Relatório do Projeto de LFA

Leandro Rito,
Lúcia Sousa,
Raquel Pinto,
Rodrigo Martins

Relatório do Projeto de LFA

Departamento de Eletrónica, Telecomunicações e Informática
Universidade de Aveiro

Linguagens Formais e Autómatos

Leandro Rito,
Lúcia Sousa,
Raquel Pinto,
Rodrigo Martins
(92975) leandro.rito@ua.pt,
(93086) luciamsousa00@ua.pt,
(92948) raq.milh@ua.pt,
(93264) rodrigomartins@ua.pt

19/06/2020

Resumo

Neste relatório, descreve-se e apresenta-se a implementação de uma linguagem de programação para análise dimensional. O projeto é constituído por duas linguagens, a primeira para ser interpretada onde se definem as dimensões e unidades. A segunda linguagem seria a compilada, onde é implementado um analisador semântico para detetar e tratar erros. São apresentados os objetivos do projeto, assim como as dificuldades e problemas ao longo do mesmo.

Conteúdo

Capítulo 1

Objetivos

O objetivo deste projeto é desenvolver uma linguagem que aceite definir dimensões e unidades. Onde seja possível a criação de dimensões compostas e relações entre as unidades. O propósito seria permitir cálculos entre valores da mesma dimensão, e caso as unidades fossem diferentes, fazer a sua conversão, através da relação anteriormente designada.

Foi necessário criar duas linguagens. A primeira, à qual demos o nome de configDimUni, onde iríamos definir as dimensões e unidades e as relações entre as últimas. A segunda linguagem, analiseDimensional, seria para efetuar os cálculos para dimensões iguais e converter unidades.

Capítulo 2

Compilador e Análise Semântica

O compilador foi gerado para a gramática analiseDimensional.g4, utilizando o teste AnaliseExample.txt que se encontra na pasta de testsAnalise, é gerado um ficheiro Output.java para onde é copiado o código compilado. Para utilizar o compiler é necessário recompilar a gramática e todos os ficheiros java.

A análise semântica encontra-se em comentário na main da gramática, pois não se encontra a funcionar corretamente. Para a sua utilização basta tirar de comentário na main, e utilizar o mesmo exemplo do compilador.

Capítulo 3

Instruções

3.1 Definição de Unidades e Dimensões

DefineDim:

Uso: DefineDim <Dimensão> (<Tipo>,<Sinal>) : <UnidadeBase>, <OutraUnidade> (<expressao>)

Usado para definir uma dimensão nova, o tipo especifica se é uma dimensão que só aceita números reais ou inteiros, o sinal serve para especificar se a dimensão apenas aceita números negativos ou apenas números positivos, se este valor não for especificado a dimensão vai aceitar por padrão números positivos e negativos. A unidade base serve para especificar a unidade padrão da dimensão, sendo possível associar mais unidades a essa dimensão, esta ultima funcionalidade é opcional, a expressão serve para definir a relação entre a unidade padrão e a outra unidade, para que depois se possa converter a outra unidade na unidade padrão. Esta relação tem de ser definida com base num polinómio ordenado. Exemplos:

- DefineDim length(int, pos) : metres,miles(miles/0,000621),cm(0.01cm)
- DefineDim temperature(real, pos) : celsius, Kelvin(K-273.15)

DefineUni:

Uso: DefineUni <Unidade> (<expressao>) : <Dimensao>

Usado para associar uma unidade a uma dimensão já declarada. Exemplos:

- DefineUni mm(0.001mm), length
- DefineUni km(1000km), length

3.2 Ciclos e Funções

Funções

function

Uso: function <tipo> nome (<expressao>) { *instruçoes* return <variavel>}

Usado para definir funções similarmente à linguagem Java.

Ciclos

loopFor

Uso: for (<variavel> = <valor>; <expressao1> ; <expressao2>) { *instruções* }

Usado para definir um ciclo for similarmente à linguagem Java. A variável corresponde à variável de contagem, a expressão1 corresponde à condição que permite a paragem do ciclo, a expressao2 corresponde ao incremento da variável de contagem na forma(variável = variável + x).

loopWhile

Uso: while (<expressao>) { *instruçoes* }

Usado para definir um ciclo while similarmente à linguagem Java.

doWhile

Uso: do { *instruçoes* } while(<expressao>)

Usado para definir um ciclo do while similarmente à linguagem Java.

ifCond

Uso: if <expressao> then *instruçoes* (else *instruçoes*) end

Usado para definir uma função "if"similarmente à linguagem Java. A expressão corresponde a uma expressão booleana que, dependendo do valor lógico, levará à realização de instruções. A palavra reservada end será utilizada para terminar esta função.

Declarações e outras instruções

print

Uso: print <variavel>

Usado para imprimir no terminal a variável.

declaration

Uso: <tipo> <variavel>

Usado para declarar um variável sem valor associado.

assignment

Uso: declaration = <expressao>

Usado para declarar uma variável e associar-lhe um valor.

importFile

Uso: import <Ficheiro>

Usado para importar um ficheiro. Esta instrução é importante pois permite definir dimensões e unidades num ficheiro à parte e depois utiliza-las noutro ficheiro sem ter de as declarar de novo.

Capítulo 4

Resultados e Problemas

Este capítulo tem como objetivo expor os problemas que se encontram ao longo do projeto e tentar propor uma explicação para os mesmos.

Os testes utilizados para testar a gramática analiseDimensional e o compilador encontram-se na pasta testsAnalise e para a gramática configDimUni os testes estão na pasta testsConfig.

O compilador é apenas um compilador simples, não há qualquer ligação com a linguagem para configuração de dimensões e unidades.

A analise semântica apresenta erros, estes erros são relativos, ao que pensamos ser, da symbol table, podendo esta última não estar devidamente implementada e por essa razão não guardar as variáveis utilizadas.

Como não conseguimos testar o Interpreter, não conseguimos determinar o que consegue fazer ou não. Durante o desenvolvimento do mesmo deparámo-nos com diversos problemas, como por exemplo na elaboração de diversas funções.

Capítulo 5

Conclusão

Concluindo, ambas as gramáticas estão desenvolvidas para fazer o que foi proposto. O interpreter, o compiler e a análise semântica foram implementados de forma a conseguir realizar o objetivo do projeto, no entanto, é evidente a presença de alguns erros.

Capítulo 6

Contribuição dos Autores

O projeto foi realizado quase na sua totalidade em "LiveShare", ou seja, todos os integrantes do grupo estavam a trabalhar ao mesmo tempo e muitas das vezes estavamos a contribuir para a mesma coisa.

- Guilherme Lopes: 0%
- Leandro Rito: 22%
  - Desenvolvimento das duas gramáticas, criação de testes e elaboração do relatório.
- Lúcia Sousa: 26%
  - Desenvolvimento das duas gramáticas, do compiler e da análise semântica. Error handling e symbol table. Criação de testes. Elaboração do relatório.
- Raquel Pinto: 26%
  - Desenvolvimento das duas gramáticas, do compiler e da análise semântica. Criação de testes. Elaboração do relatório. Assistência no desenvolvimento das classes.
- Rodrigo Martins: 26%
  - Desenvolvimento das duas gramáticas, classes da Dimensão, Unidades e Dimensões compostas, criação de testes, desenvolvimento do Interpreter e elaboração do relatório.