Engenharia de Linguagens

Universidade do Minho, LEI

Ano lectivo 2010/2011

Engenharia Gramatical - Exercício 3

José Pedro Silva - pg17628 Mário Ulisses Costa - pg15817 Pedro Faria - pg17684

8 de Janeiro de 2011

Resumo

Descrição da solução à proposta para os problemas do dia 06-12-2010

Conteúdo

1 Problema **1**

2 Solução **2**
2.1 Alínea a) . . . 2
2.2 Alínea b) . . . 2
2.3 Alínea c) . . . 3
2.4 Alínea d) . . . 3

3 AntLR **4**
3.1 Exemplo de input . . . 4
3.2 Tokens . . . 4
3.3 Gramática sem semântica . . . 5
3.4 Semântica . . . 6

4 Anexo **8**
4.1 Código da alínea a) . . . 8
4.2 Código da alínea b) . . . 9
4.3 Código da alínea c) . . . 10
4.4 Código da alínea d) . . . 11
4.5 AntLR . . . 12

1 Problema

Considere a linguagem para descrever uma Factura. Sabe-se que uma Factura é composta por um cabeçalho e um corpo, e este é composto por um ou mais movimentos.

A GIC abaixo define formalmente uma primeira versão da linguagem Factura, de acordo com a descrição acima:

```
1       T = { id, str, num}
2       N = { Lisp, SExp, SExplist }
3       S = Lisp
4       P = {
5             p1: Factura --> Cabec Corpo
6             p2: Cabec --> IdFact IdE IdR
7             p3: IdFact --> NumFact
8             p4: NumFact --> id
9             p5: IdE --> Nome NIF Morada NIB
10            p6: IdR --> Nome NIF Morada
11            p7: Nome --> str
12            p8: NIF --> str
13            p9: Morada --> str
14            p10: NIB --> str
15            p11: Corpo --> ...
16          }
```

Pede-se então que escreva uma Gramática de Atributos, GA, para

- a) calcular o total por linha e total geral.
- b) estender a linguagem original para permitir mais do que uma factura (calculando os mesmos totais).
- c) modificar a linguagem de modo a suportar inicialmente a descrição do stock (referência, descrição, preço unitário e quantidade em stock); neste caso, cada linha só terá a referência e a quantidade vendida.
- d) estender a semântica da nova linguagem de modo a também actualizar o stock.

2 Solução

As subsecções que se seguem referem-se à resolução de cada alínea do capítulo anterior.

2.1 Alínea a)

A solução para a primeira alínea passa por calcular o total por linha na produção linha, multiplicando a quantidade pelo preço unitário.

```
1 Linha -> Ref Desc Qtd PU {Linha.totalLinha = Qtd.quantidade * PU.pu;}
```

Para calcular o montante total da factura, temos que somar na produção Linhas, o preço de cada linha com o preço já calculado das restantes linhas. No caso de só haver uma linha, o montante total é igual ao montante da linha.

```
1 Linhas -> Linha {Linhas.total = Linhas.totalLinha;}
2 Linhas -> Linha Linhas {Linhas0.total = Linha1.totalLinha + Linhas2.total;}
```

2.2 Alínea b)

Para a linguagem passar a permitir multiplas facturas, temos de acrescentar as seguintes produções à gramática:

```
1 LivroF -> Facturas
2
3 Facturas -> Factura
4
5 Facturas -> Factura Facturas
```

E agora *LivroF* toma o lugar de símbolo inícial, que pertencia a *Factura*.

2.3 Alínea c)

Nesta alínea a descrição e preço unitário desaparecem de cada linha da factura, porque passa a existir um stock, que contém as informações de todos os produtos.

```
1 Inicial -> Stock LivroF { LivroF.stock = Stock.stock; }
2
3 Stock -> Produtos
4
5 Produtos -> Produto
6
7 Produtos -> Produto Produtos
8
9 Produto -> Ref Desc Qtd PU
```

As informações lidas no stock são depois herdadas pelo livro de facturas (*LivroF*), depois pelas Facturas, e assim sucessivamente atée chegar à Linha. Na linha é calculado o preço total da linha através da quantidade e do preço unitário, tal como anteriormente. A diferença é que agora o preço unitário é acessido através do stock.

```
1 Linha -> Ref Qtd {
2 int pu = Linha.stock.getPU(Ref.num);
3 Linha.totalLinha = Qtd.quantidade * pu;
4 imprime Linha.totalLinha;
5 }
```

2.4 Alínea d)

De modo a actualizar o stock, na produção *Linha* retiramos ao produto referenciado por *Ref*, a quantidade vendida.

```
1 Linha -> Ref Qtd {
2 int pu = stock.getPU(Ref.num);
3   Linha.totalLinha = Qtd.quantidade * pu;
4 imprime Linha.totalLinha;
5
6 int qt = Linha.stock.getQuantidade(Ref.num);
7 Linha.stock.setQuantidade(qt-Qtd.quantidade);
8 }
```

3 AntLR

Nesta secção será apresentado um estudo sobre a ferramenta *AntLR* usando como caso de estudo o problema e a solução proposta nas secções anteriores. Assim, começar-se-à por dar um exemplo de input da linguagem, a gramática desenvolvida e a semântica associada.

3.1 Exemplo de input

O texto seguinte representa um exemplo de frases no qual a linguagem foi baseada. Cada frase será constituída por uma factura, em que esta contém um *id* associado, dados sobre o emissor e receptor, e uma lista de produtos.

```
1  @idfact:ajf84R5dd5
2          @ide:
3                  @nome:Pedro
4                  @nif:1234512345
5                  @morada:Av. D. Joao IV 175
6                  @nib:13351342
7          @idr:
8                  @nome:Maria
9                  @nif:1234612345
10                 @morada:Urbanizacao Salgueiral
11         @corpo
12                 @p:
13                 g73cc9d90
14                 2
15                 10
16                 descricao ..
17                 @p:
18                 g23ac9gg0
19                 3
20                 1
21                 descricao2 ..
22         @fimcorpo
23
24 @idfact:zf8XXu80e
25         @ide:
26                 @nome:Josuef
27                 @nif:1278945621
28                 @morada:Avenida dos Aliados
29                 @nib:13377597
30         @idr:
31                 @nome:Maria
32                 @nif:1234612345
33                 @morada:Urbanizacao Salgueiral
34         @corpo
35                 @p:
36                 g73cc9d90
37                 2
38                 50
39                 descricao3 ..
40         @fimcorpo
```

Como objectivo final, pretendia-se implementar um *pretty print* da informação mais o número de produtos e custo totais.

3.2 Tokens

Como se pode ver no exemplo anterior, existem certas linhas com *tags* no começo. Essas linhas serviram apenas como auxílio para *debug* da linguagem e para explorar a definição de *tokens* da ferramenta. Sendo assim, desenvolveram-se as seguintes *tags*:

```
1
2  tokens{
3          TAGidfact = '@idfact:';
4          TAGide = '@ide:';
```

```
5           TAGnome = '@nome:';
6           TAGnif = '@nif:';
7           TAGmorada = '@morada:';
8           TAGnib = '@nib:';
9           TAGidr = '@idr:';
10          TAGcorpo = '@corpo';
11          TAGfimcorpo = '@fimcorpo';
12          TAGproduto = '@p:';
13  }
```

3.3 Gramática sem semântica

De seguida, apresenta-se a gramática criada com a *syntax* do *AntLR*. A explicação segue a que foi dada nas secções anteriores, com a excepção de não ter sido criado um Stock para esta linguagem. Como se pode ver também, foram adicionados os tokens referidos na subsecção anterior à gramática.

```
1
2   livrofact       :       (factura)+
3                   ;
4
5   factura         :       cabec   corpo
6                   ;
7
8   cabec           :       idfact  ide     idr
9                   ;
10
11  corpo           :       TAGcorpo        (produto)+      TAGfimcorpo
12                  ;
13
14  produto         :       (TAGproduto     referencia      precouni        quantidade
        descricao)
15                  ;
16
17  referencia      :       ID
18                  ;
19
20  descricao       :       STRINGPLUS
21                  ;
22
23  quantidade      :       NUM
24                  ;
25
26  precouni        :       NUM
27                  ;
28
29  idfact          :       numfact
30                  ;
31
32  numfact         :       TAGidfact       ID
33                  ;
34
35  ide             :       TAGide  nome    nif     morada  nib
36                  ;
37
38  idr             :       TAGidr  nome    nif     morada
39                  ;
40
41  nome            :       TAGnome STRING
42                  ;
43
44  nif             :       TAGnif  NUM
45                  ;
46
47  morada          :       TAGmorada       STRINGPLUS
48                  ;
49
50  nib             :       TAGnib  NUM
51                  ;
52
```

```
53 NUM              :       ('0'..'9')+
54                  ;
55
56 STRING           :       ('a'..'z'|'A'..'Z')+
57                  ;
58
59 ID               :       ('a'..'z'|'A'..'Z'|'0'..'9')+
60                  ;
61
62 STRINGPLUS       :       ('a'..'z'|'A'..'Z'|'0'..'9'|'\.'|' ')+
63                  ;
64
65 NS               :       (' ' | '\t' | '\n' | 'r') { skip(); }
66                  ;
```

Em relação às listas que possam surgir nesta linguagem (de facturas e/ou produtos) foram definidas com uma notação, do estilo expressões regulares, que esta ferramenta permite.

3.4 Semântica

Para desenvolvimento dos problemas decidiu-se tirar vantagem do *AntLR* ser perfeitamente compatível com *Java*. Assim, um `livrofact` seria um *ArrayList* de objectos `Factura`. Cada objecto `Factura` teria 4 variáveis: uma `String` id; um objecto `Emissor`; um objecto `Receptor`; e uma *ArrayList* de objectos `Produto`.

Os objectos `Receptor`, `Emissor` e `Produto` apenas teriam informação de tipos primitivos.

Assim, no começo da gramática, é inicializado o *ArrayList* de `Factura`, como se pode ver na linha 2 do pedaço de código seguinte:

```
1 livrofact
2 @init{ArrayList<Factura> livrof_in = new ArrayList<Factura>();}
3         :       (factura {livrof_in.add(\$factura.factura_out);})+ {System.out.println("
              Livro de Facturas\n");
4                                                   for(Factura f : livrof_in){System.out.
                                                        println(f.toString());}}
5         ;
```

A cada factura lida, seria guardado um atributo sintetizado do tipo `Factura` (linha 3). E no final seria imprimida a informação (linha 4).

O pedaço de código seguinte, diz respeito à criação de facturas. A cada início da produção `factura`, é inicializado um objecto `Factura` (linha 3). Esse objecto criado será passado como um atributo herdado para as produções `cabec` e `corpo` (linhas 4 e 5). Na produção `cabec`, o atributo herdado `factura_in` terá a variável *id* alterada pelo atributo sintetizado vindo do nodo `idfact` (linha 10). Para os nodos `ide` e `idr` serão mandados as variáveis `emi` e `rec`, os objectos `Emissor` e `Receptor` respectivamente (linhas 11 e 12). O processamento do corpo da factura ocorre de maneira semelhante ao da criação da linha de facturas, a cada ocorrência de um nodo `produto` seria adicionado à lista de produtos um atributo sintetizado do tipo `Produto`.

```
1  factura
2 returns[Factura factura_out]
3 @init{Factura factura_in = new Factura();}
4         :       cabec[factura_in] {  }
5                 corpo[factura_in] { factura_out = factura_in; }
6         ;
7
8 cabec [Factura factura_in]
9
10        :       idfact {$factura_in.id = $idfact.id_out;}
11                ide[factura_in.emi]
```

```
12                  idr[factura_in.rec]
13          ;
14
15  corpo [Factura factura_in]
16          :       TAGcorpo
17                  (produto{$factura_in.addProduto($produto.prod_out);})+
18                  TAGfimcorpo
19          ;
```

Tanto o Emissor como o Receptor e os Produto têm processos de criação semelhantes. Não serão precisos de ser inicializados objectos dos tipos Emissor e Receptor, pois o seu processo ocorre em simultâneo com a inicialização do objecto `Factura`. Assim, cada variável do objecto herdado `Emissor` fica com os valores sintetizados correspondentes (linhas 3, 4, 5 e 6).

```
1   ide [Emissor emissor_in]
2           :       TAGide
3                   nome    {$emissor_in.nome = $nome.nome_out;}
4                   nif     {$emissor_in.nif = $nif.nif_out;}
5                   morada  {$emissor_in.morada = $morada.morada_out;}
6                   nib     {$emissor_in.nib = $nib.nib_out;}
7           ;
```

Os nodos `nome`, `nif`, `morada` e `nib` são bastante semelhantes, apenas retornam o valor para os nodos superiores respectivos, como por exemplo:

```
1    nib
2   returns[int nib_out]
3           :       TAGnib  NUM{$nib_out = Integer.parseInt($NUM.text);}
4           ;
```

Todo o restante código semântico será adicionado na seção `Anexo`.

4 Anexo

4.1 Código da alínea a)

```
synthesided attribute quantidade occurs on Qtd;
synthesided attribute pu occurs on PU;
synthesided attribute totalLinha occurs on Linha;
synthesided attribute total occurs on Linhas,CorpoF,Factura,Facturas;


Factura -> CabecaF CorpoF {imprime(CorpoF.total);}

CabecaF -> IdFact IdE IdR

IdFact -> NumFact

NumFact -> id

IdE -> Nome NIF Mor NIB

IdR -> Nome NIF Mor

Nome -> string

NIF -> num

Mor -> string

NIB -> num

CorpoF -> Linhas {CorpoF.total = Linhas.total;}

Linhas -> Linha {Linhas.total = Linhas.totalLinha;}

Linhas -> Linha Linhas {Linhas0.total = Linha1.totalLinha + Linhas2.total;}

Linha -> Ref Desc Qtd PU {Linha.totalLinha = Qtd.quantidade * PU.pu;}

Ref -> NumProd

NumPro -> id

Desc -> string

Qtd -> num { Qtd.quantidade = num.lexeme;}

PU -> num { PU.quantidade = num.lexeme;}
```

4.2 Código da alínea b)

```
synthesided attribute quantidade occurs on Qtd;
synthesided attribute pu occurs on PU;

synthesided attribute totalLinha occurs on Linha;

synthesided attribute total occurs on Linhas,CorpoF,Factura;



LivroF -> Facturas

Facturas -> Factura

Facturas -> Factura Facturas

Factura -> CabecaF CorpoF {imprime(CorpoF.total);}

CabecaF -> IdFact IdE IdR

IdFact -> NumFact

NumFact -> id

IdE -> Nome NIF Mor NIB

IdR -> Nome NIF Mor

Nome -> string

NIF -> num

Mor -> string

NIB -> num

CorpoF -> Linhas {CorpoF.total = Linhas.total;}

Linhas -> Linha {Linhas.total = Linhas.totalLinha;}

Linhas -> Linha Linhas {Linhas0.total = Linha1.totalLinhas + Linhas2.total;}

Linha -> Ref Desc Qtd PU {Linha.totalLinha = Qtd.quantidade * PU.pu;}

Ref -> NumProd

NumPro -> id

Desc -> string

Qtd -> num { Qtd.quantidade = num.lexeme;}

PU -> num { PU.quantidade = num.lexeme;}
```

4.3 Código da alínea c)

```
synthesided attribute quantidade occurs on Qtd;
synthesided attribute pu occurs on PU;

synthesided attribute totalLinha occurs on Linha;

synthesided attribute total occurs on Linhas,CorpoF,Factura,Facturas;

inherited attribute stock occurs on LivroF,Facturas,Factura,CorpoF,Linhas,Linha;



Inicial -> Stock LivroF { LivroF.stock = Stock.stock; }

Stock -> Produtos

Produtos -> Produto

Produtos -> Produto Produtos

Produto -> Ref Desc Qtd PU

LivroF -> Facturas { Facturas.stock = LivroF.stock;}

Facturas -> Factura {Factura.stock = Facturas.stock;}

Facturas -> Factura Facturas {Factura1.stock = Facturas0.stock; Facturas2.stock = Facturas0.
    stock;}

Factura -> CabecaF CorpoF { imprime(CorpoF.total); CorpoF.stock = Factura.stock;}

CabecaF -> IdFact IdE IdR

IdFact -> NumFact

NumFact -> id

IdE -> Nome NIF Mor NIB

IdR -> Nome NIF Mor

Nome -> string

NIF -> num

Mor -> string

NIB -> num

CorpoF -> Linhas {CorpoF.total = Linhas.total; Linhas.stock = CorpoF.stock;}

Linhas -> Linha {Linhas.total = Linhas.totalLinha; Linha.stock = Linhas.stock;}

Linhas -> Linha Linhas {Linhas0.total = Linha1.totalLinhas + Linhas2.total; Linha1.stock =
    Linhas0.stock; Linhas2.stock = Linhas0.stock; }

Linha -> Ref Qtd { int pu = Linha.stock.getPU(Ref.num);
  Linha.totalLinha = Qtd.quantidade * pu;
imprime Linha.totalLinha;
}

Ref -> NumProd { Ref.num = Numprod.num}

NumPro -> id {NumPro.num = id.lexeme}

Desc -> string

Qtd -> num { Qtd.quantidade = num.lexeme;}

PU -> num { PU.quantidade = num.lexeme;}
```

4.4 Código da alínea d)

```
synthesided attribute quantidade occurs on Qtd;
synthesided attribute pu occurs on PU;

synthesided attribute totalLinha occurs on Linha;

synthesided attribute total occurs on Linhas,CorpoF,Factura;

inherited attribute stock occurs on LivroF,Facturas,Factura,CorpoF,Linhas,Linha;




Inicial -> Stock LivroF { LivroF.stock = Stock.stock; }

LivroF -> Facturas { Facturas.stock = LivroF.stock;}

Facturas -> Factura {Factura.stock = Facturas.stock;}

Facturas -> Factura Facturas {Factura1.stock = Facturas0.stock; Facturas2.stock = Facturas0.
    stock;}

Factura -> CabecaF CorpoF { imprime(CorpoF.total); CorpoF.stock = Factura.stock;}

CabecaF -> IdFact IdE IdR

IdFact -> NumFact

NumFact -> id

IdE -> Nome NIF Mor NIB

IdR -> Nome NIF Mor

Nome -> string

NIF -> num

Mor -> string

NIB -> num

CorpoF -> Linhas {CorpoF.total = Linhas.total; Linhas.stock = CorpoF.stock;}

Linhas -> Linha {Linhas.total = Linhas.totalLinha; Linha.stock = Linhas.stock;}

Linhas -> Linha Linhas {Linhas0.total = Linha1.totalLinhas + Linhas2.total; Linha1.stock =
    Linhas0.stock; Linhas2.stock = Linhas0.stock; }

Linha -> Ref Qtd { int pu = stock.getPU(Ref.num);
  Linha.totalLinha = Qtd.quantidade * pu;
imprime Linha.totalLinha;

int qt = Linha.stock.getQuantidade(Ref.num);
Linha.stock.setQuantidade(qt-Qtd.quantidade);
}

Ref -> NumProd { Ref.num = Numprod.num}

NumPro -> id {NumPro.num = id.lexeme}

Desc -> string

Qtd -> num { Qtd.quantidade = num.lexeme;}

PU -> num { PU.quantidade = num.lexeme;}
```

4.5 AntLR

```
1   grammar livrofact;
2
3   tokens{
4           TAGidfact = '@idfact:';
5           TAGide = '@ide:';
6           TAGnome = '@nome:';
7           TAGnif = '@nif:';
8           TAGmorada = '@morada:';
9           TAGnib = '@nib:';
10          TAGidr = '@idr:';
11          TAGcorpo = '@corpo';
12          TAGfimcorpo = '@fimcorpo';
13          TAGproduto = '@p:';
14  }
15
16  @header{
17          ;
18  }
19
20  @members{
21
22  /*============  Classe Emissor ============= */
23  public class Emissor{
24
25          public String nome;
26          public String morada;
27          public int nib;
28          public int nif;
29
30          public Emissor(){
31                  this.nome = "";
32                  this.morada = "";
33                  this.nib = 0;
34                  this.nif = 0;
35          }
36
37          public Emissor(String name, String street, int enib, int enif){
38                  this.nome = name;
39                  this.morada = street;
40                  this.nib = enib;
41                  this.nif = enif;
42          }
43
44          public String toString(){
45                  return "\nDados Emissor" + "\nNome: " + this.nome + "\nMorada: " + this.
                        morada + "\n";
46                  //return "\nEmissor\n";
47          }
48
49  }
50
51  /*============  Classe Receptor ============= */
52  public class Receptor{
53
54          public String nome;
55          public String morada;
56          public int nif;
57
58          public Receptor(){
59                  this.nome = "";
60                  this.morada = "";
61                  this.nif = 0;
62          }
63
64          public Receptor(String name, String street, int enif){
65                  this.nome = name;
66                  this.morada = street;
67                  this.nif = enif;
68          }
```

```
69
70          public String toString(){
71                  return "\nDados Receptor" + "\nNome: " + this.nome + "\nMorada: " + this.
                        morada + "\n";
72                  //return "\nReceptor\n";
73          }
74
75  }
76
77  /*============  Produto ============= */
78  public class Produto{
79          public String referencia;
80          public String descricao;
81          public int quantidade;
82          public int precouni;
83
84          public Produto(){
85                  this.referencia = "";
86                  this.descricao = "";
87                  this.precouni = 0;
88                  this.quantidade = 0;
89          }
90
91          public Produto(String ref, String desc, int pu, int qtd){
92                  this.referencia = ref;
93                  this.descricao = desc;
94                  this.precouni = pu;
95                  this.quantidade = qtd;
96          }
97
98          public String toString(){
99                  return "Produto -> " + this.referencia + "\nDescricao: " + this.descricao +
                        "\nPreco por unidade: " +
100                 this.precouni + "\nQuantidade" + this.quantidade + "\n";
101         }
102
103         public int precoTotal(){
104                 return this.precouni * this.quantidade;
105         }
106 }
107
108 /*============  Factura ============= */
109 public class Factura{
110         public Emissor emi;
111         public Receptor rec;
112         public String id;
113         public ArrayList<Produto> al;
114
115         public Factura(){
116                 this.id = "";
117                 this.emi = new Emissor();
118                 this.rec = new Receptor();
119                 this.al = new ArrayList<Produto>();
120         }
121
122         public Factura(Emissor e, Receptor r){
123                 this.id = "";
124                 this.emi = new Emissor(e.nome, e.morada, e.nib, e.nif);
125                 this.rec = new Receptor(r.nome, r.morada, r.nif);
126                 this.al = new ArrayList<Produto>();
127         }
128
129         public void addProduto(Produto p){
130                 this.al.add(p);
131         }
132
133         public String toString(){
134                 int custo = 0;
135                 for(Produto p : this.al){
136                         custo += p.precoTotal();
137                 }
```

```
138
139                 return "\nFactura " + this.id + "\n" +
140                         this.emi.toString() +
141                         this.rec.toString() +
142                         "\nTotal de Produtos = " + this.al.size() +
143                         "\nCusto Total = " + custo;
144         }
145 }
146
147 }
148
149 livrofact
150 @init{ArrayList<Factura> livrof_in = new ArrayList<Factura>();}
151         :       (factura {livrof_in.add(\$factura.factura_out);})+ {System.out.println("
                Livro de Facturas\n");
152                                                 for(Factura f : livrof_in){System.out.
                                                        println(f.toString());}}
153         ;
154
155 factura
156 returns[Factura factura_out]
157 @init{Factura factura_in = new Factura();}
158         :       cabec[factura_in] {  }//System.out.println(factura_in.toString());}
159                 corpo[factura_in] { factura_out = factura_in; }
160         ;
161
162 cabec [Factura factura_in]
163
164         :       idfact {\$factura_in.id = \$idfact.id_out;}
165                 ide[factura_in.emi]
166                 idr[factura_in.rec]
167         ;
168
169 corpo [Factura factura_in]
170         :       TAGcorpo
171                 (produto{\$factura_in.addProduto(\$produto.prod_out);})+
172                 TAGfimcorpo
173         ;
174
175 produto
176 returns[Produto prod_out]
177 @init{Produto prod_in = new Produto();}
178         :       (TAGproduto
179                 referencia {prod_in.referencia = \$referencia.id_out;}
180                 precouni {prod_in.precouni = \$precouni.pu_out;}
181                 quantidade {prod_in.quantidade = \$quantidade.qtd_out;}
182                 descricao {prod_in.descricao = \$descricao.descri_out;}) {\$prod_out =
                        prod_in;}
183
184         ;
185
186 referencia
187 returns[String id_out]
188         :       ID{\$id_out = \$ID.text;}
189         ;
190
191 descricao
192 returns[String descri_out]
193         :       STRINGPLUS{\$descri_out = \$STRINGPLUS.text;}
194         ;
195
196 quantidade
197 returns[int qtd_out]
198         :       NUM{\$qtd_out = Integer.parseInt(\$NUM.text);}
199         ;
200
201 precouni
202 returns[int pu_out]
203         :       NUM{\$pu_out = Integer.parseInt(\$NUM.text);}
204         ;
205
```

```
206
207 idfact
208 returns[String id_out]
209         :       numfact {\$id_out = \$numfact.id_out;}
210         ;
211
212 numfact
213 returns[String id_out]
214         :       TAGidfact       ID{\$id_out = \$ID.text;}
215         ;
216
217 ide [Emissor emissor_in]
218
219         :       TAGide
220                 nome    {\$emissor_in.nome = \$nome.nome_out;}
221                 nif     {\$emissor_in.nif = \$nif.nif_out;}
222                 morada  {\$emissor_in.morada = \$morada.morada_out;}
223                 nib     {\$emissor_in.nib = \$nib.nib_out;}
224         ;
225
226 idr [Receptor receptor_in]
227
228         :       TAGidr
229                 nome    {\$receptor_in.nome = \$nome.nome_out;}
230                 nif     {\$receptor_in.nif = \$nif.nif_out;}
231                 morada  {\$receptor_in.morada = \$morada.morada_out;}
232         ;
233
234 nome
235 returns[String nome_out]
236         :       TAGnome STRING{\$nome_out = \$STRING.text;}
237         ;
238
239 nif
240 returns[int nif_out]
241         :       TAGnif  NUM{\$nif_out = Integer.parseInt(\$NUM.text);}
242         ;
243
244 morada
245 returns[String morada_out]
246         :       TAGmorada       STRINGPLUS{\$morada_out = \$STRINGPLUS.text;}
247         ;
248
249 nib
250 returns[int nib_out]
251         :       TAGnib  NUM{\$nib_out = Integer.parseInt(\$NUM.text);}
252         ;
253
254 NUM     :       ('0'..'9')+
255         ;
256
257 STRING  :       ('a'..'z'|'A'..'Z')+
258         ;
259
260 ID      :       ('a'..'z'|'A'..'Z'|'0'..'9')+
261         ;
262
263 STRINGPLUS      :       ('a'..'z'|'A'..'Z'|'0'..'9'|'\.'|' ')+
264         ;
265
266 NS      :       (' ' | '\t' | '\n' | 'r') { skip(); }
267         ;
```