Projeto de Compiladores – Etapa 5 (E5)

Rodrigo Kassick

2016-2

1 Descrição

Implementar a **declaração de símbolos** através da primitiva `let` da linguagem μmML. A declaração de um símbolo deve associar um *nome* às seguintes informações:

- **Posição na memória**. Cada símbolo ocupa uma determinada posição na memória durante a execução do programa. Ao declarar um símbolo, deve ser escolhida uma posição para ele.

  Considere que o **tamanho** de **qualquer tipo de símbolo** é sempre `1`.

- **Tipo de Dados.** Cada símbolo terá um tipo derivado a partir da expressão do lado direito do sinal de `=`:

  - `x = 1 + 2.0` : `x` tem tipo `float`
  - `y = 1 + 2` : `y` tem tipo `int`
  - z = ”abc”::”def”: `z` tem tipo `str`

Para esta etapa, deve estar implementado:

- Verificação de tipos para operação de **concatenação** (::), **apenas** para strings
- Declaração de símbolos com a primitiva `let`
- Produção `metaexpr` $\rightarrow$ `symbol` deve procurar o símbolo na tabela de símbolos atual.
- Ao fim da lista de declarações de uma regra `let` (antes do `in`), deve ser apresentado na tela a tabela de símbolos atual, com seus nomes, tipos e posições.

2 Dicas

- É fornecido um código que gerencia tabelas de símbolos hierárquicas. Você pode utilizá-lo *as-it-is* ou utilizar qualquer outro código (seu ou de terceiros) para gerenciar as tabelas.
- A linguagem μmML **exige** tabelas de símbolos hierárquicas.
- Durante a avaliação de uma produção `metaexpr` $\rightarrow$ `symbol`, é necessário consultar uma tabela de símbolos **atual**. Mantenha uma variável *global* que representa a *tabela de símbolos corrente* e a atualize no início e fim da produção `let`.