

Lavanda

Exercise with Attribute Grammar and its implementation in CoCo/R

May 2, 2006

Contents

1	Problem	2
2	Attribute Grammar - Solution	3
3	CoCoR implementation	7
4	Example test	13
5	Results	14
6	Files	15

Chapter 1

Problem

Lavanda is a Domain Specific Language (*DSL*) which main goal is describe the bags of clothes that Point of Gathering of a Laundry daily send to the Center to wash. Each bag has a identification number, the client name and its content is divided in one or more items. Each item have one or more clothe type (personal clothe or *household linen*), tinged type (white or color) and line type (cotton, wool and fiber). For each one of this items we keep in register the number of pieces that belongs to that item. The Independent Context Grammar G , mentioned below, defines the language Lavanda intended. The root is Lavanda, the terminal simbols are written is lowercase (pseudo-terminais), or uppercase (reserved-words), ou between apostrophes (sinais de pontuação) and null string í noted by &; the remaining simbols are Non-Terminals.

```
p1: Lavanda --> Cabec Sacos
p2: Cabec   --> data IdPR
p3: Sacos   --> Saco '.'
p4:         | Sacos Saco '.'
p5: Saco    --> num IdCli Lotes
p6: Lotes   --> Lote Outros
p7: Lote    --> Tipo Qt
p8: Tipo    --> Classe Tinto Fio
p9: Outros  --> &
p10:        | ';' Lotes
p11: IdPR   --> id
p12: IdCli  --> id
p13: Qt     --> num
p14,15: Classe--> corpo | casa
p16,17: Tinto --> br   | cor
p18,19,20: Fio  --> alg  | la   | fib
```

After transform G in a independent contex abstract grammar (you can reduce some productions that seems redundant), writte a **Attribute Grammar** for:

- compute (and print) total of bags sended and total of items of each cliente.
- compute (and print) total of pieces of each 12 items types (since 'body/br/alg' until 'house/cor/fib') sended to wash at laundry.
- compute total cost of each bag; suppose initially is given a table with prices of each item type.

The grammar should detect error situations: the identification number of bag is duplicated and should flag an error allways show up a bag for a client already finded.

Chapter 2

Attribute Grammar - Solution

The first step is write the abstract grammar.

To do that we eliminate all terminals without semantic charge (reserved words and signs). The grammar will be simplified by eliminating productions without alternatives that in right side just show up one terminal — in this case: p11, p12, p13.

```
p1a: Lavanda --> Cabec Sacos
p2a: Cabec   --> data id
p3a: Sacos   --> Saco
p4a:         | Sacos Saco
p5a: Saco    --> num id Lotes
p6a: Lotes   --> Lote Outros
p7a: Lote    --> Tipo num
p8a: Tipo    --> Classe Tinto Fio
p9a: Outros  --> &
p10a:        | Lotes
p11a: Classe--> corpo
p12a:        | casa
p13a: Tinto  --> br
p14a:        | cor
p15a: Fio    --> alg
p16a:        | la
p17a:        | fib
```

The next step is choose the attributes.

- For first item, we will need two synthesized attributes: `nSacos: int` associated at axiom `Lavanda` and `nLotes: int` associated at symbol `Saco`.

To compute each one will be necessary associate: `nSacos: int` at symbol `Sacos` and `nLotes: int` at symbol `Lotes` and at symbol `Outros`.

The computation and translate rules are:

```
p1a: Lavanda --> Cabec Sacos
      -- Lavanda.nSacos = Sacos.nSacos
      -- escreve( Lavanda.nSacos )
p3a: Sacos   --> Saco
      -- Sacos.nSacos = 1
p4a:         | Sacos Saco
```

```

-- Sacos0.nSacos = Sacos1.nSacos + 1
p5a: Saco --> num id Lotes
-- Saco.nLotes = Lotes.nLotes
-- escreve( Saco.nLotes )
p6a: Lotes --> Lote Outros
-- Lotes.nLotes = Outros.nLotes + 1
p9a: Outros --> &
-- Outros.nLotes = 0
p10a:      | Lotes
-- Outros.nLotes = Lotes.nLotes

```

- To this item will be needed 3 attributes:

1. `inEnv`: `HashTable` — `Saco`, `Lotes` and `Lote`;
2. `outEnv`: `HashTable` — `Lavanda`, `Sacos`, `Saco`, `Lotes`, `Lote` and `Outros`;
3. `name`: `string` — `Tipo`, `Classe`, `Tinto` and `Fio`.

The computation and translate rules are:

```

p1a: Lavanda --> Cabec Sacos
-- escreveT( Sacos.outEnv )
p3a: Sacos --> Saco
-- Saco.inEnv = Sacos.inEnv
-- Sacos.outEnv = Saco.outEnv
p4a:      | Sacos Saco
-- Saco.inEnv = Sacos1.outEnv
-- Sacos1.inEnv = Sacos0.inEnv
-- Sacos0.outEnv = Saco.outEnv
p5a: Saco --> num id Lotes
-- Lotes.inEnv = Saco.inEnv
-- Saco.outEnv = Lotes.outEnv
p6a: Lotes --> Lote Outros
-- Lote.inEnv = Lotes.inEnv
-- Outros.inEnv = Lote.outEnv
-- Lotes.outEnv = Outros.outEnv
p7a: Lote --> Tipo num
-- Lote.outEnv = updateTablePrice(Lote.inEnv, Tipo.name, num)
p8a: Tipo --> Classe Tinto Fio
-- Tipo.name = Classe.name + Tinto.name + Fio.name
p9a: Outros --> &
-- Outros.outEnv = Outros.inEnv;
p10a:      | Lotes
-- Lotes.inEnv = Outros.inEnv;
-- Outros.outEnv = Lotes.outEnv;
p11a: Classe --> corpo
-- Classe.name = "corpo"
p12a: Classe --> casa
-- Classe.name = "casa"
p13a: Tinto --> br
-- Tinto.name = "br"
p14a: Tinto --> cor
-- Tinto.name = "cor"
p15a: Fio --> alg

```

```

-- Fio.name = "alg"
  p16a: Fio --> la
-- Fio.name = "la"
  p17a: Fio --> fib
-- Fio.name = "fib"

```

- To this item will be needed 5 attributes:

1. `inTable`: `HashTable` — Sacos, Saco, Lotes, Lote and Outros;
Price table (inherited attribute).
2. `inIds`: `Vector` — Sacos and Saco;
Clients identifiers (Array — inherited attribute).
3. `outIds`: `Vector` — Sacos and Saco;
Clients identifiers (Array — synthesized attribute).
4. `custoTotal`: `int` — Saco, Lotes, Lote and Outros;
Cost of each bag (synthesized attribute).
5. `name`: `string` — Tipo, Classe, Tinto and Fio. Name of each attribute associated at Tipo
(synthesized attribute).

The computation and translate rules are:

```

p1a : Lavanda -> Cabec Sacos
      -- Sacos.inTable = initTable()
-- Sacos.inIds = initIds()
  p3a: Sacos --> Saco
      -- Saco.inTable = Sacos.inTable
      -- Saco.inIds = Sacos.inIds
      -- Sacos.outIds = Saco.outIds
-- escrevePreco( Saco.custoTotal )
  p4a:          | Sacos Saco
-- Saco.inTable = Sacos0.inTable
      -- Sacos1.inEnv = Sacos0.inEnv
-- Saco.inIds = Sacos1.outIds
-- Sacos1.inIds = Sacos0.inIds
-- Sacos0.outIds = Saco.outIds
-- escrevePreco( Saco.custoTotal )
  p5a: Saco --> num id Lotes
-- Saco.outEnv = novoId( Saco.inIds, num.value() )
-- if ( pertence( num,Saco.inIds ) )
      -- erro("Cliente ja existente!")
      -- Lotes.inTable = Saco.inTable
-- Saco.custoTotal = Lotes.custoTotal
  p6a: Lotes --> Lote Outros
      -- Lote.inTable = Lotes.inTable
      -- Outros.inTable = Lotes.inTable
      -- Lotes.custoTotal = Lote.custoTotal + Outros.custoTotal
  p7a: Lote --> Tipo num
      -- Lote.custoTotal = lookupPreco( Lote.inEnv, Tipo.name ) * num.value()
  p8a: Tipo --> Classe Tinto Fio
      -- Tipo.name = Classe.name + Tinto.name + Fio.name
  p9a: Outros --> &
      -- Outros.custoTotal = 0
  p10a:          | Lotes

```

```
        -- Outros.custoTotal = Lotes.custoTotal
    p11a: Classe --> corpo
-- Classe.name = "corpo"
    p12a: Classe --> casa
-- Classe.name = "casa"
    p13a: Tinto --> br
-- Tinto.name = "br"
    p14a: Tinto --> cor
-- Tinto.name = "cor"
    p15a: Fio --> alg
-- Fio.name = "alg"
    p16a: Fio --> la
-- Fio.name = "la"
    p17a: Fio --> fib
-- Fio.name = "fib"
```

Chapter 3

CoCoR implementation

"Lavanda.atg" ? ≡

```
COMPILER Lavanda
```

```
/*
 *          SCANNER          *
 */
```

```
COMPILER Lavanda
```

```
/*
 *          SCANNER          *
 */
```

```
public OpHashtable opTable;
```

```
// -----
```

```
IGNORECASE
```

```
CHARACTERS
```

```
letter = 'a'..'z'.
```

```
digit  = '0'..'9'.
```

```
TOKENS
```

```
ident = letter { letter }.
```

```
number = digit { digit }.
```

```
IGNORE '\r' + '\n' + '\t'
```

```
◇
```

File defined by ?, ?, ?.

"Lavanda.atg" ? ≡

```

/*****
*          PARSE
*          *
*****/

PRODUCTIONS

/*----- PRODUCTION FOR GRAMMAR NAME -----*/

Lavanda      (. int nSacos = 0;
              Hashtable inTable = opTable.initNLotes(),
              outTable,
              inTPrice = opTable.initTablePrice();
              ArrayList inIds = opTable.initIds(),
              outIds = inIds;      .)
= Cabec Sacos<out nSacos, inTable, out outTable, inTPrice, inIds, out outIds>
  (. Console.WriteLine("Total sacos: " +
                      nSacos + "\n");
   opTable.printTableLotes(outTable);
   .).

Cabec
= Data ident.

Data      (. int day, month, year;      .)
= number  (. day = Convert.ToInt32(t.val);
           if (day < 1 || day > 31)
             SemErr("Wrong day!");      .)
',-'
number    (. month = Convert.ToInt32(t.val);
           if (month < 1 || month > 12)
             SemErr("Wrong month!");      .)
'-' number (. year = Convert.ToInt32(t.val);
           if (year < 2000 || year > 2010)
             SemErr("Wrong year!");      .).

Sacos<out int nSacos, Hashtable inTable,out Hashtable outTable,
      Hashtable inTPrice,
      ArrayList inIds, out ArrayList outIds>
  (. int nSacos1 = 0;
   float custoTotal;      .)
= Saco<inTable,out outTable, inTPrice, out custoTotal, inIds, out outIds>
  (. nSacos = 1;      .)
[ Sacos<out nSacos1, outTable, out outTable, inTPrice, outIds, out outIds>
  (. nSacos += nSacos1;      .) ].

Saco<Hashtable inTable,out Hashtable outTable,Hashtable inTPrice,
  out float custoTotal, ArrayList inIds, out ArrayList outIds>
  (. int nLotes = 0, num;
   custoTotal = 0;      .)

= number      (. num = Convert.ToInt32(t.val); .)
ident '( Lotes<out nLotes, inTable,out outTable, inTPrice, out custoTotal>
  )',
  (. Console.WriteLine("N. lotes n8 sacco " +
                      num + ": " + nLotes);
   opTable.writePrice(custoTotal);
   if ( inIds.Contains( num ))
     throw new Exception("Cliente already exists!");
   outIds = opTable.newId(inIds, num);
   .).

```

◇

File defined by ?, ?, ?.

"Lavanda.atg" ? ≡

```
Lotes<out int nLotes, Hashtable inTable,out Hashtable outTable,
    Hashtable inTPrice, out float custoTotal>
    (. int nLotes1 = 0;
       float custoTotal2 = 0;    .)
= Lote<inTable,out outTable, inTPrice, out custoTotal>
    Outros<out nLotes1, outTable, out outTable, inTPrice, out custoTotal2>
        (. nLotes = nLotes1 + 1;
           custoTotal += custoTotal2;    .).

Lote<Hashtable inTable,out Hashtable outTable,
    Hashtable inTPrice, out float custoTotal>
    (. string name = "";    .)
= Tipo<out name> number
    (. outTable = opTable.updateTableLotes(
        inTable, name, Convert.ToInt32(t.val));
       custoTotal = opTable.lookupPrice(inTPrice, name) *
                               Convert.ToInt32(t.val);
       .).

Outros<out int nLotes, Hashtable inTable,out Hashtable outTable,
    Hashtable inTPrice, out float custoTotal>
    (. nLotes = 0;
       outTable = inTable;
       custoTotal = 0;    .)
= [ ',,' Lotes<out nLotes, inTable,out outTable, inTPrice, out custoTotal> ].

Tipo<out string name>    (. string name1 = "",
                          name2 = "",
                          name3 = "";    .)
= Classe<out name1> '--'
  Tinto<out name2> '--'
  Fio<out name3>    (. name = name1 + "-" +
                    name2 + "-" +
                    name3;    .).

Classe<out string name>
    (. name = "";    .)
= "corpo"    (. name = "corpo";    .)
  | "casa"    (. name = "casa"; .).

Tinto<out string name>
    (. name = "";    .)
= "br"    (. name = "br";    .)
  | "cor"    (. name = "cor"; .).

Fio<out string name>
    (. name = "";    .)
= "alg"    (. name = "alg";    .)
  | "la"    (. name = "la";    .)
  | "fib"    (. name = "fib"; .).
```

END Lavanda.

◇

File defined by ?, ?, ?.

Auxiliar functions over Hashtable aren't defined like follow:

"OpHashtable.cs" ? ≡

```
using System;
using System.Collections;

namespace Lavanda
{
    public class OpHashtable
    {
        public OpHashtable()
        { }

        // alinea b
        public Hashtable initNLotes()
        {
            Hashtable env = new Hashtable();
            env.Add("corpo-br-la",0);
            env.Add("corpo-br-alg",0);
            env.Add("corpo-br-fib",0);
            env.Add("corpo-cor-la",0);
            env.Add("corpo-cor-alg",0);
            env.Add("corpo-cor-fib",0);
            env.Add("casa-br-la",0);
            env.Add("casa-br-alg",0);
            env.Add("casa-br-fib",0);
            env.Add("casa-cor-la",0);
            env.Add("casa-cor-alg",0);
            env.Add("casa-cor-fib",0);

            return env;
        }

        public Hashtable updateTableLotes(Hashtable inTable, String name, int number)
        {
            IDictionaryEnumerator env = inTable.GetEnumerator();

            while(env.MoveNext())
            {
                string key = (string)env.Key;
                if (key.Equals(name))
                {
                    int pieces = (int)env.Value + number;
                    inTable.Remove(key);
                    inTable.Add(key, pieces);
                    break;
                }
            }
            return inTable;
        }
    }
}
```

◇

File defined by ?, ?, ?.

```
public void printTableLotes( Hashtable myList )
{
    IDictionaryEnumerator myEnumerator = myList.GetEnumerator();
    Console.WriteLine( "\t-Descricao-\t-N Lotes-" );
    while ( myEnumerator.MoveNext() )
        Console.WriteLine("\t{0}:\t{1}", myEnumerator.Key, myEnumerator.Value);
    Console.WriteLine();
}
```

// alinea c

```
public Hashtable initTablePrice()
{
    Hashtable env = new Hashtable();
    env.Add("corpo-br-la",1.0f);
    env.Add("corpo-br-alg",2.2f);
    env.Add("corpo-br-fib",3.4f);
    env.Add("corpo-cor-la",4.5f);
    env.Add("corpo-cor-alg",3.7f);
    env.Add("corpo-cor-fib",1.9f);
    env.Add("casa-br-la",2.6f);
    env.Add("casa-br-alg",5.3f);
    env.Add("casa-br-fib",7.1f);
    env.Add("casa-cor-la",3.5f);
    env.Add("casa-cor-alg",2.5f);
    env.Add("casa-cor-fib",2.3f);

    return env;
}
```

```
public float lookupPrice (Hashtable inPrice, string name )
{
    float price = 0;
    IDictionaryEnumerator env = inPrice.GetEnumerator();
    while ( env.MoveNext() )
    {
        if (env.Key.Equals(name))
        {
            price = (float)env.Value;
            break;
        }
    }
    return price;
}
```

◇

File defined by ?, ?, ?.

"OpHashtable.cs" ? ≡

```
public ArrayList initIds()
{
return ( new ArrayList() );
}

public ArrayList newId(ArrayList old, int num)
{
old.Add(num);
return ( (ArrayList)old.Clone() );
}

public void writePrice(float num)
{
Console.WriteLine("Preco Total: " + num + "\n" );
}
}
} // end namespace
◇
```

File defined by ?, ?, ?.

Chapter 4

Example test

"Test.txt" ? ≡

```
10-11-2005 today 1 dani (corpo-cor-la 1 , casa-cor-alg 2)
                  2 pedro (casa-br-fib 4)
                  3 celina (corpo-cor-alg 2, corpo-cor-la 3, corpo-cor-fib 1,
                             casa-cor-alg 2, casa-cor-la 3, casa-cor-fib 1)
```

◇

Chapter 5

Results

Results produced by compiler:

```
Nº lotes no saco 1: 2
Preco Total: 9.5

Nº lotes no saco 2: 1
Preco Total: 28.4

Nº lotes no saco 3: 6
Preco Total: 40.6

Total sacos: 3

-Descricao-      -Nº Lotes-
casa-cor-alg:    4
casa-br-la:     0
corpo-br-la:    0
corpo-br-fib:   0
corpo-cor-la:   4
corpo-cor-fib:  1
casa-cor-la:    3
corpo-br-alg:   0
casa-br-fib:    4
corpo-cor-alg:  2
casa-br-alg:    0
casa-cor-fib:   1
```

Chapter 6

Files

"Lavanda.atg" Defined by ?, ?, ?.
"OpHashtable.cs" Defined by ?, ?, ?.
"Test.txt" Defined by ?.