Construção de Compiladores 1 - 2015.1 - Prof. Daniel Lucrédio
Aula 07 - Análise Semântica - roteiro

Demonstração 1 – Analisador semântico "na mão"

---

1. Criar um novo arquivo, no Desktop, com um programa de exemplo (programa.alg)

```
:DECLARACOES
numero1:INTEIRO
numero2:INTEIRO
numero3:INTEIRO
aux:INTEIRO

:ALGORITMO
% Coloca 3 números em ordem crescente
LER numero1
LER numero2
LER numero3
SE numero1 > numero2 ENTAO
   INICIO
      ATRIBUIR 2+3-4+5-6*5-1 A aux
      ATRIBUIR numero1 A numero2
      ATRIBUIR aux A numero1
   FIM
SE numero1 > numero3 E numero2 <= numero4 E numero1 > 3 OU numero2 <> numero4
ENTAO
   INICIO
      ATRIBUIR (numero3) A aux
      ATRIBUIR numero1 A numero3
      ATRIBUIR aux A numero1
   FIM
SE numero2 > numero3 ENTAO
   INICIO
      ATRIBUIR numero3 A aux
      ATRIBUIR numero2 A numero3
      ATRIBUIR aux A numero2
   FIM
IMPRIMIR numero1
IMPRIMIR numero2
IMPRIMIR numero3
```

2. Abrir o NetBeans, e abrir projeto Java "AlgumaParser"
3. Copiar para projeto "AlgumaParserComAnalisadorSemantico"

4. Criar enumeration TipoVariavel

```
public enum TipoVariavel {
    INTEIRO, REAL
}
```

5. Criar classe EntradaTabelaDeSimbolos

```
public class EntradaTabelaDeSimbolos {
    public String nome;
    public TipoVariavel tipo;
}
```

6. Criar classe TabelaDeSimbolos

```
public class TabelaDeSimbolos {
    private List<EntradaTabelaDeSimbolos> tabelaDeSimbolos;

    public TabelaDeSimbolos() {
        tabelaDeSimbolos = new ArrayList<EntradaTabelaDeSimbolos>();
    }

    public int instalarNome(String nome, TipoVariavel tipo) {
        if(jaFoiDeclarado(nome)) {
            throw new RuntimeException("Erro semântico: Variável "+nome+" foi
declarada duas vezes");
        }
        EntradaTabelaDeSimbolos etds = new EntradaTabelaDeSimbolos();
        etds.nome = nome;
        etds.tipo = tipo;
        tabelaDeSimbolos.add(etds);
        return tabelaDeSimbolos.size()-1;
    }

    public TipoVariavel determinaTipo(String nome) {
        for(EntradaTabelaDeSimbolos etds:tabelaDeSimbolos) {
            if(etds.nome.equals(nome))
                return etds.tipo;
        }
        return null;
    }

    public boolean jaFoiDeclarado(String nome) {
        for(EntradaTabelaDeSimbolos etds:tabelaDeSimbolos) {
            if(etds.nome.equals(nome))
                return true;
        }
        return false;
    }
}
```

7. Adicionar o objeto da tabela de símbolos no parser

```
    TabelaDeSimbolos ts;

    public AlgumaParser(AlgumaLexico lex) {
        ts = new TabelaDeSimbolos();
      ...
```

8. Modificar a regra tipoVar

```
    //tipoVar : 'INTEIRO' | 'REAL';
    TipoVariavel tipoVar() {
        if (lookahead(1).nome == TipoToken.PalavraChave &&
lookahead(1).lexema.equals("INTEIRO")) {
            match(TipoToken.PalavraChave, "INTEIRO");
            return TipoVariavel.INTEIRO;
        } else if (lookahead(1).nome == TipoToken.PalavraChave &&
lookahead(1).lexema.equals("REAL")) {
            match(TipoToken.PalavraChave, "REAL");
            return TipoVariavel.REAL;
        } else {
            erroSintatico("INTEIRO","REAL");
            return null;
        }
    }
```

9. Modificar a regra declaracao

```
    //declaracao : VARIAVEL ':' tipoVar;
    void declaracao() {
        String nomeVar = lookahead(1).lexema;
        match(TipoToken.Var);
        match(TipoToken.Delim);
        TipoVariavel tipoVar = tipoVar();
        ts.instalarNome(nomeVar, tipoVar);
    }
```

10. Modificar as regras para expressões aritméticas

```
    TipoVariavel expressaoAritmetica() {
        TipoVariavel tipoTermo = termoAritmetico();
        TipoVariavel tipoExp = expressaoAritmetica2();
        if (tipoTermo == TipoVariavel.REAL || tipoExp == TipoVariavel.REAL) {
            return TipoVariavel.REAL;
        } else {
            return TipoVariavel.INTEIRO;
        }
    }

    TipoVariavel expressaoAritmetica2() {
        if (lookahead(1).nome == TipoToken.OpAritSoma || lookahead(1).nome ==
TipoToken.OpAritSub) {
            TipoVariavel tipoExp1 = expressaoAritmetica2SubRegra1();
            TipoVariavel tipoExp2 = expressaoAritmetica2();
            if (tipoExp1 == TipoVariavel.REAL || tipoExp2 ==
TipoVariavel.REAL) {
                return TipoVariavel.REAL;
            } else {
```

```
                return TipoVariavel.INTEIRO;
            }
        } else { // vazio
            return null;
        }
    }

    TipoVariavel expressaoAritmetica2SubRegra1() {
        if (lookahead(1).nome == TipoToken.OpAritSoma) {
            match(TipoToken.OpAritSoma);
            return termoAritmetico();
        } else if (lookahead(1).nome == TipoToken.OpAritSub) {
            match(TipoToken.OpAritSub);
            return termoAritmetico();
        } else {
            erroSintatico("+", "-");
            return null;
        }
    }

    //termoAritmetico : termoAritmetico '*' fatorAritmetico | termoAritmetico
'/' fatorAritmetico | fatorAritmetico;
    // também precisa fatorar à esquerda e eliminar recursão à esquerda
    // termoAritmetico : fatorAritmetico termoAritmetico2
    // termoAritmetico2 : ('*' fatorAritmetico | '/' fatorAritmetico)
termoAritmetico2 | <<vazio>>
    TipoVariavel termoAritmetico() {
        TipoVariavel tipoTermo1 = fatorAritmetico();
        TipoVariavel tipoTermo2 = termoAritmetico2();
        if (tipoTermo1 == TipoVariavel.REAL || tipoTermo2 ==
TipoVariavel.REAL) {
            return TipoVariavel.REAL;
        } else {
            return TipoVariavel.INTEIRO;
        }
    }

    TipoVariavel termoAritmetico2() {
        if (lookahead(1).nome == TipoToken.OpAritMult || lookahead(1).nome ==
TipoToken.OpAritMult) {
            TipoVariavel tipoTermo1 = termoAritmetico2SubRegra1();
            TipoVariavel tipoTermo2 = termoAritmetico2();
            if (tipoTermo1 == TipoVariavel.REAL || tipoTermo2 ==
TipoVariavel.REAL) {
                return TipoVariavel.REAL;
            } else {
                return TipoVariavel.INTEIRO;
            }
        } else { // vazio
            return null;
        }
```

```
    }

    TipoVariavel termoAritmetico2SubRegra1() {
        if (lookahead(1).nome == TipoToken.OpAritMult) {
            match(TipoToken.OpAritMult);
            return fatorAritmetico();
        } else if (lookahead(1).nome == TipoToken.OpAritDiv) {
            match(TipoToken.OpAritDiv);
            return fatorAritmetico();
        } else {
            erroSintatico("*", "/");
            return null;
        }
    }

    //fatorAritmetico : NUMINT | NUMREAL | VARIAVEL | '(' expressaoAritmetica
')'
    TipoVariavel fatorAritmetico() {
        if (lookahead(1).nome == TipoToken.NumInt) {
            match(TipoToken.NumInt);
            return TipoVariavel.INTEIRO;
        } else if (lookahead(1).nome == TipoToken.NumReal) {
            match(TipoToken.NumReal);
            return TipoVariavel.REAL;
        } else if (lookahead(1).nome == TipoToken.Var) {
            String nomeVar = lookahead(1).lexema;
            if (!ts.jaFoiDeclarado(nomeVar)) {
                throw new RuntimeException("Erro semântico: Variável " +
nomeVar + " não foi declarada!");
            }
            TipoVariavel tipoVar = ts.determinaTipo(nomeVar);
            match(TipoToken.Var);
            return tipoVar;
        } else if (lookahead(1).nome == TipoToken.AbrePar) {
            match(TipoToken.AbrePar);
            TipoVariavel tipoExp = expressaoAritmetica();
            match(TipoToken.FechaPar);
            return tipoExp;
        } else {
            erroSintatico(TipoToken.NumInt.toString(),
TipoToken.NumReal.toString(), TipoToken.Var.toString(), "(");
            return null;
        }
    }
```

11. Modificar a regra de atribuição

```
    void comandoAtribuicao() {
        match(TipoToken.PalavraChave, "ATRIBUIR");
        TipoVariavel tipoExp = expressaoAritmetica();
        match(TipoToken.PalavraChave, "A");
```

```
        String nomeVar = lookahead(1).lexema;
        if (!ts.jaFoiDeclarado(nomeVar)) {
            throw new RuntimeException("Erro semântico: Variável " + nomeVar
+ " não foi declarada!");
        }
        TipoVariavel tipoVar = ts.determinaTipo(nomeVar);
        match(TipoToken.Var);
        if(tipoVar == TipoVariavel.INTEIRO && tipoExp == TipoVariavel.REAL)
            throw new RuntimeException("Erro semântico: Variável " + nomeVar
+ " deve ser do tipo REAL!");
    }
```

12. Comentar as linhas com System.out (Dentro dos métodos lerToken e match 2x)
13. Testar
    13.1. No programa de exemplo, vai dar erro que a variável numero4 não foi declarada
14. Modificar o programa para incluir um erro de atribuição

Ex: `ATRIBUIR numero4 A numero1`

Demonstração 2 – Analisador semântico no ANTLR usando DDS S-atribuída

---

1. Criar um novo arquivo, no Desktop, com um programa de exemplo (teste.txt)

```
345
```

2. Abrir o Antlrworks e criar a seguinte gramática

```
grammar Numeros;

programa returns [ int val ]
      :     numero { $val = $numero.val; } '\r'?'\n'
      ;

numero returns [ int val ]
      :     digito n2=numero { $val = $n2.val + $digito.val; }
      |     digito { $val = $digito.val; }
      ;

digito returns [ int val ]
      :     '0' { $val = 0; }
      |     '1' { $val = 1; }
      |     '2' { $val = 2; }
      |     '3' { $val = 3; }
      |     '4' { $val = 4; }
      |     '5' { $val = 5; }
      |     '6' { $val = 6; }
      |     '7' { $val = 7; }
      |     '8' { $val = 8; }
      |     '9' { $val = 9; }
      ;
```

3. Rodar num projeto do NetBeans (método main abaixo)

```
            ANTLRInputStream input = new ANTLRInputStream(
                new
FileInputStream("C:\\Users\\Daniel\\Desktop\\teste.txt"));
            NumerosLexer lexer = new NumerosLexer(input);
            CommonTokenStream tokens = new CommonTokenStream(lexer);
            NumerosParser parser = new NumerosParser(tokens);
            int val = parser.programa().val;

            System.out.println("Valor = "+val);
```

Demonstração 3 – Analisador semântico no ANTLR usando DDS L-atribuída

---

1. Criar um novo arquivo, no Desktop, com um programa de exemplo

```
(34 - 3 + 2) * (41 + 3)
```

2. Abrir o Antlrworks e criar a seguinte gramática

```
grammar Expressoes;

programa returns [ int val ]
	:	expressao EOF { $programa.val = $expressao.val; }
	;

expressao returns  [ int val ]
	:	termo expressao2[$termo.val] { $expressao.val =
$expressao2.sint; }
	;

expressao2 [ int her ] returns [ int val, int sint ]
	:	'+' termo exp=expressao2[$termo.val+$her] { $expressao2.sint =
$exp.sint; }
	|	'-' termo exp=expressao2[$her-$termo.val] { $expressao2.sint =
$exp.sint; }
	|	{ $expressao2.sint = $her; }
	;

termo returns [ int val ]
	:	fator termo2[$fator.val] { $termo.val = $termo2.sint; }
	;

termo2 [ int her ] returns [ int val, int sint ]
	:	'*' fator term=termo2[$fator.val*$her] { $termo2.sint =
$term.sint; }
	|	{ $termo2.sint = $her; }
	;

fator returns [ int val ]
	:	'(' expressao ')' { $fator.val = $expressao.val; }
	|	NUM { $fator.val = Integer.parseInt($NUM.getText()); }
	;

NUM	:	'0'..'9'+
	;

WS	:	( ' ' | '\n' | '\r' | '\t' ) { skip(); }
	;
```

3. Rodar num projeto do NetBeans (método main abaixo)

```
            ANTLRInputStream input = new ANTLRInputStream(new
```

```
FileInputStream("C:\\Users\\Daniel\\Desktop\\teste.txt"));
            ExpressoesLexer lexer = new ExpressoesLexer(input);
            CommonTokenStream tokens = new CommonTokenStream(lexer);
            ExpressoesParser parser = new ExpressoesParser(tokens);
            int val = parser.programa().val;
            System.out.println("Valor = "+val);
```

Demonstração 4 – Analisador semântico no ANTLR usando DDS não-L-atribuída

---

1. Criar um novo arquivo, no Desktop, com um programa de exemplo

```
53c
```

2. Abrir o Antlrworks e criar a seguinte gramática

```
grammar Numeros2;
programa returns [ int val ]
      :     numero[$sufixo.base] sufixo '\r'? '\n' { $programa.val =
$numero.val; }
      ;
sufixo returns [ int base ]
      :     'c' { $base = 100; }
      |     'm' { $base = 1000; }
      ;
numero[int base] returns [ int val ]
      :     digito n2=numero { $val = ($n2.val + $digito.val)*$base; }
      |     digito { $val = $digito.val*$base; }
      ;
digito returns [ int val ]
      :     '0' { $val = 0; }
      |     '1' { $val = 1; }
      |     '2' { $val = 2; }
      |     '3' { $val = 3; }
      |     '4' { $val = 4; }
      |     '5' { $val = 5; }
      |     '6' { $val = 6; }
      |     '7' { $val = 7; }
      |     '8' { $val = 8; }
      |     '9' { $val = 9; }
      ;
```

3. Mostrar que o ANTLR não deixa nem compilar
4. Modificar a gramática (agora virou S-atribuída)

```
grammar Numeros2;
programa returns [ int val ]
      :     numero sufixo '\r'? '\n' { $programa.val =
$numero.val*$sufixo.base; }
      ;
sufixo returns [ int base ]
      :     'c' { $base = 100; }
      |     'm' { $base = 1000; }
      ;
numero returns [ int val ]
      :     digito n2=numero { $val = $n2.val + $digito.val; }
      |     digito { $val = $digito.val; }
      ;
digito returns [ int val ]
```

```
      :     '0' { $val = 0; }
      |     '1' { $val = 1; }
      |     '2' { $val = 2; }
      |     '3' { $val = 3; }
      |     '4' { $val = 4; }
      |     '5' { $val = 5; }
      |     '6' { $val = 6; }
      |     '7' { $val = 7; }
      |     '8' { $val = 8; }
      |     '9' { $val = 9; }
      ;
```

5. Rodar num projeto do NetBeans (método main abaixo)

```
            ANTLRInputStream input = new ANTLRInputStream(new

FileInputStream("C:\\Users\\Daniel\\Desktop\\teste.txt"));
            Numeros2Lexer lexer = new Numeros2Lexer(input);
            CommonTokenStream tokens = new CommonTokenStream(lexer);
            Numeros2Parser parser = new Numeros2Parser(tokens);
            int val = parser.programa();
            System.out.println("Valor = "+val);
```

Demonstração 5 – Analisador semântico em duas passadas usando ANTLR

---

1. Criar um novo arquivo, no Desktop, com um programa de exemplo

```
53d
```

2. Abrir o Antlrworks e criar a seguinte gramática

```
grammar Numeros2;

Digito
      :     '0'
      |     '1'
      |     '2'
      |     '3'
      |     '4'
      |     '5'
      |     '6'
      |     '7'
      |     '8'
      |     '9'
      ;

programa
      :     numero sufixo '\r'? '\n'
      ;
sufixo
      :     'b'
```

```
      |     'o'
      |     'd'
      ;
numero
      :     Digito numero
      |     Digito
      ;
```

3. Testar no NetBeans (código main abaixo)

```
        ANTLRInputStream input = new ANTLRInputStream(
                new FileInputStream("<caminho para>/teste.txt"));
        Numeros2Lexer lexer = new Numeros2Lexer(input);
        CommonTokenStream tokens = new CommonTokenStream(lexer);
        Numeros2Parser parser = new Numeros2Parser(tokens);
        ProgramaContext programa = parser.programa();
        String sufixo = programa.sufixo().getText();
        String strNumero = programa.numero().getText();
        int base = 0;
        if(sufixo.equals("b")) {
            base = 2;
        } else if(sufixo.equals("o")){
            base = 8;
        } else {
            base = 10;
        }

        int numero = Integer.parseInt(strNumero, base);

        System.out.println("Numero: "+numero);
```