Construção de um Compilador para a Linguagem MiniCalc: Da Análise Semântica à Geração de Código

Introdução A construção de compiladores é uma área fundamental da Ciência da Computação, responsável pela tradução de linguagens de alto nível para instruções executáveis pela máquina. Estudantes frequentemente compreendem a análise sintática (front-end), mas encontram dificuldades na visualização das fases subsequentes de síntese. A complexidade aumenta na transição da árvore sintática para a geração de código efetivo. Este projeto aplica os conceitos fundamentais de construção de compiladores para implementar um ciclo completo de tradução, focando especificamente na análise semântica e nas etapas de back-end (geração de código intermediário, otimização e execução).

Objetivo O objetivo principal é desenvolver um protótipo funcional de um compilador para a linguagem "MiniCalc", uma linguagem voltada para operações aritméticas e manipulação de variáveis inteiras. A solução visa demonstrar praticamente o funcionamento da Tabela de Símbolos para validação semântica, a conversão da árvore sintática para uma Representação Intermediária (RI) baseada em Código de Três Endereços, e a aplicação de técnicas de otimização antes da execução final em uma Máquina Virtual.

Metodologia A solução foi implementada utilizando a linguagem Python, escolhida por sua clareza na manipulação de estruturas de dados. A metodologia seguiu estritamente o pipeline de compilação descrito na literatura acadêmica:

1. **Análise Semântica:** Foi implementada uma Tabela de Símbolos utilizando tabelas hash (dicionários) para garantir a declaração prévia de variáveis e a consistência de tipos (checagem estática), impedindo o uso de identificadores não declarados.
2. **Geração de Código Intermediário:** Ao invés de gerar código de máquina diretamente, o compilador traduz a Árvore Sintática Abstrata (AST) para Código de Três Endereços (TAC), uma representação linear e portátil.
3. **Otimização:** Foi desenvolvido um passo de otimização focado na técnica de "Dobra de Constantes" (Constant Folding), que pré-calcula expressões aritméticas constantes durante a compilação, reduzindo o custo computacional em tempo de execução.
4. **Execução:** A fase final utiliza um interpretador (Máquina Virtual) para executar as instruções da RI.

Resultado O resultado é um compilador funcional que processa arquivos fonte da linguagem MiniCalc. O sistema demonstra com êxito a fase de síntese: o código fonte é lido, validado semanticamente e transformado em uma lista de instruções otimizadas. Testes realizados mostraram que expressões complexas contendo apenas literais (ex: `2 + 3 * 4`) são reduzidas a um único valor na etapa de otimização, eliminando instruções redundantes antes da execução. O sistema valida corretamente o escopo de variáveis,

emitindo erros para variáveis não declaradas, e executa o programa final exibindo os resultados no console.

Conclusão O sistema solidifica o entendimento prático das fases de síntese de um compilador. A implementação evidenciou como a modularização entre front-end e back-end, intermediada por uma RI eficiente, facilita a otimização e a manutenção do código. O projeto cumpre os requisitos técnicos ao conectar a teoria de verificação de tipos e grafos de fluxo de controle à prática de engenharia de software.