Compiladores – Análise Léxica
Carlos Henrique Vieira Marques Veeck

Objetivo da Análise Léxica

- Antes da tradução, o compilador precisa entender a estrutura e o significado do programa.
- A análise léxica transforma o programa-fonte em uma sequência de tokens.
  Exemplo:
  `programa-fonte → analisador léxico → tokens`

Elementos Básicos

- **Token**: nome do token + atributos opcionais. Ex: `NUMBER(42)`
- **Padrão**: descrição dos possíveis lexemas para um token. Ex: `[0-9]+`
- **Lexema**: sequência de caracteres que casa com um padrão. Ex: `"42"`
- **Tabela de símbolos**: armazena informações associadas aos tokens (como identificadores e valores).

Construção do Analisador Léxico

- Definir a microsintaxe da linguagem: tokens e seus padrões.
- Definir critérios de agregação/separação de palavras.
- Estabelecer palavras reservadas (ex: `if`, `while`, `return`).
- Implementar ou gerar automaticamente o analisador (ex: com ferramentas como Lex ou Flex).

Erros Léxicos

- Nem sempre o erro pode ser detectado só pelo lexer.
- Exemplo: `fi (a == f(x))` → lexer pode interpretar `fi` como identificador, mas o parser detectará erro.

Lexer vs Parser

- **Separar análise léxica da sintática traz vantagens:**
    - **Simplicidade:** deixa o parser mais limpo
    - **Eficiência:** otimizações separadas
    - **Portabilidade:** adaptações ficam concentradas no analisador léxico

---

Expressões Regulares e Tokens

- Tokens são definidos por expressões regulares.
- Operações regulares principais:
    - Concatenação
    - União (`|`)
    - Fecho de Kleene (`*`)
- São importantes para identificar padrões de texto (ex: número, palavra, símbolo).

Reconhecimento de Tokens com Autômatos

- Expressões regulares são convertidas em:
    - Diagramas de transição
    - Máquinas de estados finitos (autômatos)
- Tipos de autômatos:

- **AFN (Não Determinístico)**: múltiplas transições possíveis
- **AFD (Determinístico)**: apenas uma transição por símbolo