Compiladores

Linguagem *DimAna*

Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro

Abril de 2023

Objectivos

O objectivo geral deste trabalho é o desenvolvimento de uma linguagem de programação compilada – i.e. que crie programas equivalentes ao programa a compilar numa linguagem de programação genérica (Java, C++, Python, ...) – que permita o uso de análise dimensional[1] nas expressões numéricas (inteiras e reais).

A inspiração para esta linguagem deriva – naturalmente – da física onde a análise dimensional serve para garantir sanidade nas expressões matemáticas aplicáveis (por forma a não permitir, por exemplo, que se some uma distância a uma velocidade). No entanto, a análise dimensional pode (e deve) ser aplicada a muitas outras áreas que utilizem quantidades numéricas para representar coisas diferentes (por exemplo, números mecanográficos e notas como acontece no primeiro exemplo dado).

Assim, pretende-se estender o sistema de tipos de uma linguagem de programação (tanto quanto baste, de uso geral) com a possibilidade de definir dimensões distintas (e interoperáveis) a expressões numéricas (e como tal, a variáveis e outras entidades com tipos da linguagem). Por exemplo, poder definir tipos de dados distância e tempo (expressáveis, por exemplo, com unidades metro [m] e segundo [s]) e poder definir um novo tipo de dados velocidade como sendo distância/tempo [m/s]. O sistema de tipos da linguagem não só deve permitir álgebra sobre dimensões existentes, com validar a respectiva correcção (atribuir e/ou somar distância com tempo não deve ser permitido, muito embora ambos sejam números).

Seria necessário aplicar uma álgebra dimensional (neste projecto, aplicável apenas a números inteiros e reais) com as seguintes regras:

- comparações, atribuições de valor, somas e subtracções apenas para a mesma dimensão;
- multiplicações e divisões aplicáveis a expressões dimensionais a gerar outra dimensão;
- multiplicações e divisões por expressões adimensionais a manter a mesma dimensão;

[1] `https://en.wikipedia.org/wiki/Dimensional_analysis`

Neste trabalho a parte da análise semântica (estática, isto é em tempo de compilação) é particularmente importante e desafiante. Será nesta fase se fará uma verificação dimensional completa.

Características da solução

Apresentam-se a seguir um conjunto de características que a solução desenvolvida pode ou deve contemplar. Essas características estão classificadas a 3 níveis:

- mínima – característica que a solução tem obrigatoriamente que implementar;
- desejável – característica não obrigatória, mas fortemente desejável que seja implementada pela solução (apenas considerada se as mínimas forem cumpridas);
- avançada – característica adicional apenas considerada para avaliação se as obrigatórias e as desejáveis tiverem sido contempladas na solução.

Características mínimas

Os exemplos `grades.da`, `example1.da`, `example2.da`, `physics.da` e `example3.da` indicam algum código fonte que tem de ser aceite (e devidamente compilado) pela linguagem a desenvolver.

A linguagem deve implementar:

- Instrução para definir uma nova dimensão. Esta definição pode ser independente (isto é definir de raiz uma nova dimensão), ou dependente. Neste último caso a definição usa uma expressão aritmética envolvendo apenas multiplicações, divisões e/ou potências envolvendo dimensões já definidas (ver exemplos). Em qualquer um deste case, uma dimensão funciona como um novo tipo de dados numérico (assente em inteiros ou em reais). A definição duma dimensão envolve também a definição da sua unidade base (por exemplo, a unidade *metro* para a dimensão *distância*), e, eventualmente, um sufixo (por exemplo, o sufixo *m* para a unidade *metro*).
- Instrução para definir outra unidade para um dimensão existente (por exemplo, *polegadas* para a dimensão *distância*).
- Os tipo de de dados inteiro, real, texto e lista (ver exemplo).
- Aceitar expressões aritméticas standard para os tipos de dados numéricos. Aceita a operação de concatenação de texto (operador da soma).
- Instrução de escrita no *standard output*.

- Instrução de leitura de texto a partir do *standard input*.
- Operadores de conversão entre tipos de dados (por exemplo, `string(10)` para converter para texto; ou `integer("10")` para converter para inteiro).
- Instrução para adicionar elemento no fim de uma lista.
- Operadores para aceder a elementos de uma lista (similar ao uso de arrays nativos em Java com a diferença do primeiro elemento ter o índice 1).
- Instrução de iteração (loop) entre dois valores inteiros.
- Instrução estática (isto é, apenas com significado em tempo de compilação) para incluir o conteúdo de outro ficheiro (semanticamente similar ao *include* da linguagem *C*)
- Verificação semântica do sistema de tipos com análise dimensional.

Características desejáveis

- Permitir a definição de expressões booleanas (predicados) contendo, pelo menos relações de ordem e operadores booleanos (conjunção, disjunção, etc.).
- Incluir a instrução condicional (operando sobre expressões booleanas).
- Incluir outras instrução repetitivas (operando sobre expressões booleanas).
- Permitir a definição de valores dimensionais literais com sufixo (por exemplo: `Length l = 5m;`).
- Instrução prefixa (ver ficheiro `prefix.da`) e aplicação de prefixos nos valores literais (por exemplo: `Length l = 5cm;`).
- Permitir a soma/subtracção de valores com a mesma dimensão, mas com unidades diferentes (por exemplo: `Length l = 5m - 5cm;`).

Características avançadas

- Implementar funções e variáveis locais às mesmas.
- Implementar estruturas.
- ...