LINGUAGENS FORMAIS E AUTÓMATOS

TRABALHO PRÁTICO

UNIVERSIDADE DE AVEIRO

DEPARTAMENTO DE ELETRÓNICA

TELECOMUNICAÇÕES E INFORMÁTICA

2018-2019, 2º SEMESTRE

Rui Miguel da Silva Oliveira [89216],

Gonçalo Filipe Figueiredo Perna [88823],

André Ribeiro Almeida [88960],

Rui Pedro Pereira Santos [89293],

José Pedro Fonseca da Silva [89195]

CONTEÚDO

INTRODUÇÃO

OBJETIVOS

No âmbito da disciplina de linguagens formais e autómatos foi proposto a realização de duas linguagens, uma para um compilador e outra para ler informação estruturada.

Deste modo seria necessário definir a sintaxe e a semântica da linguagem a programar, implementar em ANTLR4 as respetivas análises, definir as regras semânticas da linguagem, criar exemplos de programas e definir os padrões de geração de código para as instruções da linguagem.

Assim pretende-se uma linguagem bem expressa e simples, gramáticas funcionais, análise semântica, corrigir e gerir erros, escrita de código legível e por fim geração de código.

TEMA

O tema escolhido pelo nosso grupo foi uma linguagem para análise dimensional (física). Assim seria necessário definir variáveis, instruções iterativas, expressões booleanas e condicionais, funções e operações interativas com o utilizador.

DIVISÃO DO TRABALHO

O trabalho foi dividido essencialmente em cinco partes.

O aluno Rui Oliveira ficou encarregue da gramática do compilador e relatório, os alunos Rui Santos e André Almeida pela estrutura de dados, o interpretador de tipos e respetivo *visitor* ficou ao cargo do aluno José Silva e o aluno Gonçalo Perna fez *visitor* do compilador e análise semântica.

“MANUAL DE INSTRUÇÕES”

COMPILADOR

A gramática do compilador consiste num conjunto de possíveis operações que podem ser recebidas e futuramente tratadas. A regra **stat** tem a possibilidade de escolher entre diversas opções como está ilustrado na figura ao lado.

```
1   grammar Compiler;
2
3   program : (stat)* end
4           ;
5
6   stat    : define
7           | update
8           | show
9           | decisao
10          | cicloWhile
11          | function
12          | callFunction
13          | increment
14          ;
```

Assim existe a podemos **definir** uma como “Distancia d1”, atribuir-lhe um possível valor por exemplo “Distancia d1 = 2.5”, ou até uma **operação** do género “Distancia d2 = 2*K - (-2)”. Visto que **Distancia** foi definido com o símbolo m no ficheiro de declaração (Distancia [m]: Double), a variável d1 fica com o valor 2.5m porque é uma distância e a variável com 2002m pois a grandeza K corresponde a 10^3.

A regra **update** permite, a uma variável já definida, atribuir um novo valor do género “t1 = 4”.

Com o **show** torna-se possível imprimir qualquer string do tipo '"' .*? '"' ou seja, qualquer entrada dentro de aspas ou também imprimir uma **operação**.

Com regra **decisao** conseguimos tratar de expressões como o if, else if, else e suas respetivas combinações válidas. Exemplo: “if(m1==m2){show d1+d2}”.

A gramática do compilador permite também trabalhar com ciclos do tipo while. Com a regra **cicloWhile** é possível fazer algo do tipo “while(m1<m2){show m1 m1++}”

Funções (void ou não) são outra possibilidade para o uso desta linguagem de física. A sintaxe da regra **funtion** na definição é do género “Distancia soma (Distancia a, Distancia b) { return a+b }”. Com a regra **callFunction** efetuamos a sua chamada “Distancia = soma(a,b)” (aplicação da regra define com funções).

Nota1: cada função só pode receber um e só um return.

Nota2: as funções têm de ser todas definidas no início do programa.

INTERPRETADOR DE TIPOS

A gramática TypesInterpreter.g4 consiste na análise de um ficheiro de texto com um conjunto de declarações em linhas diferentes. Como foi explicado no compilador, a sintaxe é do género **Distancia [m]: Double.**

O corpo da atribuição aceita desde operações de multiplicar e divisão e até expoentes.

Nota: numa nova atribuição complexa aceita-se apenas uma variável simples sem expoente, no entanto é tratado em analise semântica este caso específico que é sintaticamente errado.

Na declaração de constantes na ausência de [identificador] considera-se como um escalar.

EXEMPLOS DE ERROS SEMANTICOS

Com o intuito de testar os erros das gramáticas e outros ficheiros, foram criados ficheiros de texto com exemplo de possíveis erros.

Deste modo tenta-se "quebrar" as regras semânticas da linguagem e consequentemente *print* do(s) erro(s) encontrado(s).

- teste1: definição de funções
- teste2: chamada de funções
- teste3: instruções if e while
- teste4: demonstração do uso de contextos
- teste5: declarações e tipos de variáveis
- teste6: erros semânticos do interpretador

CONCLUSÃO

CONTRIBUIÇÃO DOS AUTORES

Consideramos que o trabalho foi distribuído homogeneamente e atribuímos uma participação de 20% a todos os autores.