The Potatoes Project 2.0

Luis Pedro Moura

July 2018

0.1 Introduction

0.2 Unit Types definition language

A linguagem de definição de tipos, definida pelas gramáticas do ficheiro *Types.g4*, define as instruções necessárias para a definição de tipos e de prefixos[1].

0.2.1 Palavras Reservadas

The language reserved words are: *types* e *prefixes*.

0.2.2 Instruções básicas e comentários

All source code files for this language must contain the instruction:

```
types {
    ...
}
```

Listing 1: Linguagem de Definição de Tipos - Instrução `types {...}`

and can optionally contain the instruction:

```
prefixes {
     ...
}
```

Listing 2: Linguagem de Definição de Tipos - Instrução `prefixes {...}`

before or after the instruction `types`.

Terminação de instruções

Nenhuma instrução necessita de um terminador.

Comentários

São suportados comentários *in-line*, iniciados pela sequência `//` (equivalente à linguagem Java). Não são suportados comentários *multiline*.

[1]Deprecated. Não há análise semântica para o uso de prefixos na linguagem general purpose e nenhum exemplo fornecido para a linguagem general purpose utiliza prefixos.

0.2.3 Unit/Types Definition

Inside the instruction `units {...}`, can be defined zero or more units or dimensions using distinct instructions:

Base Unit (Numeric)

A Base Unit is an independent Unit that represents a numeric measurement in any one-dimensional context.

```
<ID> "<STRING>"
```

Listing 3: Declaration of Base Unit (Numeric) instruction

in which:

- `ID` represents the Unit Type name.
- `STRING` represents the Unit Type symbol. This field is optional.

For example:

```
meter "m";
```

represents a Unit called *meter*, with the symbol *m*.

Derived Unit

The derived Units are formed by powers, products or quotients of the Base Units. Although the combinations are unlimited, the goal is to define the ones that will be used later in a programming environment. A later definition of equivalent Units will allow to expand the defined combinations of Base Units.

```
<ID> "<STRING>" : <OPERATION BETWEEN PRE-EXISTING TYPES>
```

Listing 4: Declaration of Derived Unit

in which:

- `OPERATION BETWEEN PRE-EXISTING TYPES` represents 1 or more multiplications (`typeA * typeB`), quotients (`typeA / typeB`) and powers (`typeA ^ <INT>`) between existing unit types.

For example:

```
velocity "v" : meter / second;
```

represents a unit type called *velocity*, with the symbol *v*, defined as the division between the Units *meter* and *second*.

```
acceleration "m/s^2" : meter / second^2;
OR
acceleration "m/s^2" : velocity / second;
```

represents a Unit Type called *acceleration*, with the symbol m/s^2, defined as the quotient between the Unit *meter* and the square of the Unit *second*.

Dimensions and Equivalent Units

A Base Unit represents a measurement in any one-dimensional context. But there may be infinite systems used to measure in the same dimension (distance can be measured with meters, yards or any other made up unit). This means that all these measurements will have a relation, given the correct conversion factor, as they are in fact equivalent.

```
<ID> [DIMENSION BASE UNIT] : (<REAL NUMBER>) <EQUIVALENT UNIT> | (<REAL NUMBER>) <
    EQUIVALENT UNIT> ...
```

Listing 5: Declaration of Dimensions and Equivalent Units

in which:

- `<ID>` represents the name of the dimension.

- `<DIMENSION BASE UNIT>` represents the Unit that will be main representative of the dimension, and will be the default Unit to convert to unless otherwise specified.
- the pattern `| (<REAL NUMBER>) <EQUIVALENT UNIT>` can be repeated zero or more times.
- `<REAL VALUE>` represents a real number or an operation between real numbers. Supported operations are power, multiplication, division, addition, subtraction ans association with parenthesis. This real number **represents the conversion factor from the Unit `DIMENSION BASE UNIT` to the Unit `EQUIVALENT UNIT`.**
- `<EQUIVALENT UNIT>` represents the name of the equivalent unit.

For example:

```
length "L" [meter] : (10 + 29.37)inch | (1.094)yard;
```

represents a Unit called *distance*, with the optional symbol *d*, represented by the unit *meter*, in which *1 meter* is equivalent to $(10 + 29.37)$ `inch` or (1.094) `yard`. In natural language, it's equivalent to say that *1 meter equals 39.37 inches or 1 yard equals 36 inches (the conversion factor between yard and inch is implied in the definition (precision must be guaranteed by the user)).*

0.2.4 Definição de Prefixos

Dentro da instrução `prefixes {...}`, podem ser definidos 0 ou mais prefixos. Cada prefixo pode ser definido através da seguinte instrução:

```
<ID> "<STRING>" : <REAL VALUE>
```

Listing 6: Linguagem de Definição de Tipos - Instrução de definição de prefixo

em que:

- `ID` representa o nome do prefixo.
- `STRING` representa o nome de impressão do prefixo.
- `REAL VALUE` representa um número real ou uma operação entre números reais. As operações suportadas são as mesmas das referidas na sub-subsecção **??**.

Por exemplo:

```
DECA "da" : 10^1
```

representa um prefixo chamado *DECA*, com nome de impressão *da*, definida como sendo o valor 10^1.