

Lógica da Computação

Aula 03

Raul Ikeda

1º semestre de 2020

Esta Aula

- Formas de representação de Linguagens:
 - Diagrama Sintático
- Melhorias no Compilador:
 - Tokenização
 - Embrião do Sintático

Aula Passada: Gramáticas

- Gramática atual:

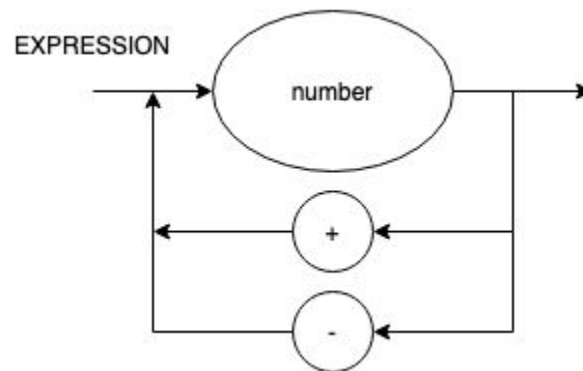
$$G = (\{E, T, +, -, n\}, \{+, -, n\}, P, E)$$

$$P = \begin{cases} E \rightarrow nT \\ T \rightarrow +nT \\ T \rightarrow -nT \\ T \rightarrow \lambda \end{cases}$$

- Vamos representá-la de outra forma que fique mais simples de implementar

Diagrama Sintático

- É uma outra forma de representar uma linguagem. Vamos tomar o exemplo da gramática do compilador:

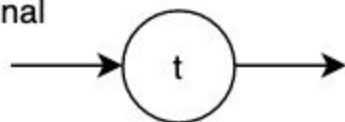


- Olhando o fluxo, pode-se dizer facilmente que as seguintes cadeias são aceitáveis:
 - número
 - número + número
 - número - número
 - número + número - número - número
 - etc

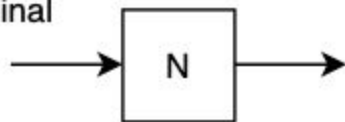
Como será que isso se conecta a teoria vista na aula passada?

Diagrama Sintático

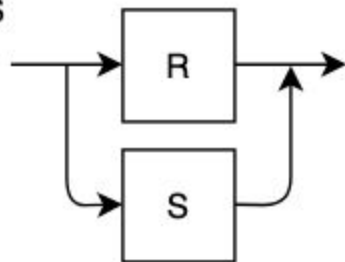
Terminal



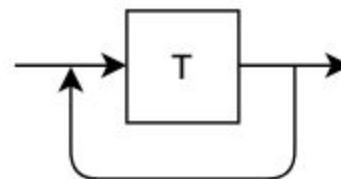
Não Terminal



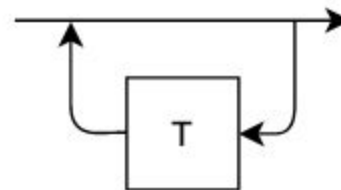
R ou S



T^+



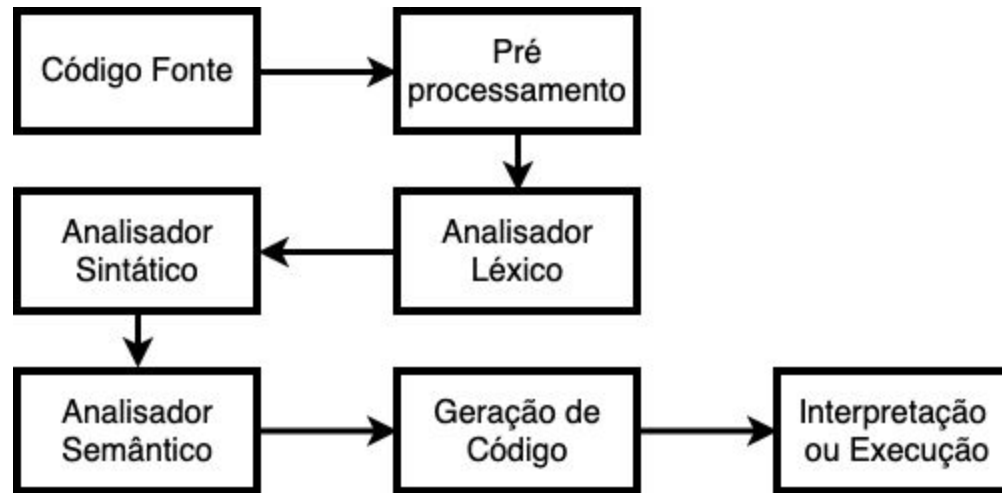
T^*



λ



De volta ao Compilador



- Já temos uma ideia de como estruturar o compilador para reconhecer uma cadeia de palavras (Análise sintática).
- Mas como alimentamos o analisador com os cadeias (Análise Léxica)?

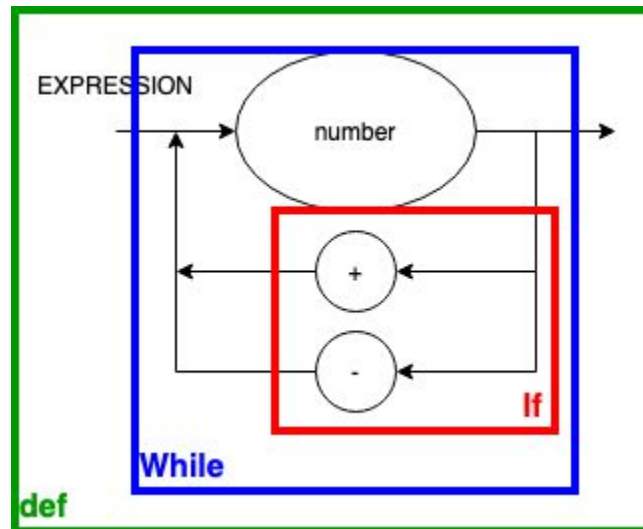
Tokenizador

- Relembrando: Parte do compilador responsável por capturar um token (átomo) do texto-fonte.
- Um token é normalmente composto pelo seu valor e seu tipo.
- Normalmente ignora espaços e linefeed.
- Funciona como uma máquina de estados.
- Por enquanto reconhecemos apenas 4 tipos de Tokens:
 - INT
 - PLUS
 - MINUS
 - EOF (end of file - quando acaba a cadeia)

Exemplo: “1+22” = [(1, INT), ('+', PLUS), (22, INT), ("", EOF)]

DS para o código

- É possível "implementar" facilmente um Diagrama Sintático.



- Consegue imaginar como ficaria um **símbolo não terminal**?

DS para o código

- Código

Função Expressão:

Se o token atual for número:

 Copiar número para o resultado

 Pegar próximo token

Enquanto token for + ou -:

 Se o token atual é +:

 Pegar próximo token

 Se o token atual for número:

 Somar o número no resultado

 Senão ERRO

 Se o token atual é -:

 Pegar próximo token

 Se o token atual for número:

 Subtrair o número no resultado

 Senão ERRO

 Pegar próximo token

Retornar resultado

Senão ERRO

Atividade: Roteiro 1

- Roteiro Impresso ou PDF no Blackboard

Próxima Aula

- Gramáticas Regulares
- Autômatos Finitos

Referências:

- Marcus et al Cap. 3.1 e Cap. 3.3