Exercício para Avaliação n.º 3

Bruno Azevedo*and Miguel Costa†

Módulo Engenharia Gramatical,
UCE30 Engenharia de Linguagens,
Mestrado em Engenharia Informátical,
Universidade do Minho

3 de Fevereiro de 2012

Resumo

Este documento apresenta a resolução do Exercício Prático n.º 3 do módulo de Engenharia de Linguagens. O exercício está relacionado com a geração automática de Processadores de Linguagens a partir de Gramáticas.

Era pretendido criar uma linguagem para fazer movimentar um Robo num Terreno e depois criar um processador para as frases da linguagem com algumas funcionalidades.

*Email: azevedo.252@gmail.com
†Email: miguelpintodacosta@gmail.com

Conteúdo

1 Ambiente de Trabalho **3**

2 Descrição do problema **3**

3 Criação da linguagem **3**

4 Conclusões **7**

1 Ambiente de Trabalho

Foi necessário usar um Gerador de Compiladores para gerar o nosso próprio compilador, por isso usamos o AnTLR que é também usado nas aulas. Para facilitar o debug durante a resuloção do problema que era dada, usamos a ferramenta AnTLRWorks, que tem um interface bastante agradável e simpático para problemas desta natureza.

A linguem de programação adoptada foi o JAVA. De forma tornar a nossa solução mais legível e estruturada, criamos classes com o auxilio do IDE NetBeans que nos ajuda no desenvolvimento do código JAVA e ainda na criação da sua documentação (javadoc).

2 Descrição do problema

Imaginemos um robo com a função de aspirar um terreno de forma retangular. Este terreno tem uma área que é conhecida pela robo e que acaba por limitar o raio de ação dele.

O robo pode ter definida uma posição inicial e os seus movimentos podem ser em quatro direções diferentes (norte, sul, este e oeste) com uma peso associado que representa a distância que se vai deslocar (por exemplo `NORTE 4`, desloca-se 4 unidades para norte). Tem ainda a opção de estar ligado ou desligado que define se está ativo ou não para aspirar.

Com base na descrição do robo, era pedido:

1. Criar uma linguagem que conseguisse descrever uma rotina possivél para o robo. Esta linguagem deve permitir ainda que tenha no inicio certas definições como a dimensão do terreno e a posição inicial do robo.
2. Depois de definida a linguagem tinhamos de criar um processador para as possíveis frases que podiam ser geradas com as funcionalidades de:
   - Verificar que o robo não se movimenta para fora da área de limpeza.
   - Calcular a distância (em cm) que o robo percorreu durante a sua rotina.
   - Determinar quantas mudanças de direção foram feitas pelo robo.
   - Determinar a distância média que o robo se desloca por cada movimento.

3 Criação da linguagem

Analisando o que era pretendido para descrever a rotina do robo, tentamos criar uma linguagem com uma sintaxe de fácil leitura e sem ambiguidades. Depois de analisar várias alternativas, definimos a linaguem com a seguinte estrutura:

Listing 1: Estrutura da gramática

```
1  ASPIRADOR
2  {
3      DEFINICOES
4      {
5          definicao1; definicao2;
6      }
7      MOVIMENTOS
8          instrucao1;
9          instrucao2;
10         ....
11 }
```

Uma linguagem tem de ter simbolos terminais e neste caso nós definimos os símbolos:

- DIM
- POS
- LIGAR
- DESLIGAR
- NORTE
- SUL
- ESTE
- OESTE
- ID
- INT

Definindo formalmente a gramática ficou:

Listing 2: Gramática

```
grammar robot;

robot
        : 'ASPIRADOR' '{' corpo '}'
        ;

corpo
        : 'DEFINICOES' definicoes 'MOVIMENTOS' movimentos
        ;

definicoes
        : '{' (definicao)+ '}'
        ;

definicao
        : DIM '=' '(' x=INT ',' y=INT ')' ';'
        | POS '=' '(' x=INT ',' y=INT ')' ';'
        ;

movimentos
        : movimento (movimento)*
        ;

movimento
        : LIGAR ';'
        | DESLIGAR ';'
        | NORTE INT ';'
        | SUL INT ';'
        | ESTE INT ';'
        | OESTE INT ';'
        ;

DIM     : ('d'|'D')('i'|'I')('m'|'M');
POS     : ('p'|'P')('o'|'O')('s'|'S');

LIGAR    : ('l'|'L')('i'|'I')('g'|'G')('a'|'A')('r'|'R');
DESLIGAR : ('d'|'D')('e'|'E')('s'|'S')('l'|'L')('i'|'I')('g'|'G')('a'|'A')('r'|'R');

NORTE   : ('n'|'N')('o'|'O')('r'|'R')('t'|'T')('e'|'E');
SUL     : ('s'|'S')('u'|'U')('l'|'L');
```

```
ESTE    : ('e'|'E')('s'|'S')('t'|'T')('e'|'E');
OESTE   : ('o'|'O')('e'|'E')('s'|'S')('t'|'T')('e'|'E');

ID      : ('a'..'z'|'A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'0'..'9'|'_')*
        ;

INT     : '0'..'9'+
        ;

COMMENT
        : '//' ~('\n'|'\r')* '\r'? '\n' {$channel=HIDDEN;}
        | '/*' ( options {greedy=false;} : . )* '*/' {$channel=HIDDEN;}
        ;

WS      : ( ' '
        | '\t'
        | '\r'
        | '\n'
        ) {$channel=HIDDEN;}
        ;
```

Depois de gerada a gramática, algumas das frases que se podem gerar, são:

Listing 3: Frase gerada 1

```
ASPIRADOR
{
        DEFINICOES
        {
                dim = (100 , 150) ; pos = (0 , 0) ;
        }
        MOVIMENTOS
                LIGAR;
                NORTE 2 ;
                ESTE 150 ;
                ESTE 3 ;
                ESTE 8 ;
                OESTE 25 ;
                OESTE 2 ;
                SUL 5;
                DESLIGAR ;
                SUL 0;
                NORTE 10;
                LIGAR;
                OESTE 0;
}
```

Listing 4: Frase gerada 2

```
ASPIRADOR
{
        DEFINICOES
        {
                dim = (15 , 15) ; pos = (7 , 7) ;
        }
        MOVIMENTOS
                LIGAR;
                NORTE 2 ;
                ESTE 150 ;
                ESTE 3 ;
                ESTE 2 ;
                SUL 1 ;
                OESTE 5 ;
```

```
15                  SUL 5;
16                  DESLIGAR ;
17                  SUL 0;
18                  NORTE 10;
19                  LIGAR;
20                  OESTE 4;
21  }
```

Analisando a gramática a as frases geradas a partir dela, vericámos que o elemento raiz é `robot` e o parser terá de encontrar no início a palavra `ASPIRADOR` seguido de um `corpo` que se encontra dentro de chavetas.

4 Conclusões

A resolução deste exercício permitiu perceber melhor a forma como as linguagens de estrutura para a resolução de determinados problemas. Depois de definida a GIC e criando a GA, conseguimos realizar os cálculos que eram pretendidos para a soma.

Apesar de serem dois exercícios para calcular um resultado de forma diferente, deu para perceber que o reaciocínio para resolver é idêntifo com ambos os casos.

Serviu de consolidação da matéria dada até agora no módulo de Engenharia de Linguagens, tendo em conta que conseguimos resolver os exercícios com sucesso.