Lavanda

Exercise with Attribute Grammar and its implementation in Silver

July 10, 2007

Contents

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), or 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 writte 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 sinthesized 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.inEv;
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 — sinthesized attribute).
  4. `custoTotal: int` — `Saco`, `Lotes`, `Lote` and `Outros`;
     Cost of each bag (sinthesized attribute).
  5. `name: string` —- `Tipo`, `Classe`, `Tinto` and `Fio`. Name of each attribute associated at Tipo (sinthesized 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

Silver implementation

"Lavanda.sv" 7 ≡

```
grammar Lavanda;

import core;



----- TERMINALS -----


terminal NUMBER /[0-9]+/  lexer precedence = 5;
terminal DATE /[0-3]?[0-9][\/\-][0-1]?[0-9][\/\-][0-2][0-9][0-9][0-9]/ lexer precedence = 10;

terminal ID /[a-zA-Z]+/ lexer precedence = 5;
terminal CORPO /corpo/  lexer precedence = 10;
terminal CASA  /casa/   lexer precedence = 10;
terminal BR    /br/     lexer precedence = 10;
terminal COR   /cor/    lexer precedence = 10;
terminal ALG   /alg/    lexer precedence = 10;
terminal LA    /la/     lexer precedence = 10;
terminal FIB   /fib/    lexer precedence = 10;
terminal COMMA  ',';
terminal SEP    '-';
terminal PARR   '(';
terminal PARL   ')';


-- Symbols to ignore - spaces
ignore terminal WhiteSpace /[\t\n\r\ ]+/ ;

◇
```

File defined by 7, 8, 9, 10, 11, 12, 13, 14, 15, 16.

"Lavanda.sv" 8 ≡

```
---- NONTERMINALS AND ATTRIBUTES ----

start nonterminal Lavanda        with prt, nSacos, outEnv;
nonterminal Sacos                with prt, nSacos, inIds, outIds, inEnv, outEnv, inTable;
nonterminal Saco                 with prt, nLotes, inIds, outIds, inEnv, outEnv, inTable,
                                 custoTotal;
nonterminal Lotes, Outros        with prt, nLotes, inEnv, outEnv, inTable, custoTotal;
nonterminal Cabec                with prt;
nonterminal Lote                 with prt, inEnv, outEnv, inTable, custoTotal;
nonterminal Tipo                 with prt, name;
nonterminal Classe               with prt, name;
nonterminal Tinto                with prt, name;
nonterminal Fio                  with prt, name;


synthesized attribute nSacos :: Integer;
synthesized attribute nLotes :: Integer;
synthesized attribute prt    :: String;
synthesized attribute name   :: String;
synthesized attribute outIds :: [Integer];
synthesized attribute outEnv :: [PairStruct];
synthesized attribute custoTotal :: Float;



autocopy attribute inIds :: [Integer];
autocopy attribute inEnv :: [PairStruct];
autocopy attribute inTable :: [PairStruct];



--===== DataType (String,Float) =========

synthesized attribute id       :: String;
synthesized attribute value    :: Float;

nonterminal PairStruct         with id, value;

abstract production mkPair
pair :: PairStruct ::= s :: String f::Float
{
   pair.id = s;
   pair.value = f;
}

--=====================================

◇
```

File defined by 7, 8, 9, 10, 11, 12, 13, 14, 15, 16.

"Lavanda.sv" 9 ≡

```
---- PRODUCTIONS ----

concrete production Lavanda1
lavanda::Lavanda ::= cabec::Cabec sacos::Sacos
{
  --ITEM 1
  lavanda.nSacos = sacos.nSacos;

  --ITEM 2
  sacos.inEnv = initLotes();

  --ITEM 3
  sacos.inTable = initPrices();
  sacos.inIds = [];

  --PRINT
  lavanda.prt = sacos.prt ++ "\nTabela de Lotes:\n" ++ toStringHash(sacos.outEnv);
}


concrete production Cabec1
cabec::Cabec ::= DATE ID
{}


concrete production Sacos1
sacos::Sacos ::= saco::Saco
{
   --ITEM 1
   sacos.nSacos = 1;

   --ITEM 2
   sacos.outEnv = saco.outEnv;

   --ITEM 3
   sacos.outIds = saco.outIds;

   --PRINT
   sacos.prt = saco.prt ++
               "Custo Total do saco: " ++ toString(saco.custoTotal)++"\n\n";

}



concrete production Sacos2
sacos::Sacos ::= sacos2::Sacos saco::Saco
{
  --ITEM 1
  sacos.nSacos = sacos2.nSacos + 1;

  --ITEM 2
  saco.inEnv = sacos2.outEnv;
  sacos.outEnv = saco.outEnv;


}
```

◇

File defined by 7, 8, 9, 10, 11, 12, 13, 14, 15, 16.

```
  --ITEM 3
  saco.inIds = sacos2.outIds;
  sacos.outIds = saco.outIds;

  --PRINT
  sacos.prt = sacos2.prt ++ saco.prt ++
              "Custo Total do saco: " ++ toString(saco.custoTotal)++"\n\n";

}



concrete production Saco1
saco::Saco ::= n::NUMBER ID '(' lotes::Lotes ')'
{
   --ITEM 1
   saco.nLotes = lotes.nLotes;

   --ITEM 2
   saco.outEnv = lotes.outEnv;

   --ITEM 3
   saco.prt = if isElem(toInt(n.lexeme),saco.inIds)
              then error("Erro: Saco "++ n.lexeme ++" repetido!!\n")
              else "Saco n." ++ n.lexeme ++ " tem --> " ++ toString(saco.nLotes) ++ " lotes!\n"
         ;

   saco.outIds = updateList(toInt(n.lexeme), saco.inIds);
   saco.custoTotal = lotes.custoTotal;

}


concrete production Lotes1
lotes::Lotes ::= lote::Lote outros::Outros
{
  --ITEM 1
  lotes.nLotes = outros.nLotes + 1;

  --ITEM 2
  outros.inEnv = lote.outEnv;
  lotes.outEnv = outros.outEnv;

  --ITEM 3
  lotes.custoTotal = lote.custoTotal + outros.custoTotal;

}


concrete production Lote1
lote::Lote ::= tipo::Tipo n::NUMBER
{
   --ITEM 2
   lote.outEnv = updateHash(lote.inEnv, tipo.name, toFloat(n.lexeme));

◇
```

File defined by 7, 8, 9, 10, 11, 12, 13, 14, 15, 16.

"Lavanda.sv" 11 ≡

```
      --ITEM 3
      lote.custoTotal = lookup(lote.inTable,tipo.name) * toFloat(n.lexeme);
   }


   concrete production Outros1
   outros::Outros ::=
   {
      --ITEM 1
      outros.nLotes = 0;

      --ITEM 2
      outros.outEnv = outros.inEnv;

      --ITEM 3
      outros.custoTotal = 0.0;
   }


   concrete production Outros2
   outros::Outros ::= ',' lotes::Lotes
   {
      --ITEM 1
      outros.nLotes = lotes.nLotes;

      --ITEM 2
      outros.outEnv = lotes.outEnv;

      --ITEM 3
      outros.custoTotal = lotes.custoTotal;
   }




   concrete production Tipo1
   tipo::Tipo ::= classe::Classe '-' tinto::Tinto '-' fio::Fio
   {
     --ITEM 2
     tipo.name = classe.name ++ "-" ++ tinto.name ++ "-" ++ fio.name;
   }


   concrete production Classe1
   classe::Classe ::= CORPO
   {
     --ITEM 2
     classe.name = "corpo";
   }

   ◇
```

File defined by 7, 8, 9, 10, 11, 12, 13, 14, 15, 16.

"Lavanda.sv" 12 ≡

```
concrete production Classe2
classe::Classe ::= CASA
{
  --ITEM 2
  classe.name = "casa";
}


concrete production Tinto1
tinto::Tinto ::= BR
{
  --ITEM 2
  tinto.name = "br";
}


concrete production Tinto2
tinto::Tinto ::= COR
{
  --ITEM 2
  tinto.name = "cor";
}


concrete production Fio1
fio::Fio ::= ALG
{
  --ITEM 2
  fio.name = "alg";
}


concrete production Fio2
fio::Fio ::= LA
{
  --ITEM 2
  fio.name = "la";
}


concrete production Fio3
fio::Fio ::= FIB
{
  --ITEM 2
  fio.name = "fib";
}


◇
```

File defined by 7, 8, 9, 10, 11, 12, 13, 14, 15, 16.

"Lavanda.sv" 13 ≡

```
abstract production main
m::Main ::= args::String
{

  local attribute isF :: IOBoolean;
  isF = isFile(args, m.ioIn);

  local attribute file :: IOString;
  file = readFile(args, m.ioIn);

  local attribute text :: String;
  text = if isF.bValue then file.sValue else "" ;

  -- concrete Lavanda
  local attribute r :: Lavanda ;
  r = parse(text) ;

  m.ioOut
   = if isF.bValue
     then
       print("\n\n" ++ r.prt ++ "\nTotal de Sacos: " ++ toString(r.nSacos) ++
           "\n\n\n"
             ,
             file.io )
     else
       print ("File \"" ++ args ++ "\" not found.\n", isF.io)
    ;
}
```

◇

File defined by 7, 8, 9, 10, 11, 12, 13, 14, 15, 16.

"Lavanda.sv" 14 ≡

```
-------- FUNCTIONS --------

function isElem
Boolean ::= n::Integer lista::[Integer]
{
   local attribute res :: Boolean;

   res = if null(lista)
    then false
       else    if (head(lista) == n)
      then true
      else isElem(n,tail(lista))
   ;
   return res;
}


function updateList
[Integer] ::= n::Integer lista::[Integer]
{
   return cons(n,lista);
}



function insertHash
[PairStruct] ::= ps::[PairStruct] s::String n::Float
{
  local attribute res :: PairStruct;

  res = mkPair(s,n);

  return cons(res,ps);
}




function updateHash
[PairStruct] ::= ps::[PairStruct] s::String n::Float
{
  local attribute str :: String;
  local attribute act :: Float;
  local attribute pair :: PairStruct;

  str = case head(ps) of mkPair(id,_) => id end ;
  act = case head(ps) of mkPair(_,v) => v end;
  pair = mkPair(str,n+act);

  return  if null(ps)
        then insertHash(ps,s,n)
        else if str == s
             then cons(pair,tail(ps))
             else cons(head(ps),updateHash(tail(ps), s, n))
        ;

}
```


◇

File defined by 7, 8, 9, 10, 11, 12, 13, 14, 15, 16.

"Lavanda.sv" 15 ≡

```
function lookup
Float ::= ps::[PairStruct] s::String
{
  local attribute res :: Float;
  local attribute str  :: String;

  str = case head(ps) of mkPair(i,_) => i end;

  res = if null(ps)
        then 1.0
        else if str == s
             then case head(ps) of mkPair(_,v) => v end
             else lookup(tail(ps),s)
        ;

  return res;
}
```

```
function toStringHash
String ::= lista::[PairStruct]
{
   local attribute str:: String;
   local attribute ident :: String;
   local attribute val:: Float;

   ident  = case head(lista) of mkPair(s,_) => s end;
   val = case head(lista) of mkPair(_,v) => v end;

   str = if null(lista)
    then ""
    else ident ++ " " ++ toString(val) ++ "\n" ++ toStringHash(tail(lista))
   ;

   return str;

}
```

◇

File defined by 7, 8, 9, 10, 11, 12, 13, 14, 15, 16.

"Lavanda.sv" 16 ≡

```
function initLotes
[PairStruct] ::=
{
   local attribute ps :: [PairStruct];
   ps = insertHash(
          insertHash(
           insertHash(
            insertHash(
             insertHash(
              insertHash(
               insertHash(
                insertHash(
                 insertHash(
                  insertHash(
                   insertHash(
                    insertHash([]
                    ,"corpo-br-la", 0.0)
                   ,"corpo-br-alg",0.0)
                  ,"corpo-br-fib",0.0)
                 ,"corpo-cor-la",0.0)
                ,"corpo-cor-alg", 0.0)
               ,"corpo-cor-fib", 0.0)
              ,"casa-br-la", 0.0)
             ,"casa-br-alg", 0.0)
            ,"casa-br-fib", 0.0)
           ,"casa-cor-la", 0.0)
          ,"casa-cor-alg", 0.0),
         "casa-cor-fib", 0.0);

   return ps;

}

function initPrices
[PairStruct] ::=
{
   local attribute ps :: [PairStruct];
   ps = insertHash(
          insertHash(
           insertHash(
            insertHash(
             insertHash(
              insertHash(
               insertHash(
                insertHash(
                 insertHash(
                  insertHash(
                   insertHash(
                    insertHash([]
                    ,"corpo-br-la", 1.0)
                   ,"corpo-br-alg",2.2)
                  ,"corpo-br-fib",3.4)
                 ,"corpo-cor-la",4.5)
                ,"corpo-cor-alg", 3.7)
               ,"corpo-cor-fib", 1.9)
              ,"casa-br-la", 2.6)
             ,"casa-br-alg", 5.3)
            ,"casa-br-fib", 7.1)
           ,"casa-cor-la", 3.5)
          ,"casa-cor-alg", 2.5),
         "casa-cor-fib", 2.3);

   return ps;
}
```

◇

File defined by 7, 8, 9, 10, 11, 12, 13, 14, 15, 16.

Chapter 4

Example test

"test.txt" 17 ≡

```
10-07-2007
Lavanda
1 Dani   (corpo-cor-la 1, casa-cor-alg 2)
2 Celina (casa-br-fib 4)
3 Pedro  (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

```
user:~/Silver_Grammar/Lavanda$ ./lav test.txt


Saco n.1 tem --> 2 lotes!
Custo Total do saco: 9.5

Saco n.2 tem --> 1 lotes!
Custo Total do saco: 28.4

Saco n.3 tem --> 6 lotes!
Custo Total do saco: 40.6


Tabela de Lotes:
casa-cor-fib 1.0
casa-cor-alg 4.0
casa-cor-la 3.0
casa-br-fib 4.0
casa-br-alg 0.0
casa-br-la 0.0
corpo-cor-fib 1.0
corpo-cor-alg 2.0
corpo-cor-la 4.0
corpo-br-fib 0.0
corpo-br-alg 0.0
corpo-br-la 0.0

Total de Sacos: 3
```

Chapter 6

Files

"Lavanda.sv" Defined by 7, 8, 9, 10, 11, 12, 13, 14, 15, 16.
"test.txt" Defined by 17.