Lavanda

Exercise with Attribute Grammar and its implementation in SableCC

July 13, 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

SableCC implementation

3.1 The Grammar

"Lavanda.sable" 7 ≡

```
Package lavanda;

Helpers
        sep     = '-';
        dig     = ['0' .. '9'];
        day     = ['0' .. '3']? dig;
        month   = ['0' .. '1']? dig;
        year    = ['0' .. '2'] dig dig dig;
        letter  = [['a' .. 'z'] + ['A' .. 'Z']];
        space   = ' ';
        newline = 0x000A;
        creturn = 0x000D;
        tab     = 0x0009;

Tokens
        number  = dig+;
        date    = day sep month sep year;
        dash    = sep;
        l_par   = '(';
        r_par   = ')';
        comma   = ',';
        cor     = 'cor';
        br      = 'br';
        alg     = 'alg';
        fib     = 'fib';
        la      = 'la';
        casa    = 'casa';
        corpo   = 'corpo';
        id      = letter+;
        blank   = (space | newline | creturn | tab);

Ignored Tokens
        blank;
```

◇

File defined by 7, 8.

"Lavanda.sable" 8 ≡

```
     Productions

             lavanda = cabec sacos
                     ;

             cabec   = date id
                     ;

             sacos   = {one} saco
                     | {many} sacos saco
                     ;

             saco    = number id l_par lotes r_par
                     ;

             lotes   = lote outros
                     ;

             lote    = tipo number
                     ;

             outros  = {empty}
                     | {not_empty} comma lotes
                     ;

             tipo    = classe [d1]:dash tinto [d2]:dash fio
                     ;

             classe  = {body} corpo
                     | {house} casa
                     ;

             tinto   = {white} br
                     | {color} cor
                     ;

             fio     = {cotton} alg
                     | {fiber} fib
                     | {wool} la
                     ;
```

◇
File defined by 7, 8.

3.2 The Semantic Analyser

"SemanticAnalyser.java" 9 ≡

```
package lavanda;

import lavanda.analysis.*;
import lavanda.node.*;
import java.util.*;


public class SemanticAnalyser extends DepthFirstAdapter{

   private int numSacos = 0;
   private int nLotes = 0;
   private String name;
   private Double custoTotal = 0.0;
   private Hashtable<String, Double> priceTable = Environment.InitTablePrice();
   private Hashtable<String, Integer> typeTable = Environment.InitTableMaterials();
   private LinkedList<Integer> idClients = new LinkedList<Integer>();



   public void outALavanda(ALavanda no){
      //prints
      System.out.println(Environment.printTable(typeTable));
      System.out.println("Total de Sacos: " + numSacos);
   }



   public void outAOneSacos(AOneSacos no){

      //item 1
      this.numSacos = 1;

      //item 3
      System.out.println("Custo total do saco: " + this.custoTotal);
      this.custoTotal = 0.0;
   }



   public void outAManySacos(AManySacos no){

      //item 1
      this.numSacos += 1;

      //Item 3
      System.out.println("Custo total do saco: " + custoTotal);
      this.custoTotal = 0.0;
   }
```

◇

File defined by 9, 10.

"SemanticAnalyser.java" 10 ≡

```
        public void outASaco(ASaco no){

           //item 1
           String num = no.getNumber().getText().trim();

           //Item 3
           Integer n = new Integer(num).intValue();
           if(this.idClients.contains(n)){
              System.err.println("Erro: Cliente no. "+ num + " aparece repetido!");
              System.exit(0);
           }
           this.idClients.add(n);

           //print
           System.out.println("\nSaco no. " + num + " tem --> "+ nLotes +" Lotes!");
        }

        public void outALotes(ALotes no){

           //item 1
           this.nLotes += 1;
        }



        public void outAEmptyOutros(AEmptyOutros no){
           //item 1
           this.nLotes = 0;
        }



        public void outALote (ALote no){

           //item 2
           String num = no.getNumber().getText().trim();
           Integer n = new Integer(num).intValue();
           Integer ntab = this.typeTable.get(name);
           this.typeTable.put(name,n+ntab);

           //item3
           this.custoTotal += this.priceTable.get(name) * n;
        }



        public void outATipo(ATipo no){

           //item2
           String classe = no.getClasse().toString().trim();
           String tinto = no.getTinto().toString().trim();
           String fio = no.getFio().toString().trim();
           this.name = classe + "-" + tinto + "-" + fio;
        }
     }
```

◇

File defined by 9, 10.

3.3 The Main Class

"Main.java" 11 ≡

```
package lavanda;

import lavanda.lexer.*;
import lavanda.node.*;
import lavanda.parser.*;

import java.io.*;

public class Main {

   public static void main(String[] args) {
         if (args.length > 0) {
            try {
               /* Criar a AST */
               Lexer lexer = new Lexer (new PushbackReader(
                  new FileReader(args[0]), 1024));
               Parser parser = new Parser(lexer);
               Start ast = parser.parse() ;

               /* Fazer a anlise semntica */
               SemanticAnalyser sa = new SemanticAnalyser () ;
               ast.apply(sa) ;
            }
            catch (Exception e) {
               System.out.println (e) ;
            }
         } else {
            System.err.println("usage: java simpleAdder inputFile");
            System.exit(1);
         }
      }
}
```

◇

Chapter 4

Example test

"test.txt" 12 ≡

```
13-07-2007
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

```
C:\java -jar lavanda.jar teste.txt

Saco no. 1 tem --> 2 Lotes!
Custo total do saco: 9.5

Saco no. 2 tem --> 1 Lotes!
Custo total do saco: 28.4

Saco no. 3 tem --> 6 Lotes!
Custo total do saco: 40.6

casa-br-la      0
corpo-br-fib    0
casa-br-alg     0
corpo-cor-la    4
corpo-cor-alg   2
casa-cor-alg    4
casa-br-fib     4
corpo-br-la     0
casa-cor-la     3
corpo-br-alg    0
corpo-cor-fib   1
casa-cor-fib    1

Total de Sacos: 3
```

Chapter 6

Files

`"Lavanda.sable"` Defined by 7, 8.
`"Main.java"` Defined by 11.
`"SemanticAnalyser.java"` Defined by 9, 10.
`"test.txt"` Defined by 12.