Exercício prático 06

BCC328 - Construção de Compiladores I

Prof. Rodrigo Ribeiro

12-07-2025

Finalização do compilador e interpretador para linguagem L2

O objetivo desta atividade é finalizar a implementação de um interpretador e compilador para a linguagem L2.

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 $\varphi(v) = \bot$ denota que não há valor associado a variável v no ambiente φ.

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

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

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

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

Análise semântica de L2

A etapa de análise semântica de L2 é bem simples: Basta verificar cada atribuição e garantir que toda variável do lado esquerdo não seja uma variável imutável. Sua implementação deverá utilizar um contexto, que consiste de uma lista de variáveis imutáveis visíveis em um determinado ponto do programa. A cada nova definição de variável imutável, você deverá incluí-la no contexto e assim que o escopo de sua definição terminar, esta deve ser removida deste contexto.

A inserção de variáveis imutáveis no contexto deve ser feita utilizando a função `insertVar` e a remoção por `removeVar`. Ambas as funções já estão implementadas no módulo `L.L2.Frontend.TypeCheck`.

Detalhes da entrega

O que deverá ser implementado

Você deverá implementar:

- Verificação semântica para L2.
- Geração de código para a máquina virtual V1.
- Geração de código C a partir do código L2 e uso do GCC para geração do executável.

A seguir, detalharemos a estrutura pré-definida do projeto para L2. Primeiramente, você deverá modificar a implementação da função

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

para que esta realize as análises léxica, sintática e semântica antes da execução do código fornecido no arquivo de entrada.

A geração de código para a máquina virtual V1 deve ser implementada no módulo `L.L2.Backend.V1Codegen`. No arquivo `L2.hs`, você deverá implementar a função:

```
v1Compiler :: FilePath -> IO ()
v1Compiler file = error "Not implemented!"
```

que realizará toda a etapa de análise do código e então gerará um arquivo contendo as instruções V1 correspondentes ao programa L2 fornecido como entrada. O texto gerado por seu compilador deve utilizar a função `pretty` definida no módulo `Utils.Pretty` para produzir código sintaticamente correto.

Geração de código executável é feita usando código fonte C que deve ser compilado usando o compilador GCC.

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

Como será feita a entrega

- As entregas serão feitas utilizando a plataforma Github classroom.