Insper

Lógica da Computação

Aula 09

Raul Ikeda

2º semestre de 2018

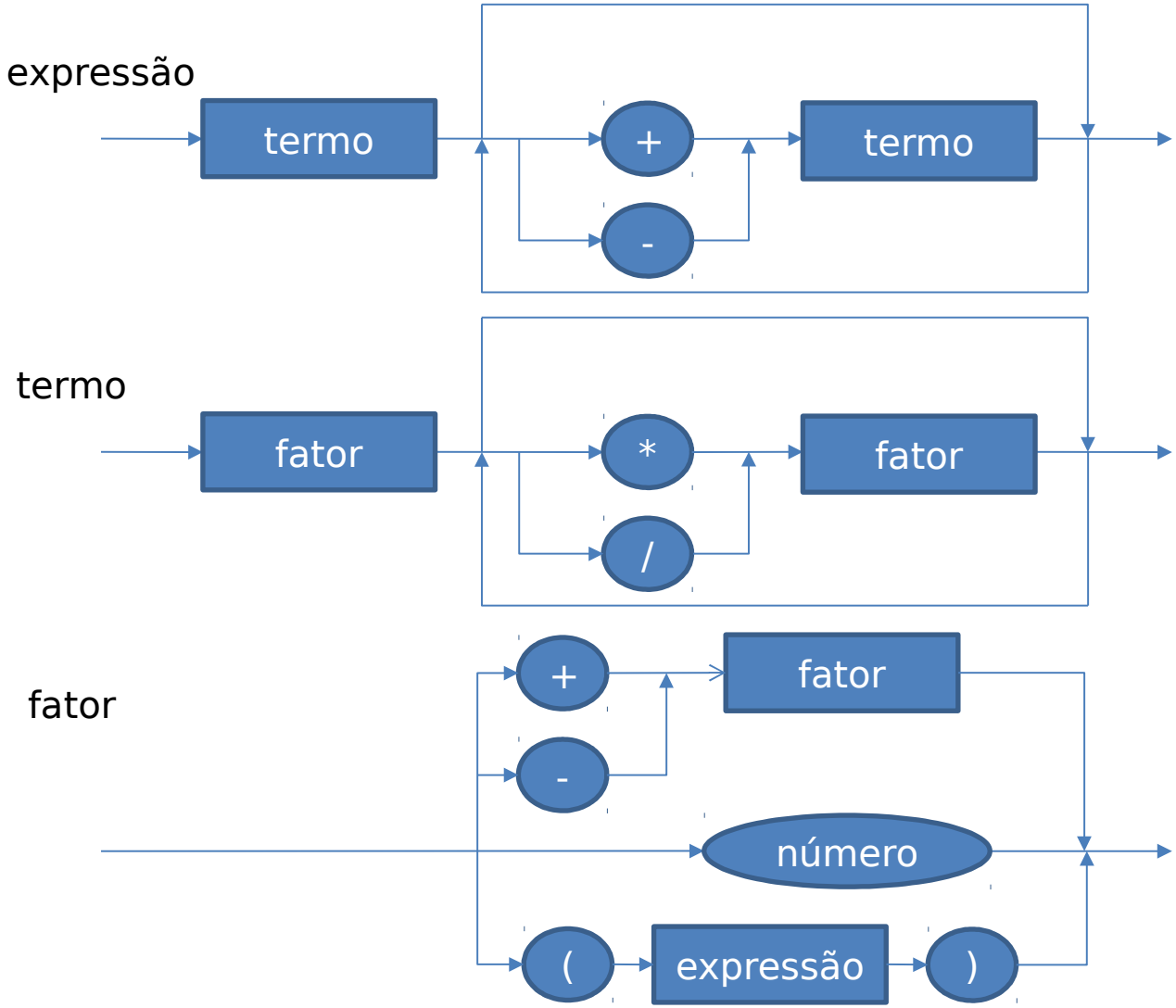
Aula Passada

1. Gramática Livres de Contexto
2. Autômatos de Pilha

Esta Aula

1. Abstract Syntax Tree

Situação Atual



EBNF e Gramática

```
expressão = termo, { (“+” | “-”), termo } ;
termo = fator, { (“*” | “/”), fator } ;
fator = (“+” | “-”) fator | número | “(”, expressão, “)” ;
número = “-2^63” | ... | “2^63” ;
```

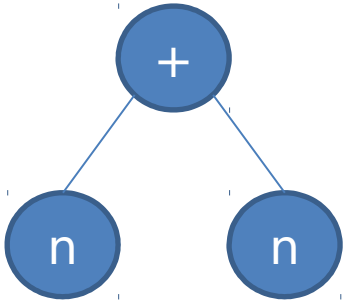
$$G = (\{E, T, F, +, -, *, /, (,), n\}, \{+, -, *, /, (,), n\}, P, E)$$

$$P = \begin{cases} E \to T \\ E \to E + T \\ E \to E - T \\ T \to F \\ T \to T * F \\ T \to T / F \\ F \to -F \\ F \to +F \\ F \to (E) \\ F \to n \end{cases}$$

AST - Abstract Syntax Tree

- Assim como árvore de derivação é uma forma de representar uma cadeia em função de uma gramática. A AST representa a ordem com que o diagrama sintático é percorrido.
- Ideia: **alguns** símbolos terminais da nossa gramática vão virar **nós** de uma **árvore**. Dependendo do símbolo, este poderá possuir 0 ou mais filhos.
- Exemplos:

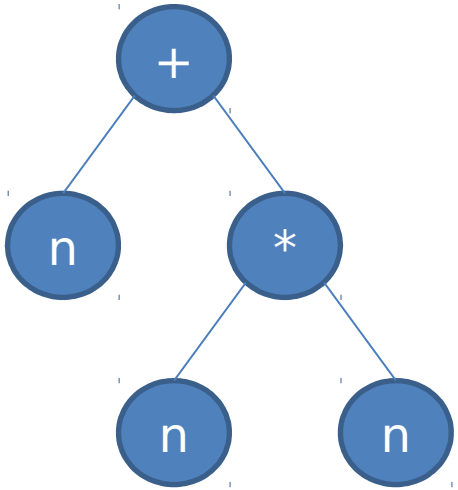
n+n



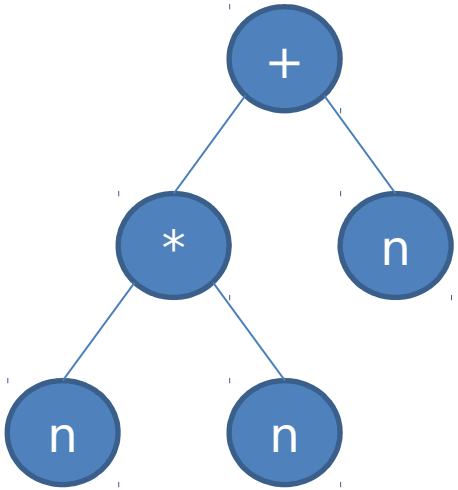
--n

AST – A ordem importa

n + n * n

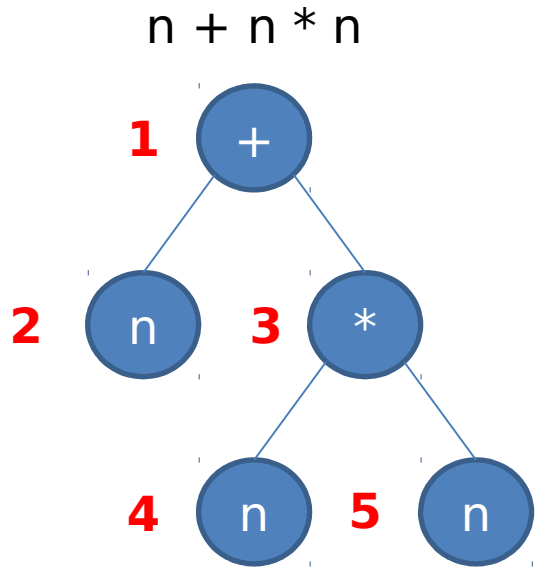


n * n + n



Para montar corretamente, é preciso realizar o ***tracing*** do diagrama sintático.

AST – Para que serve?

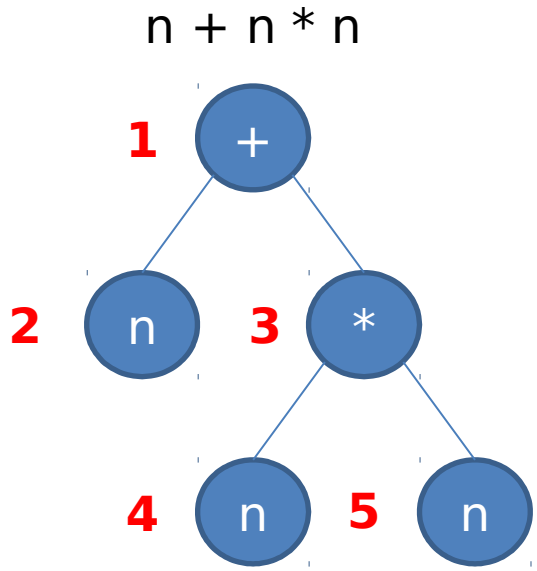


Percorrendo a árvore em profundidade da esquerda para a direita:

+ n (* n n) **Notação Polonesa ou Prefix**

Pergunta do milhão: aonde isso será útil mesmo?

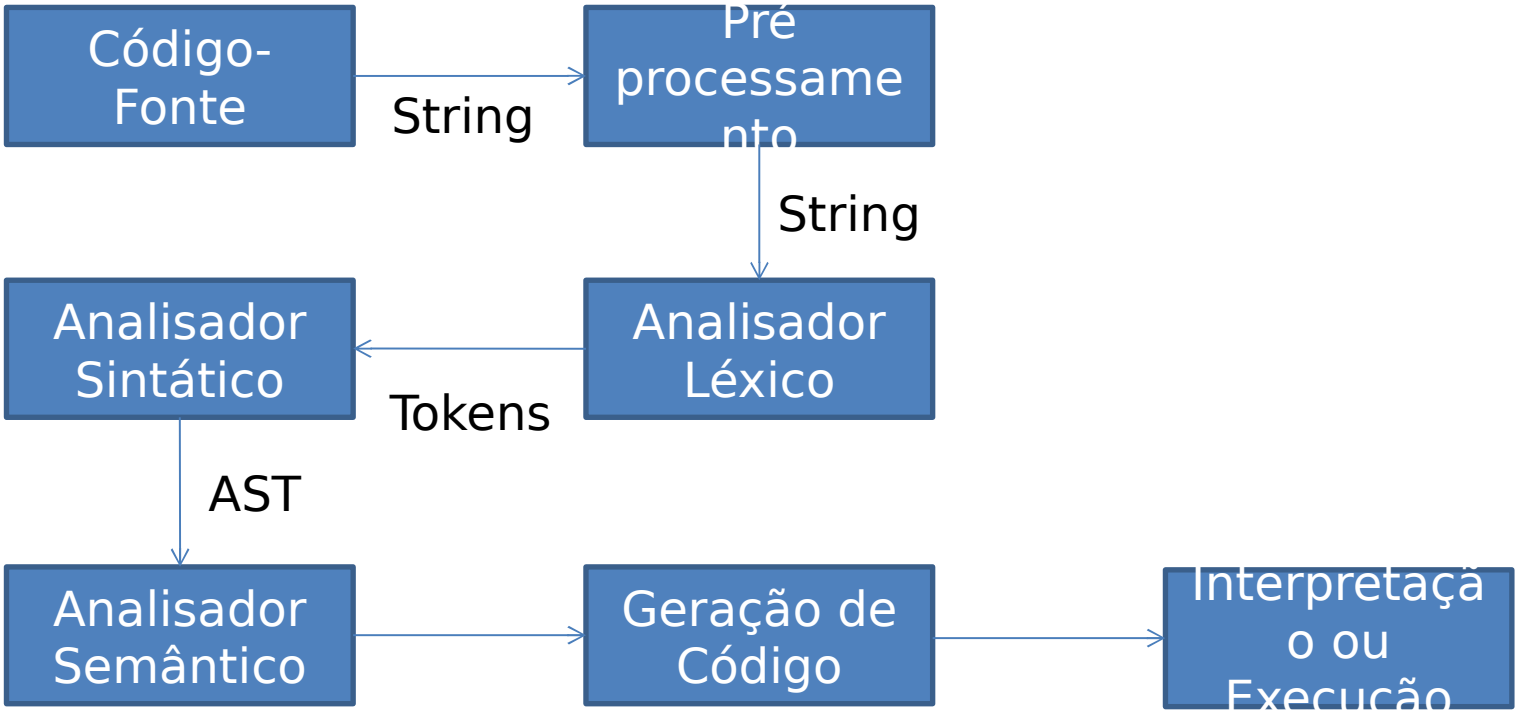
AST – Como funciona?



Cada nó pede para que cada filho interprete a sua parte recursivamente:

1. #1: Resolve aí #2
2. #2: Toma n
3. #1: Resolve aí #3
4. #3: Resolve aí #4
5. #4: Toma n
6. #3: Resolve aí #5
7. #4: Toma n
8. #3: Toma n * n
9. #1: Toma n + n * n

Como estamos?



Atividade: Roteiro 4

- Roteiro Impresso ou PDF no Blackboard.

Próxima Aula

- Pumping Lemma para CFL
- Propriedades de fecho
- Questões agora não tão decidíveis

- Referências:
    - Hopcroft et al. Cap. 7
    - Sipser Cap 2.3