Dimensional analysis language (DAL)

Universidade de Aveiro (UA)

Afonso Teixeira, Miguel Cabral,
Bernardo Barreto, João Oliveira

DAL

Departamento de Eletrónica, Telecomunicações e Informática

UA

Afonso Teixeira, Miguel Cabral,
Bernardo Barreto, João Oliveira
(93170) apteixeira@ua.pt, (93091) miguel.f.cabral@ua.pt,
(93272) bernardo.barreto@ua.pt,(93282) joaoroliveira@ua.pt

Junho de 2020

Queremos agradecer a todos os docentes que leccionam a Unidade Curricular (UC) de Linguas Formais e Autómatos (LFA) que no conjunto deste semestre nos deram os conhecimentos para a realização deste projecto.

Conteúdo

Capítulo 1

Introdução

Este relatório foi elaborado com o propósito de descrever e analisar o projecto realizado no âmbito da UC de LFA.

Este documento está dividido em quatro capítulos. Depois desta introdução, são apresentadas as instruções para a utilização da nossa linguagem depois no Capítulo 3 é analisada e explicada a nossa linguagem e posteriormente no Capítulo 4 é explicado como podemos configurar novas dimensões.

Capítulo 2

Instruções de Utilização

Para compilar é necessário correr os seguintes comandos a partir do terminal: **java -ea MainGrammarMain Output progTest/DimensoesCONFIG.txt "programa.txt"** em que o "programa.txt"é o programa que queremos correr, seguidamente é só fazer **javac Output.java** para compilar o ficheiro em java e finalmente correr o programam com **java Output**.

Capítulo 3

Linguagem

A DAL é uma linguagem compilada cuja linguagem destino é Java. A análise sintáctica e semântica é feita em **antlr4** .

3.1 Declaração de Variáveis

A declaração de variáveis é deita através do **declarations** pode ser do tipo **int, real, boolean** e **unit**, o **ID** tem que começar por um caractere minúsculo e pode conter caracteres maiúsculos seguido do caractere '=' sendo depois atribuída uma expressão. Esta é finalizada pelo caractere ';' :

```
declarations:  type ID '=' expression;

type returns [Type t_type]:
'int'        #IntType
| 'real'     #RealType
| 'unit'     #UnitType
| 'boolean' #BooleanType
;
```

Exemplo:

```
unit ola = 15 kg;
int i = 1;
int quarentena = 31;
boolean teste = false;
```

3.2 Alteração de Variáveis

Após a inicialização de uma variável é possível alterar o seu valor usando o **assignments** dando o **ID** da variável seguido do '=' e a expressão. Sendo finalizada pelo caractere ';' :

```
assignments:  ID '=' expression;
```

Exemplo:

```
int i = 1+3;
int quarentena = 60;
int i = 6*7;
```

3.3 Expressões

A **expression** pode ser uma **expression** entre parêntesis, a soma/subtracção/multiplicação/divisão de duas **expressions**, uma condição entre duas **expressions**, a incrementação ou o decremento de um **ID**, um **unit**, um **real**, um **boolean** ou um **ID**.

```
expression returns[Type e_type, String var]:  '(' e=expression ')'
#parenExpr
| e1=expression op=('*' | '/') e2=expression #multDivExpr
| e1=expression op=('+' | '-') e2=expression #addSubExpr
| e1=expression op=OP_COND e2=expression   #conditionalExpr
| e=ID op=('++' | '-')                  #incrementsExpr
| e=(INT|REAL) unit=ID     #unitVarExpr
| INT        #intExpr
| REAL        #realExpr
| BOOLEAN     #booleanExpr
| ID         #idExpr
;
```

3.4 Block

O block é um conjunto de um ou mais stats que estão entre os caracteres ""e .

```
block:  ''  stat* '';
```

3.5 If Clause

O **If_clause** é começado pelo conjunto de caracteres 'if (' seguido de uma **expression** e depois para fechar o caracter ')', em seguida o **block** que respeita a condição do if. Esta **If_clause** também pode conter um **else** seguido do **block** que respeita essa condição.

```
if_clause:'if' '(' expression ')' blockIf = block ('else' blockElse
= block) ?;
```

Exemplo:

```
int i = 1;
if(i == 0) {
print(i);
i++;
}else {
print(i);
i++;
}
```

3.6 Loops

Também é permitida a criação de Loops estes podem ser ciclos **while**, **for** ou do **while**.
Os ciclos **while** são começados pelo conjunto de caracteres 'while (' seguido de uma **expression** depois o fecho é feito com ')' seguido do **block** que será repetido enquanto a condição for verdadeira.
Os ciclos **for** são começados por 'for (' seguidos de um **declarations** e depois duas **expressions** tudo separado pelo caractere ';' e terminado pelo caractere ')' seguido do **block** que será repetido enquanto a condição for verdadeira.
Os **do while** são começados por 'do' seguido de um **block** e depois o conjunto de caracteres 'while (' e em seguida uma **expression** seguida de ') ;'

```
loop:
'for' '(' declarations ';' expression ';' expression ')'
block                 #forLoop
| 'while' '(' expression ')' block     #whileLoop
| 'do' block 'while' '(' expression ')' ';'     #doWhileLoop
;
```

Exemplo:

```
while(i < quarentena) {
peso = peso + aumentodepeso;
i++;
}
```

```
for(int j = 0;j<quarentena;j++){
peso = peso + aumentodepeso;
}
do{ peso = peso + aumentodepeso;
i++;
}while(i < quarentena);
```

3.7 Prints

Para realizar um **print** é só começar por escrever 'print (' seguido da expression a imprimir e fechando com ');'

```
prints:  'print' '(' expression ')';
```

Exemplo:
`print(j);`

3.8 Comentários

A Linguagem admite comentários de uma linha com os caracteres '//' e multi linha com '/*' e ' */', ou seja tudo o que tiver entre estes dois caracteres é um comentário.

```
COMMENT: '//'.*?  ('\ n'|EOF) -> skip;
COMMENTMULTILINE: '/*' .*?  '*/' -> skip;
```

3.9 Palavras reservadas

if	while	real	else
do	unit	for	int
boolean	dim	true	false

Capítulo 4

Configuração de dimensões

4.1 Criação de unidades

Para definir novas unidades é só adicionar ao **DimensoesCNFIG.txt** o seguinte: 'dim' seguido do nome da dimensão depois '->' a unidade SI em seguida outra '->' e o símbolo .

```
dim:  'dim' dimension=ID '->' unit=ID '->' symbol=ID;
```

Exemplo:

```
dim Distancia -> Metro -> m;
```

4.2 Criação de uma nova unidade de conversão

No ficheiro **DimensoesCNFIG.txt** adicionar o nome do submúltiplo depois o caractere '-' em seguida o símbolo desse submúltiplo depois '»' e o valor correspondente na unidade SI.

```
valueConvertion:  fullName=ID '-' dest=ID '»' value=(INT|REAL) src=ID;
```

Exemplo:

```
Centimetro - cm » 0.01 m;
```

Contribuições dos autores

- **Afonso Teixeira** - 25 %
  - gramática Maingrammar
  - MainCompiler
  - String Template
  - MainSemCheck
  - UnitSemCeck
  - Classes auxiliares
- **Miguel Cabral** - 25 %
  - gramática Maingrammar
  - MainCompiler
  - String Template
  - MainSemCheck
  - Relatório
  - Classes auxiliares
- **Bernardo Barreto** - 25 %
  - gramática Maingrammar
  - gramática UnitGrammar
  - String Template
  - UnitInterpreter
  - UnitSemCeck
  - Classes auxiliares
- **João Oliveira** - 25 %
  - gramática Maingrammar
  - gramática UnitGrammar
  - String Template

- UnitInterpreter
- UnitSemCeck
- Classes auxiliares

Acrónimos

LFA Linguas Formais e Autómatos

UA Universidade de Aveiro

UC Unidade Curricular

DAL Dimensional analysis language