Insper

Lógica da Computação

Aula 03

Raul Ikeda

2º semestre de 2018

Aula Passada

1. Compiladores
2. Gramáticas e Linguagens

Esta Aula

1. Formas de representação de Linguagens:
    1. Diagrama Sintático
2. Melhorias no Compilador:
    1. Tokenização
    2. Embrião do Sintático

Aula Passada: Gramáticas

- Como ficou a gramática do nosso compilador até o momento?

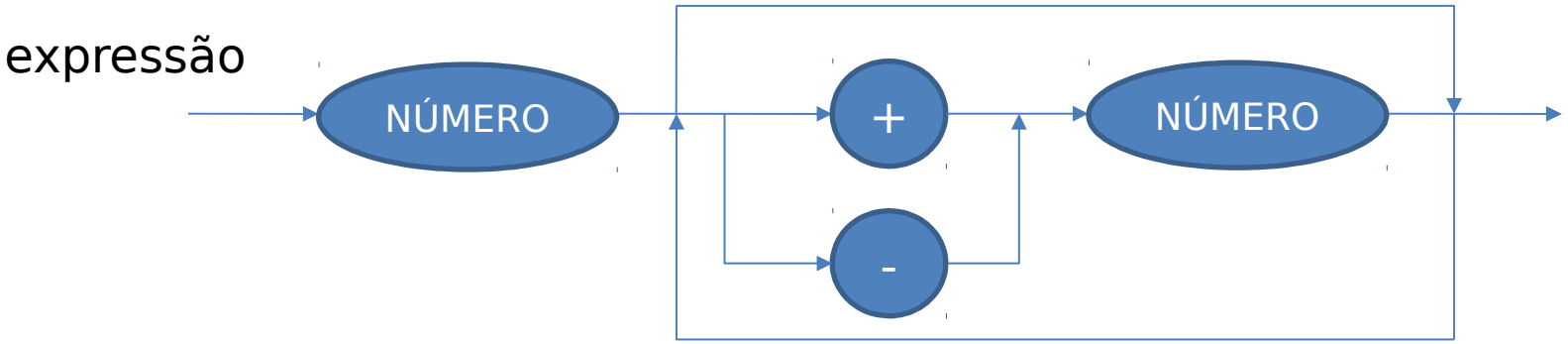
$$G = (\{E, T, +, -, n\}, \{+, -, n\}, P, E)$$

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

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

Diagrama Sintático

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

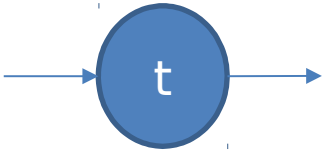


- Olhando o fluxo, pode-se dizer facilmente que as seguintes cadeias são aceitáveis:
    1. número
    2. número + número
    3. número - número
    4. número + número - número - número
    5. etc

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

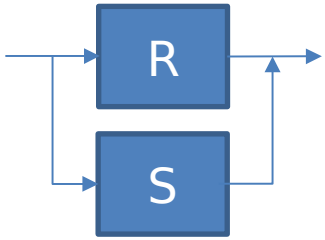
Diagrama Sintático

Terminal

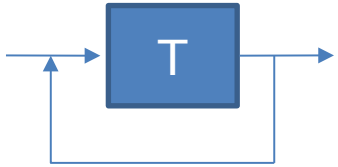


Não Terminal

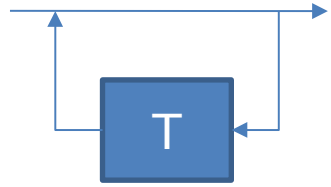
R ou S



T^+



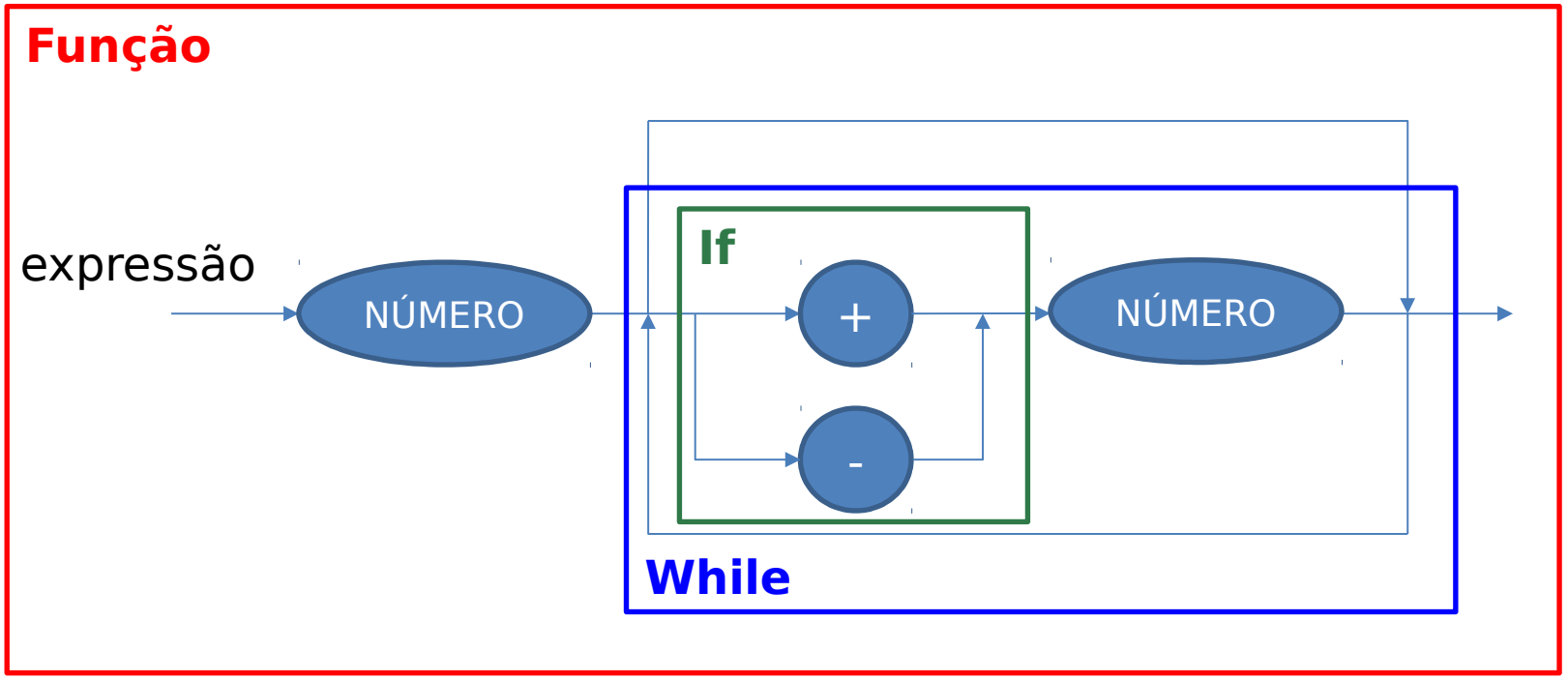
T^*



λ

DS para Código

- É possível “implementar” facilmente um Diagrama Sintático.



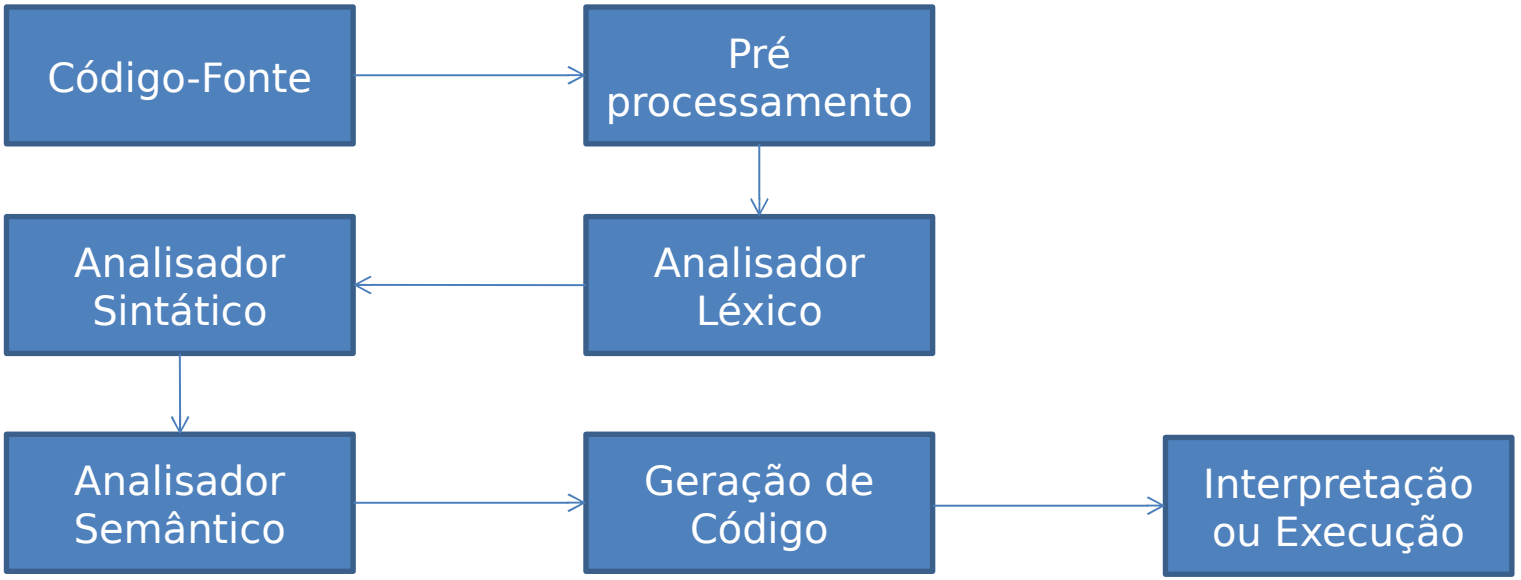
- Consegue imaginar como ficaria um **símbolo não terminal**?

DS para Código

- Algoritmo:

```
Função expressão
  Pegar próximo token
  Se token atual é um número
    Copiar número para resultado
    Pegar próximo token
    Enquanto existir token
      Se token atual é +
        Pegar próximo token
        Se token atual é um número: somar número ao resultado
        Senão erro
      Senão se token atual é –
        Pegar próximo token
        Se token atual é um número: subtrair número do resultado
        Senão erro
      Senão erro
      Pegar próximo token
    Fim do Enquanto
  Senão erro
  Retornar resultado
Fim da função
```

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 comentários, espaços e linefeed.
- Funciona como uma máquina de estados.

- Por enquanto reconhecemos apenas 3 tipos de Tokens:
    - INT
    - PLUS
    - MINUS

- Exemplo: “1+22” = [(1, INT), (“+”, PLUS), (22, INT)]

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