UNIVERSIDADE FEDERAL DA FRONTEIRA SUL

CAMPUS CHAPECÓ

CURSO DE CIÊNCIA DA COMPUTAÇÃO

COMPONENTE CURRICULAR DE CONSTRUÇÃO DE COMPILADORES

Prof. Dr. BRAULIO ADRIANO DE MELLO

RIAN BORGES BARBOSA E JONATHAN GOTZ CORREA

ANALISADOR SINTÁTICO

CHAPECÓ

2025

Sumário

1 Resumo

O objetivo deste trabalho foi construir um analisador sintático para uma linguagem de programação simples, utilizando uma tabela de parsing LR e uma gramática livre de contexto previamente definida. O analisador sintático foi integrado ao analisador léxico desenvolvido no trabalho anterior, permitindo a leitura dos tokens reconhecidos e a verificação da estrutura sintática do código-fonte.

A implementação utiliza uma tabela de parsing gerada a partir da gramática da linguagem, realizando operações de shift, reduce, goto e aceitação conforme o processamento dos tokens. O sistema também realiza o tratamento de erros sintáticos, indicando o tipo de erro e a linha de ocorrência, além de registrar atributos dos símbolos reconhecidos e construir uma tabela de símbolos com informações relevantes para etapas posteriores da compilação.

O analisador fornece, ao final do processamento, um relatório detalhado das operações realizadas, incluindo o estado das pilhas, a fita de entrada e as ações executadas, facilitando a compreensão do funcionamento do parser e a identificação de possíveis falhas na análise sintática.

2 Referencial Teórico

A análise sintática é a etapa subsequente à análise léxica no processo de compilação. Seu objetivo é verificar se a sequência de tokens reconhecida pelo analisador léxico está de acordo com as regras gramaticais da linguagem, geralmente definidas por uma Gramática Livre de Contexto (GLC). A análise sintática constrói uma estrutura hierárquica (geralmente uma árvore sintática) que representa a organização dos comandos e expressões do programa.

Entre as técnicas de análise sintática, destacam-se os métodos baseados em tabelas de parsing, como os algoritmos LR, SLR e LALR, que permitem a implementação eficiente de parsers para gramáticas complexas. Esses métodos utilizam uma tabela de parsing, construída a partir da GLC da linguagem, para determinar as ações a serem tomadas (shift, reduce, goto ou accept) a cada passo do processamento dos tokens.

A GLC utilizada neste trabalho define as regras de formação dos comandos válidos da linguagem, incluindo declarações, atribuições, comandos condicionais e blocos. Um exemplo de GLC traduzida para a linguagem do trabalho é apresentado abaixo:

$$
\begin{aligned}
\text{PROGRAM}' &\rightarrow \text{SEQUENCE} \\
\text{SEQUENCE} &\rightarrow \text{ESTATEMENT SEQUENCE} \\
\text{SEQUENCE} &\rightarrow \text{ESTATEMENT} \\
\text{SEQUENCE} &\rightarrow \text{let IDENT = STATEMENT ;} \\
\text{ESTATEMENT} &\rightarrow \text{true} \\
\text{ESTATEMENT} &\rightarrow \text{false} \\
\text{ESTATEMENT} &\rightarrow \text{IDENT} \\
\text{CORPO\_IF} &\rightarrow \text{if ESTATEMENT \{ ESTATEMENT \}}
\end{aligned}
$$

A análise sintática, portanto, garante que o código-fonte não apenas contenha tokens válidos, mas que estes estejam organizados de acordo com a estrutura sintática da linguagem, possibilitando a correta interpretação e tradução do programa nas etapas seguintes do compilador.

3 Implementação e resultado

A implementação do analisador sintático foi realizada com base na GLC especificada no arquivo `glc.txt`. A tabela de parsing LR foi gerada com o auxílio do site SLR Parser Generator fornecido em aula. O analisador sintático utiliza operações de shift, reduce, goto e accept para validar a estrutura do código-fonte, processando a sequência de tokens fornecida pelo analisador léxico.

3.1 Tokens

Os tokens definidos para a linguagem incluem identificadores, palavras-chave (*let*, *if*, *true*, *false*), operadores (=, ;) e delimitadores (,). O analisador léxico, desenvolvido

no trabalho anterior, foi adaptado para reconhecer esses tokens e fornecer a sequência de entrada para o analisador sintático.

3.2 Resultados

O analisador sintático processa a sequência de tokens gerada pelo analisador léxico, realizando operações de shift, reduce e goto conforme a tabela de parsing. Ao final do processamento, o sistema apresenta um relatório detalhado das operações realizadas, incluindo o estado das pilhas, a fita de entrada e as ações executadas. Além disso, são reportados eventuais erros sintáticos, com indicação do tipo de erro e da linha de ocorrência, e é construída uma tabela de símbolos contendo os identificadores reconhecidos e seus respectivos atributos.

A implementação demonstrou ser capaz de reconhecer corretamente programas válidos segundo a gramática definida, bem como identificar e reportar erros sintáticos em casos de desvios das regras gramaticais.

4 Conclusão

Este trabalho apresentou a implementação de um analisador sintático baseado em tabela de parsing LR, integrado a um analisador léxico desenvolvido previamente. O sistema é capaz de identificar e classificar tokens, validar a estrutura sintática do código-fonte, construir uma tabela de símbolos e reportar erros léxicos e sintáticos com indicação de linha.

Entre as principais dificuldades enfrentadas, destacam-se o ajuste da gramática para garantir a correta geração da tabela de parsing e o tratamento de casos de erro durante a análise sintática. A experiência proporcionou uma compreensão aprofundada sobre o funcionamento de parsers automáticos e a importância da integração entre as etapas do processo de compilação.