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

11 de Dezembro de 2010

Resumo

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

Conteúdo

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 Anexo

3.1 Código da alínea a)

```
1  synthesided attribute quantidade occurs on Qtd;
2  synthesided attribute pu occurs on PU;
3  synthesided attribute totalLinha occurs on Linha;
4  synthesided attribute total occurs on Linhas,CorpoF,Factura,Facturas;
5
6
7  Factura -> CabecaF CorpoF {imprime(CorpoF.total);}
8
9  CabecaF -> IdFact IdE IdR
10
11 IdFact -> NumFact
12
13 NumFact -> id
14
15 IdE -> Nome NIF Mor NIB
16
17 IdR -> Nome NIF Mor
18
19 Nome -> string
20
21 NIF -> num
22
23 Mor -> string
24
25 NIB -> num
26
27 CorpoF -> Linhas {CorpoF.total = Linhas.total;}
28
29 Linhas -> Linha {Linhas.total = Linhas.totalLinha;}
30
31 Linhas -> Linha Linhas {Linhas0.total = Linha1.totalLinha + Linhas2.total;}
32
33 Linha -> Ref Desc Qtd PU {Linha.totalLinha = Qtd.quantidade * PU.pu;}
34
35 Ref -> NumProd
36
37 NumPro -> id
38
39 Desc -> string
40
41 Qtd -> num  { Qtd.quantidade = num.lexeme;}
42
43 PU -> num       { PU.quantidade = num.lexeme;}
```

3.2 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;}
```

3.3 Código da alínea c)

```
1  synthesided attribute quantidade occurs on Qtd;
2  synthesided attribute pu occurs on PU;
3
4  synthesided attribute totalLinha occurs on Linha;
5
6  synthesided attribute total occurs on Linhas,CorpoF,Factura,Facturas;
7
8  inherited attribute stock occurs on LivroF,Facturas,Factura,CorpoF,Linhas,Linha;
9
10
11
12 Inicial -> Stock LivroF { LivroF.stock = Stock.stock; }
13
14 Stock -> Produtos
15
16 Produtos -> Produto
17
18 Produtos -> Produto Produtos
19
20 Produto -> Ref Desc Qtd PU
21
22 LivroF -> Facturas { Facturas.stock = LivroF.stock;}
23
24 Facturas -> Factura {Factura.stock = Facturas.stock;}
25
26 Facturas -> Factura Facturas {Factura1.stock = Facturas0.stock; Facturas2.stock = Facturas0.
       stock;}
27
28 Factura -> CabecaF CorpoF { imprime(CorpoF.total); CorpoF.stock = Factura.stock;}
29
30 CabecaF -> IdFact IdE IdR
31
32 IdFact -> NumFact
33
34 NumFact -> id
35
36 IdE -> Nome NIF Mor NIB
37
38 IdR -> Nome NIF Mor
39
40 Nome -> string
41
42 NIF -> num
43
44 Mor -> string
45
46 NIB -> num
47
48 CorpoF -> Linhas {CorpoF.total = Linhas.total; Linhas.stock = CorpoF.stock;}
49
50 Linhas -> Linha {Linhas.total = Linhas.totalLinha; Linha.stock = Linhas.stock;}
51
52 Linhas -> Linha Linhas {Linhas0.total = Linha1.totalLinhas + Linhas2.total; Linha1.stock =
       Linhas0.stock; Linhas2.stock = Linhas0.stock; }
53
54 Linha -> Ref Qtd {  int pu = Linha.stock.getPU(Ref.num);
55                                        Linha.totalLinha = Qtd.quantidade * pu;
56                                        imprime Linha.totalLinha;
57                                                }
58
59 Ref -> NumProd { Ref.num = Numprod.num}
60
61 NumPro -> id {NumPro.num = id.lexeme}
62
63 Desc -> string
64
65 Qtd -> num  { Qtd.quantidade = num.lexeme;}
66
67 PU -> num       { PU.quantidade = num.lexeme;}
```

3.4 Código da alínea d)

```
1  synthesided attribute quantidade occurs on Qtd;
2  synthesided attribute pu occurs on PU;
3
4  synthesided attribute totalLinha occurs on Linha;
5
6  synthesided attribute total occurs on Linhas,CorpoF,Factura;
7
8  inherited attribute stock occurs on LivroF,Facturas,Factura,CorpoF,Linhas,Linha;
9
10
11
12
13 Inicial -> Stock LivroF { LivroF.stock = Stock.stock; }
14
15 LivroF -> Facturas { Facturas.stock = LivroF.stock;}
16
17 Facturas -> Factura {Factura.stock = Facturas.stock;}
18
19 Facturas -> Factura Facturas {Factura1.stock = Facturas0.stock; Facturas2.stock = Facturas0.
       stock;}
20
21 Factura -> CabecaF CorpoF { imprime(CorpoF.total); CorpoF.stock = Factura.stock;}
22
23 CabecaF -> IdFact IdE IdR
24
25 IdFact -> NumFact
26
27 NumFact -> id
28
29 IdE -> Nome NIF Mor NIB
30
31 IdR -> Nome NIF Mor
32
33 Nome -> string
34
35 NIF -> num
36
37 Mor -> string
38
39 NIB -> num
40
41 CorpoF -> Linhas {CorpoF.total = Linhas.total; Linhas.stock = CorpoF.stock;}
42
43 Linhas -> Linha {Linhas.total = Linhas.totalLinha; Linha.stock = Linhas.stock;}
44
45 Linhas -> Linha Linhas {Linhas0.total = Linha1.totalLinhas + Linhas2.total; Linha1.stock =
       Linhas0.stock; Linhas2.stock = Linhas0.stock; }
46
47 Linha -> Ref Qtd {  int pu = stock.getPU(Ref.num);
48                                          Linha.totalLinha = Qtd.quantidade * pu;
49                                          imprime Linha.totalLinha;
50
51                                          int qt = Linha.stock.getQuantidade(Ref.num);
52                                          Linha.stock.setQuantidade(qt-Qtd.quantidade);
53                                                  }
54
55 Ref -> NumProd { Ref.num = Numprod.num}
56
57 NumPro -> id {NumPro.num = id.lexeme}
58
59 Desc -> string
60
61 Qtd -> num  { Qtd.quantidade = num.lexeme;}
62
63 PU -> num       { PU.quantidade = num.lexeme;}
```