Relatório de Projeto: Compilador SL

Rafael Louback Ferraz
Matrícula: 11.1.4089
BCC328 - Construção de Compiladores I - DECOM/UFOP

21 de fevereiro de 2026

Resumo

Sumário

1 Introdução

Neste trabalho, foi desenvolvido um compilador em Haskell para a linguagem de programação SL, uma linguagem simples com suporte a funções, registros e arranjos. Este relatório abordará a segunda parte do trabalho que foca na analise semântica e interpretador de um compilador.

2 Metodologia

A metodologia consiste em pesquisa no material da disciplina e exemplos encontrados na internet. Além de aprofundamento de conhecimento em Haskell ou nas bibliotecas utilizadas na implementação deste trabalho pelo uso de consultas a inteligencias artificiais como por exemplo Google Gemini, sem utilizar recursos de geração de código pronto.

2.1 Estrutura sintática de SL

A linguagem SL possui tipagem estática, com suporte aos tipos de dados int, void, float, string e bool. As estruturas de controle incluem if-else, while e for, enquanto as estruturas de dados suportam arranjos unidimensionais e registros (structs). A linguagem permite a definição de funções com parâmetros e retorno. Adicionalmente, a linguagem deve prover suporte a polimorfismo paramétrico (generics) e inferência de tipos.

3 Arquitetura do Compilador

A arquitetura desenvolvida consiste na implementação do compilador com os Frontends para análise léxica, sintática e semântica e uma porção de logica que gerencie o prompt de comando além da entrada e saída de dados.

3.1 Análise semântica

Depois que a arvore de sintaxe abstrata é gerada o próximo passo para a eventual execução do código fonte da linguagem SL é a analise semântica da árvore. Esta etapa consiste em verificar se os tipos definidos são adequados, se as variáveis acessadas estão declaradas em seus devidos escopos, verificação de fluxo de controle (e.g., if, for e while) e checagem de unicidade. E com isso verificar se o código descrito faz sentido lógico e obedece às regras de significado da linguagem.

Na implementação da análise semântica foram definidos a variavel Env que consiste em um mapa com uma string identificadora e um descritor de tipo para cada variável, estrutura ou função. Além de um conjunto de erros que podem ser lançados durante a verificação da analise semântica. Por ultimo foi utilizado o tipo Checker como uma Monada ReaderT que pode lançar erros caso os encontre. A execução da analise semântica é feita processando os elementos da arvore de sintaxe abstrata de modo recursivo, lançando erros se necessário ou não retornando algo e continuando o percorrimento da arvore até sua completude.

```haskell
type Env = Map Id Type

data SemanticError
  = UnboundVariable Id String  -- Variable name and context
  | UnboundFunction Id String  -- Function name and context
  | TypeMismatch Type Type String  -- Expected, Actual and context
  deriving (Show)

type Checker a = ReaderT Env (Except SemanticError) a
```

Me baseie neste prompt feito ao Google Gemini para entender como a análise semântica poderia ser implementada em Haskell: `https://gemini.google.com/share/5b3ecd8c8c0f`

4 Resultados e Discussão

Neste segundo trabalho temos como resultado a implementação da analise semântica, que utiliza da Monada ReaderT para sua execução.

4.1 Instruções de Uso

Para utilização do projeto deve-se configurar um ambiente Docker com Ubuntu Linux e inicializar o container Docker com os seguintes comandos:

```
docker-compose up -d
docker-compose exec sl bash
```

Após entrar no ambiente de execução do container deve-se navegar à pasta onde o projeto cabal é organizado e executar o seguinte comando:

```
cabal run SL -- -f 'Exemplo 1.sl' --Interp
```

Este comando executara a analise semântica, gerando erros no prompt de comando caso problemas sejam detectados no código fonte da linguagem SL.

4.2 Testes Realizados

Os testes consistiram em executar os exemplos de código fonte para a linguagem SL descritos no enunciado do trabalho, com exceção dos exemplos de função identidade e função map.

4.3 Limitações

Não foi implementado suporte a funções identidade e map. E neste segundo trabalho foi implementado a analise semântica mas não foi implementado o interpretador.

5 Conclusão

Neste trabalho foi implementado a analise semântica para a linguagem SL com foco na aprendizagem dos conceitos envolvidos em sua operação.

6 Referências

Referências

[1] Hutton, G. (2016). *Programming in Haskell*. Cambridge University Press.

[2] Appel, A. W. (1998). *Modern Compiler Implementation in ML*. Cambridge University Press.

[3] WebAssembly Community Group. (2023). *WebAssembly Specification*.