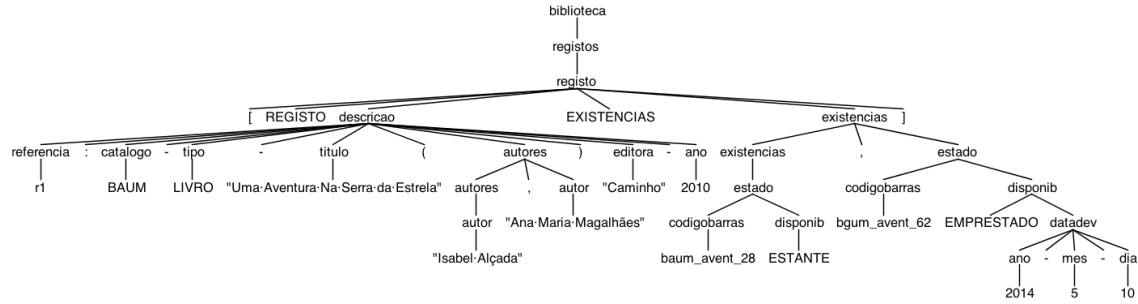


Figura 2: Árvore de Derivação para uma GIC que suporta múltiplos autores

2.3 Alínea C

De forma a permitir que o par de produções **p1/p2** definam uma lista com recursividade à direita, alterou-se a GIC da seguinte forma:

```
Registos --> Registro
          | Registro ',' Registos
```

Essa alteração implicou mudanças na árvore de derivação. Usando a frase usada na Secção 2.1 obteve-se a árvore de derivação representada na figura 1 como sendo exemplo de uma lista com recursividade à esquerda. Numa lista com recursividade à direita a árvore de derivação seria a seguinte:

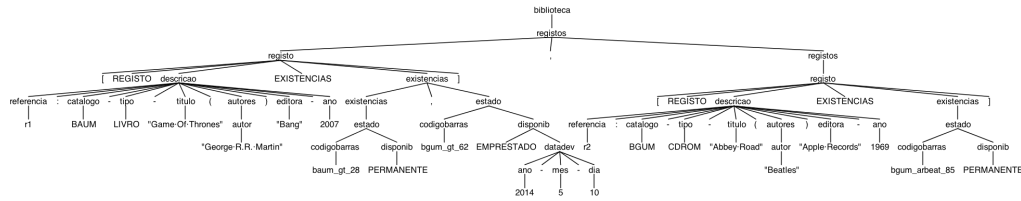


Figura 3: Árvore de Derivação para listas com recursividade à direita

Comparando as duas árvores conclui-se que no caso da gramática constituída por listas com recursividade à esquerda, a árvore de derivação tende a crescer para o lado esquerdo. No caso da gramática constituída por listas com recursividade à direita, a árvore de derivação tende a crescer para o lado direito.

2.4 Alínea D

No Apêndice D estão as funções respetivas a um parser RD-puro(recursivo-descendente), com o objetivo de reconhecer o Símbolo *Existencias*.

No caso dos símbolos não-terminais usa-se a função *void rec_N(s)*, em que *N* é o símbolo não-terminal que se quer reconhecer e *s* o próximo símbolo a analisar. O objetivo é verificar para cada produção *p* de *N* se *s* faz parte do *lookahead* de *p(la(p))*, caso não acontece é retornado um erro.

No caso dos simbolo terminais usa-se a função *void rec_T(s)*, em que *T* é o simbolo terminal que se quer reconhecer e *s* o próximo simbolo a analisar. O objetivo é verificar *s* é igual ao simbolo terminal *T* ou se faz *match* com *T* no caso dos simbolos terminais variáveis *match_T(s)*. Caso *s* seja reconhecido, este passa a ter o valor do próximo símbolo a analisar, caso contrário é retornado um erro.

2.5 Alínea E

O estado inicial do autômato LR(0) para a GIC apresentada é:

```
[Z -> . Biblioteca $]
-----
[Biblioteca -> . Registo]
[Registos -> . Registos]
[Registos -> . Registos ', ' Registo]
[Registo -> . '[' REGISTO Descricao EXISTENCIAS Existencias '']
```

2.6 Alínea F

Na tabela 2.6 mostra-se as medidas que permitem avaliar a GIC apresentada.

Métrica	Medida	Observações
#T	20	$= 3(TV) + 10(PR) + 7(Sin)$
#N	18	
#P	26	
#PU	18	$S \not\subseteq PU$ portanto $p0 \notin PU$
§RHS	2,038	53/26
§RHS-Max	13	p4

Tabela 1: Métricas sobre a GIC

Legenda:

#T N° de Símbolos Terminais

TV Terminais Variáveis

PR Palavras Reservadas

Sin Sinais de Pontuação

#N Símbolos Não-Terminais

#P N° de Produções

#PU N° de Produções Unitárias

§RHS Comprimento médio do lado direito das produções (*Right Hand Size*)

§RHS-Max Comprimento máximo do lado direito das produções

2.7 Alínea G

No Apêndice B está presente a gramática de atributos reconhecidas. Respectivamente à mesma gramática optou-se por criar duas classes auxiliares *Registo* e *Registos* com o objetivo de tornar a gramática mais legível. Os atributos herdados foram escritos na forma *in_** e os atributos sintetizados foram escritos na forma *out_**.

A Gramática Independente de Contexto - AnTLR

```
grammar gic_fa1_a;

biblioteca : registros ;

registros  : registo
           | registros ',' registo
           ;

registo    : '[' REGISTO descricao EXISTENCIAS existencias ']' ;

descricao  : referencia ':' catalogo '-' tipo '-' titulo '(' autor ')' editora '-'
ano ;

referencia : ID ;

tipo       : LIVRO
           | CDROM
           | OUTRO
           ;

titulo     : STRING ;

autor      : STRING ;

editora    : STRING ;

ano        : NUM ;

catalogo   : BGUM
           | BAUM
           | OUTRO
           ;

existencias : estado
            | existencias ',' estado
            ;

estado      : codigobarras disponib ;

codigobarras: ID ;

disponib    : Estante
            | Permanente
            | EMPRESTADO datadev
            ;

datadev     : ano '-' mes '-' dia ;

mes         : NUM ;

dia         : NUM ;
```

```

REGISTO      : 'REGISTO' ;

EXISTENCIAS  : 'EXISTENCIAS' ;

LIVRO        : 'LIVRO' ;

CDROM        : 'CDROM' ;

OUTRO        : 'OUTRO' ;

BGUM         : 'BGUM' ;

BAUM         : 'BAUM' ;

ESTANTE      : 'ESTANTE' ;

PERMANENTE   : 'PERMANENTE' ;

EMPRESTADO   : 'EMPRESTADO' ;

ID           : [a-z][a-z0-9_]* ;

STRING       : '"' ( '\\"' | . ) * ? '"' ;

NUM          : [0-9]+ ;

Sep          : ('\\r'? '\\n' | ' ' | '\\t')+ -> skip;

```

B Gramática de Atributos - AnTLR

Gramática usada na alinea g e explicada na secção 2.7.

```
grammar ga_fa1_g;

@header{import java.util.*;}

@members{class Registo {
    String ref;
    int nrLivros;

    public Registo(){
        nrLivros = 0;
    }
}

class Registos {
    HashMap<String,Registo> registos ;
    TreeSet<String> livros;
    int permanentes;

    public Registos(){
        registos = new HashMap<String,Registo>();
        livros = new TreeSet<String>();
        permanentes = 0;
    }
}

biblioteca : r=registos
{System.out.println("Nº de Registos: "+$r.out_registos.registos.size());
 System.out.println("Nº de Livros com o estado permanente: "+$r.out_registos.permanentes);
 System.out.println("Lista de livros:");
 for(String s : $r.out_registos.livros){
     System.out.println("\t"+s);
 }
}
;

registos returns [Registos out_registos]
: r = registo
{ $out_registos = new Registos();
  $out_registos.registos.put($r.out_registo.ref,$r.out_registo);
  if($r.out_isLivro){
      $out_registos.livros.add($r.out_titulo);
      $out_registos.permanentes = $r.out_NPermanente;
  }
  System.out.println("Foram inseridos "+$r.out_registo.nrLivros+" livros");
}
| r1 = registos ',' r2 =
  registo
{ $out_registos = $r1.out_registos;
  Registo registo_antigo = $out_registos.registos.put($r2.out_registo.ref,$r2.out_registo);
  if(registo_antigo != null){
```

```

        System.err.println("ERRO: Já existe um registro com a referencia
"+registro_antigo.ref);
    }else{
        if($r2.out_isLivro){
            $out_registros.livros.add($r.out_titulo);
            $out_registros.permanentes = $r2.out_NPermanente;
        }
        System.out.println("Foram inseridos "+$r2.out_registro.nrLivros+"
livros");
    }
}
;

registro    returns [Registo out_registro, boolean out_isLivro, String out_titulo,
int out_NPermanente]
@init      {$out_registro = new Registo();}
: '[' REGISTO d = descricao EXISTENCIAS e = existencias ']'
{
    $out_registro.ref = $d.out_referencia;
    if($d.out_isLivro){
        $out_registro.nrLivros = $e.out_quantidade;
    }
    $out_NPermanente = $e.out_NPermanente;
    $out_isLivro = $d.out_isLivro;
    $out_titulo = $d.out_titulo;
}
;

descricao  returns [String out_referencia, boolean out_isLivro,String out_titulo]
: r = referencia {$out_referencia = $r.text;}':' catalogo '-' t1 = tipo
{$out_isLivro = $t1.out_isLivro;} '-' t2 = titulo {$out_titulo = $t2.text;} '(' autor
')' editora '-' ano ;

referencia : ID ;

tipo       returns [boolean out_isLivro]
: LIVRO {$out_isLivro = true;}
| CDROM {$out_isLivro = false;}
| OUTRO {$out_isLivro = false;}
;

titulo     : STRING ;

autor      : STRING ;

editora    : STRING ;

ano        : NUM ;

catalogo   : BGUM
| BAUM
| OUTRO
;

existencias returns [int out_quantidade,int out_NPermanente]

```

```

: e = estado {$out_quantidade=1;
               $out_NPermanente=$e.out_NPermanente;
            }
| e1 = existencias
  ', ' e2 = estado {$out_quantidade=$e1.out_quantidade + 1;
                   $out_NPermanente=$e1.out_NPermanente + $e2.out_NPermanente;
                }
;

estado      returns [int out_NPermanente]
: codigobarras d = disponib {$out_NPermanente = $d.out_NPermanente;}
;

codigobarras: ID ;

disponib    returns [int out_NPermanente]
: Estante {$out_NPermanente = 0;}
| Permanente {$out_NPermanente = 1;}
| Emprestado datadev {$out_NPermanente = 0;}
;

datadev     : ano '-' mes '-' dia ;

mes         : NUM ;

dia         : NUM ;

REGISTO     : 'REGISTO' ;

EXISTENCIAS : 'EXISTENCIAS' ;

LIVRO       : 'LIVRO' ;

CDROM       : 'CDROM' ;

OUTRO       : 'OUTRO' ;

BGUM        : 'BGUM' ;

BAUM        : 'BAUM' ;

ESTANTE     : 'ESTANTE' ;

PERMANENTE  : 'PERMANENTE' ;

EMPRESTADO  : 'EMPRESTADO' ;

ID          : [a-z][a-z0-9_]* ;

STRING      : '"' ( '\\"' | . ) * ? '"' ;

NUM         : [0-9]+ ;

Sep         : ( '\\r'? '\\n' | ' ' | '\\t' ) + -> skip;

```

C Árvores de Derivação

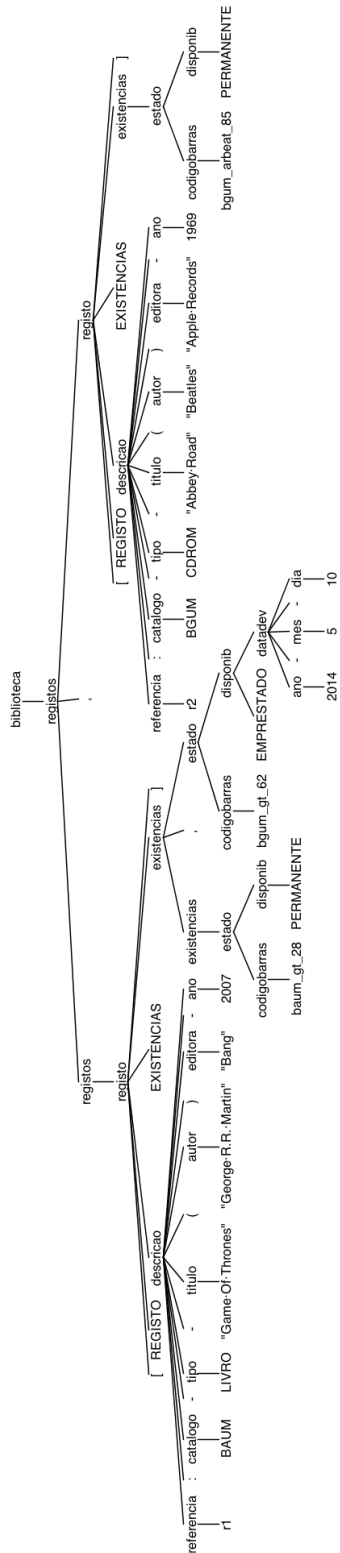


Figura 4: Árvore de Derivação usando listas com recursividade à esquerda

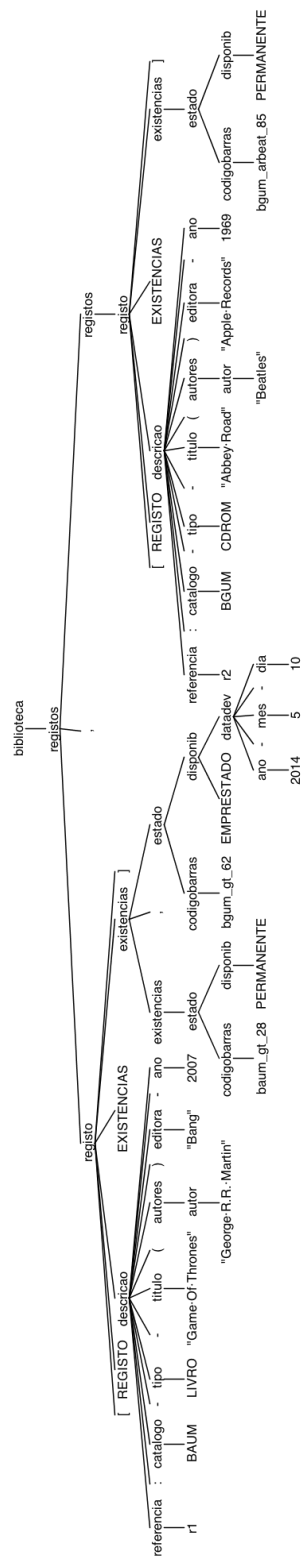


Figura 5: Arvore de Derivação usando listas com recursividade à direita

D Parser Recursivo-Descendente

Funções de um parser RD-puro, com respetiva explicação na secção 2.4.

```
void rec_Existencias(s){
  if(s in la(p1)){
    rec_Estado(s);
  }else if(s in la(p2)){
    rec_Existencias(s);
    rec_Virgula(s);
    rec_Estado(s);
  }else{
    erro();
  }
}
```

```
void rec_Estado(s){
  if(s in la(p1)){
    rec_Codigobarras(s);
    rec_Disponib(s);
  }else{
    erro();
  }
}
```

```
void rec_Codigobarras(s){
  rec_ID(s);
}
```

```
void rec_ID(s){
  if(matchID(s)){
    s = daSimbolo();
  }else{
    erro();
  }
}
```

```
void rec_Disponib(s){
  switch(s){
    case Estante :
      s = daSimbolo();
      break;
    case Permanente :
      s = daSimbolo();
      break;
    case Emprestado :
      s = daSimbolo();
      rec_Datadev(s);
      break;
    default:
      erro();
  }
}
```



```

void rec_Datadev(s){
if(s in la(p1)){
rec_Ano(s);
rec_Hifen(s);
rec_Mes(s);
rec_Hifen(s);
rec_Dia(s);
}else{
erro();
}
}

```

```

void rec_Ano(s){
if(matchNUM(s)){
s = daSimbolo();
}else{
erro();
}
}

```

```

void rec_Hifen(s){
if(s == '-'){
s=daSimbolo();
}
}

```

```

void rec_Mes(s){
rec_NUM(s);
}

```

```

void rec_Dia(s){
rec_NUM(s)
}

```

```

void rec_NUM(s){
if(s == '-'){
s = daSimbolo();
}else{
erro();
}
}

```

```

void rec_Virgula(s){
if(s == ','){
s = daSimbolo(s);
}
}

```