Exercício prático 05

BCC328 - Construção de Compiladores I

Prof. Rodrigo Ribeiro

14-07-2025

Construção de um interpretador para a linguagem L2

O objetivo desta atividade é a implementação de um interpretador para a linguagem L2, que adiciona escopos e variáveis imutáveis a L1.

A Linguagem L2

A linguagem L2 consiste na extensão de L1 por permitir a definição de variáveis imutáveis e de seu escopo de visibilidade:

```
def v := e in
   block
end
```

A construção anterior define a variável `v` como um nome para a expressão `e` e esta variável é visível dentro do escopo dos comandos representados por `block`. Como exemplo concreto, considere o seguinte trecho de código:

```
def x := 1 in
  a := x + 1 ;
  print(a);
  def x := 5 in
    a := x - 1;
    print(a);
  end
  print(x);
end
```

No trecho anterior temos que a variável `x` tem o valor `1` nos comandos

```
a := x + 1;
print(a);
```

e o valor impresso pelo comando `print(a)` será 2.]

Em seguida, temos o bloco

```
def x := 5 in
  a := x - 1;
  print(a);
end
```

no qual a variável `x` terá o valor 5 nos comandos

```
a := x - 1;
print(a);
```

Dessa forma, o comando `print(a)` irá imprimir o valor 4 no console. É importante notar que variáveis imutáveis tem o valor definido no escopo de sua definição. Isso quer dizer que o comando `print(x)` irá imprimir o valor 1, que é o definido para a varável `x` no escopo deste comando.

Sintaxe da linguagem L2

A sintaxe da linguagem L1 é definida pela seguinte gramática livre de contexto:

$$
\begin{array}{rcl}
P & \rightarrow & B \\
B & \rightarrow & S\,B \mid \lambda \\
S & \rightarrow & v := E; \\
 & \mid & read(E, v); \\
 & \mid & print(E); \\
 & \mid & def\ v := E\ in\ P\ end \\
E & \rightarrow & n \\
 & \mid & v \\
 & \mid & s \\
 & \mid & E + E \\
 & \mid & E - E \\
 & \mid & E * E \\
 & \mid & E\ E
\end{array}
$$

A gramática é formada por quatro variáveis: P, B, S e E; e pelos seguintes tokens (símbolos do alfabeto):

- def: inicia a declaração de uma variável imutável.
- in: marca o início do bloco de escopo de uma variável imutável.
- end: encerra o bloco de escopo de uma variável imutável.
- v: representam identificadores. O token de identificador segue as regras usuais presentes em linguagens de programação: um identitificador começa com uma letra seguida de uma sequência de zero ou mais dígitos ou letras.
- n: representam constantes numéricas. No momento, vamos suportar apenas números inteiros (tanto positivos, quanto negativos).
- s: representam literais de strings. A linguagem L2 utiliza aspas duplas para delimitar literais de string.

A sintaxe abstrata de L2 é representada pelos seguintes tipos de dados:

```
data L2
  = L2 [S2]

data S2
  = Def Var E2 [S2]
  | LRead String Var
  | LPrint E2
  | LAssign Var E2

data E2
  = LVal Value
  | LVar Var
  | LAdd E2 E2
  | LMinus E2 E2
  | LMul E2 E2
  | LDiv E2 E2
```

O tipo `L2` representa a variável P, `S2` denota a variável S e `E2` representa a variável E da gramática de L2.

Semântica de L2

A semântica de L2 é exatamente a de L1 com novas regras para lidar com variáveis imutáveis. Para isso, vamos introduzir um novo ambiente para armazenar os valores deste tipo de variável. Vamos adotar a variável φ para representar esse ambiente de variáveis imutáveis.

Vamos modificar a regra de variável, para dar suporte a variáveis imutáveis. A notação $\varphi(v) = \perp$ denota que não há valor associado a variável v no ambiente φ.

$$\frac{\varphi(v) = \perp \quad \sigma(v) = n}{\varphi; \sigma; v \Downarrow n}$$

$$\frac{\varphi(v) = n}{\varphi; \sigma; v \Downarrow n}$$

A regra para lidar com definições de variáveis imutáveis é como se segue:

$$\frac{\varphi; \sigma; e \Downarrow n \quad \varphi' = \varphi[v \mapsto n] \quad \varphi'; \sigma; B \Downarrow \sigma'}{\varphi; \sigma; def\ v := e\ in\ B \Downarrow \varphi; \sigma'}$$

Detalhes da entrega

O que deverá ser implementado

Você deverá implementar:

- Analisador léxico para L2.
- Analisador sintático para L2.
- Interpretador para L2.

A seguir, detalharemos a estrutura pré-definida do projeto para L2. A primeira função `lexerOnly` deve realizar a análise léxica sobre o arquivo de entrada e imprimir os tokens encontrados, como feito para a implementação de L1, em exercícios anteriores.

```
lexerOnly :: FilePath -> IO ()
lexerOnly file = error "Not implemented!"
```

A segunda função, `parserOnly`, deve realizar a análise sintática sobre o arquivo de entrada e imprimir a árvores de sintaxe produzida, como feito para a implementação de L1.

```
parserOnly :: FilePath -> IO ()
parserOnly file = error "Not implemented!"
```

Finalmente, a última função, `interpret`, deve realizar a interpretação do programa contido no arquivo fonte fornecido. Para isso, você deverá executar a análise léxica, sintática e executar o programa representado pela árvore produzida pelo analisador sintático de L2.

```
interpret :: FilePath -> IO ()
interpret file = error "Not implemented!"
```

todas essa funções estão presentes no arquivo src/L2/L2.hs, que é o arquivo principal para implementações da linguagem L2. A implementação da árvore sintática para programas L2 está presente no arquivo `Syntax.hs` na pasta L2.Frontend.

Como será feita a entrega

- As entregas serão feitas utilizando a plataforma Github classroom.