Lavanda

Exercise with Attribute Grammar and its implementation in AntLR

May 2, 2006

Contents

1 Problem **2**

2 Attribute Grammar - Solution **3**

3 AntLR implementation **7**

3.1 Grammar 7

3.2 Price Table 11

3.3 Main 13

4 Example test **14**

5 Files **15**

Chapter 1

Problem

Lavanda is a Domain Specific Language (*DSL*) which main goal is drescribe 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 writte the abstract grammar.
To do that we eliminate all terminals without semantic charge (reserved words and signs). The grammarr 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

AntLR implementation

3.1 Grammar

"LexerParser.g" 7 ≡

```
header
{
    // gets inserted in the C# source file before any
    // generated namespace declarations
    // hence -- can only be using directives
    using System.Collections;
}

options {

   language  = "CSharp";
   namespace = "Lavanda";          // encapsulate code in this namespace
}

class MyLexer extends Lexer;
options {
   caseSensitive=false;                 // D = d     (on LEXER rules match)
   caseSensitiveLiterals=false;         // FOR = fOr (only for tokens)
}

COMMA            : ','   ;
LPAREN            : '('   ;
RPAREN            : ')'   ;
MINUS            : '-'  ;

protected LETTER  : 'a'..'z';
protected DIGIT         : '0'..'9';


IDENTIFIER        : LETTER ( LETTER )* ;
NUMBER            : DIGIT ( DIGIT )* ;

◇
```

File defined by 7, 8, 9, 10.

"LexerParser.g" 8 ≡

```
protected NEWLINE
   :
         (
               options { generateAmbigWarnings=false; }
           : '\n'
         | '\r'
           | '\r''\n'
         )
           { newline(); }
   ;

WHITESPACE
   :
      ( ' '
      | '\t'
      | NEWLINE
      )+
      { $setType(Token.SKIP); }
   ;


class MyParser extends Parser;
options {
   //exportVocab=My;
}
{
public OpHashtable opTable;
}

lavanda
options {defaultErrorHandler=false;}
   {
      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    : date IDENTIFIER
      ;

date     : NUMBER MINUS NUMBER MINUS NUMBER
      ;

◇
```

File defined by 7, 8, 9, 10.

"LexerParser.g" 9 ≡

```
     sacos [ out int nSacos , Hashtable inTable, out Hashtable outTable,
          Hashtable inTPrice, ArrayList inIds, out ArrayList outIds]
     options {defaultErrorHandler=false;}
        {
           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]
     options {defaultErrorHandler=false;}
        {
           int nLotes = 0, num;
           custoTotal = 0;
        }
           :  t1:NUMBER { num = Convert.ToInt32(t1.getText()); }
              IDENTIFIER
              LPAREN
              lotes[out nLotes, inTable, out outTable, inTPrice, out custoTotal]
              RPAREN { Console.WriteLine("N lotes no saco " + num + ": " + nLotes);
                     opTable.writePrice(custoTotal);
                      if ( inIds.Contains( num ))
                       throw new Exception("Cliente already exists!");
                      outIds = opTable.newId(inIds, num);
                     }
           ;

     lotes [out int nLotes, Hashtable inTable, out Hashtable outTable, Hashtable inTPrice, out float custoTotal]
     options {defaultErrorHandler=false;}
        {
           int nLotes1 = 0;
           float custoTotal2 = 0;
        }
           : lote[inTable, out outTable, inTPrice, out custoTotal]
              {nLotes = 1;}
            ( COMMA lotes[out nLotes1, inTable, out outTable, inTPrice, out custoTotal2] )*
              {nLotes += nLotes1; outTable=inTable; custoTotal +=custoTotal2;}
           ;

     lote  [Hashtable inTable, out Hashtable outTable, Hashtable inTPrice, out float custoTotal]
     options {defaultErrorHandler=false;}
        {
           string name = "";
        }
           : tipo[out name] t1:NUMBER
              {
                 outTable = opTable.updateTableLotes(inTable, name, Convert.ToInt32(t1.getText()));
                 custoTotal = opTable.lookupPrice(inTPrice, name) * Convert.ToInt32(t1.getText());
              }
           ;
     ◇
```

File defined by 7, 8, 9, 10.

"LexerParser.g" 10 ≡

```
    tipo  [out string name]
    options {defaultErrorHandler=false;}
       {
          string name1 = "",
          name2 = "",
          name3 = "";
       }
          : classe[out name1] MINUS tinto[out name2] MINUS fio[out name3]
             { name = name1 + "-" + name2 + "-" + name3; }
          ;

    classe   [out string name]
    options {defaultErrorHandler=false;}
       {
          name = "";
       }
          : "corpo"   { name = "corpo"; }
          | "casa" { name = "casa"; }
          ;

    tinto [out string name]
    options {defaultErrorHandler=false;}
       {
          name = "";
       }
          : "br"      { name = "br"; }
          | "cor"     { name = "cor"; }
          ;

    fio      [out string name]
    options {defaultErrorHandler=false;}
       {
          name = "";
       }
          : "alg"     { name = "alg"; }
          | "la"      { name = "la"; }
          | "fib"     { name = "fib"; }
          ;


    ◇
```

File defined by 7, 8, 9, 10.

3.2 Price Table

"Op_Hashtable.cs" 11 ≡

```
    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 11, 12, 13a.

"Op_Hashtable.cs" 12 ≡

```
        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 11, 12, 13a.

"Op_Hashtable.cs" 13a ≡

```
        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 11, 12, 13a.

3.3 Main

"LavandaCompiler.cs" 13b ≡

```
  using System;
  using System.Collections;
  using System.IO;
  using antlr;
  using Lavanda;

  public class LavandaCompiler {

     public static void Main (string[] args)
     {
        if (args.Length > 0) {
                Stream stream = new FileStream(args[0], FileMode.Open, FileAccess.Read, FileShare.Read);
                MyLexer lexer = new MyLexer(stream);
                MyParser parser = new MyParser(lexer);
                parser.opTable = new OpHashtable();
        parser.lavanda();


        } else
           Console.WriteLine("-- No source file specified");
     }



  }
  ◇
```

Chapter 4

Example test

"Test.txt" 14 ≡

```
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

Files

`"LavandaCompiler.cs"` Defined by 13b.
`"LexerParser.g"` Defined by 7, 8, 9, 10.
`"Op_Hashtable.cs"` Defined by 11, 12, 13a.
`"Test.txt"` Defined by 14.