Construção de compiladores

Profs. Mário César San Felice (e Helena Caseli,
Murilo Naldi, Daniel Lucrédio)
Departamento de Computação - UFSCar
1º semestre / 2018
Tópico 5 - Análise Sintática Ascendente

Análise sintática ascendente

Ou análise sintática bottom-up

Introdução

- Vimos que existem duas formas de se reconhecer uma linguagem através de uma gramática
  - Inferência recursiva
  - Derivação

- Ex: Gramática para expressões aritméticas
  - V = {E,I}
  - T = {+,*,(,),a,b,0,1}
  - P = conjunto de regras ao lado
  - S = E

```
E → I
  | E + E
  | E * E
  | (E)

I → a
  | b
  | Ia
  | Ib
  | I0
  | I1
```

Introdução

- Inferência recursiva
  - Dada uma cadeia (conjunto de símbolos terminais)
  - Do corpo para a cabeça
  - Ex: a*(a+b00)
    - a*(a+b00) $\Leftarrow$ a*(a+I00) $\Leftarrow$ a*(a+I0) $\Leftarrow$ a*(a+I) $\Leftarrow$ a*(a+E) $\Leftarrow$ a*(I+E) $\Leftarrow$ a*(E+E) $\Leftarrow$ a*(E) $\Leftarrow$ a*E $\Leftarrow$ I*E $\Leftarrow$ E*E $\Leftarrow$ E

Introdução

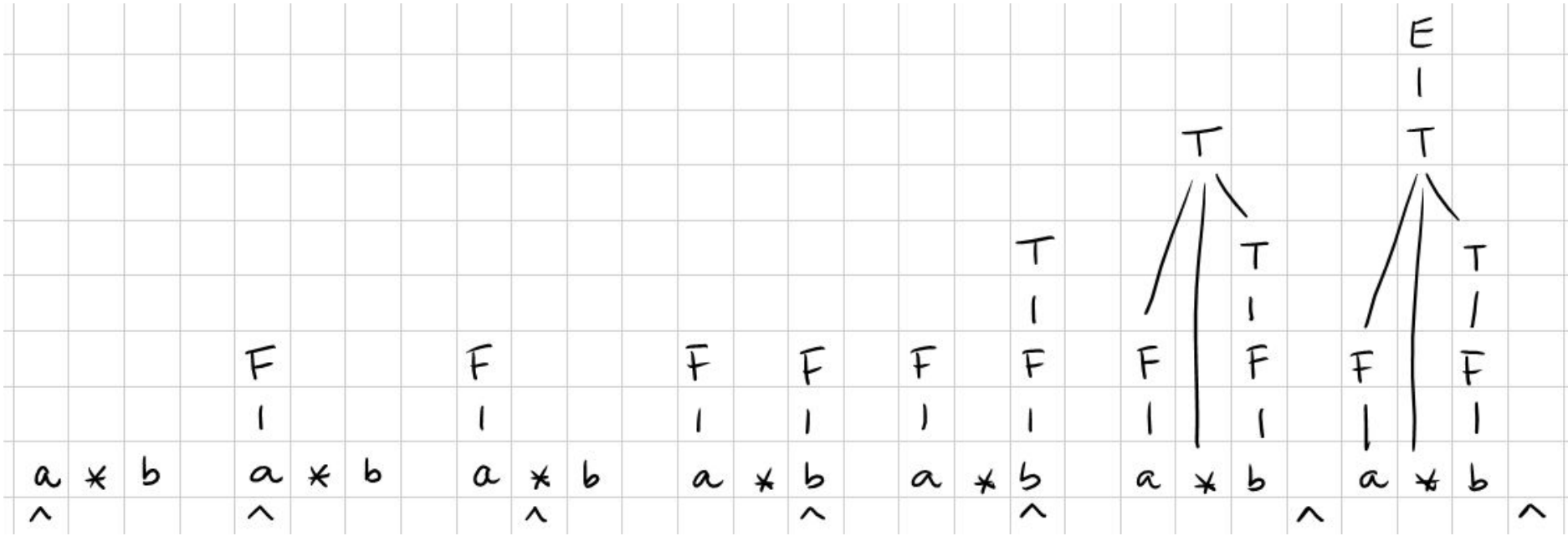
- Análise sintática ascendente
  - Faz o processo de inferência
    - Ou
  - Cria a árvore de análise sintática “de baixo para cima”
    - Análise bottom-up

Introdução

- Exemplo

```
E → T + E | T
T → F * T | F
F → a | b | (E)
```

Cadeia = `a * b`



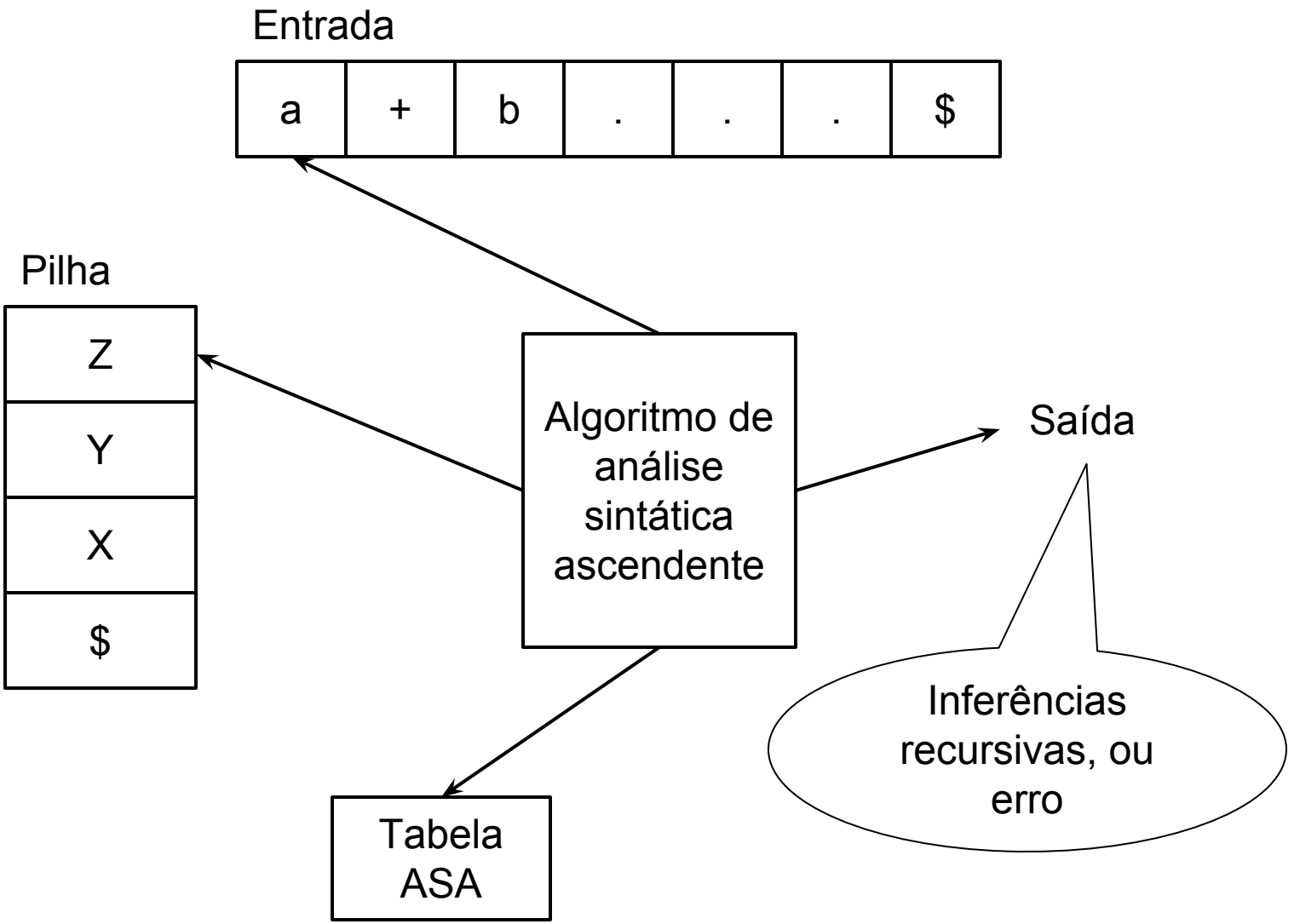
Introdução

- Em LFA
  - Analisador sintático = autômato de pilha

- Na análise sintática descendente
  - A pilha armazena os símbolos a serem substituídos
  - Quando a pilha esvaziar, acabou

- Na análise sintática ascendente
  - A pilha vai armazenar os símbolos aguardando “redução”
  - Quando sobrar só o símbolo inicial na pilha, acabou

Introdução

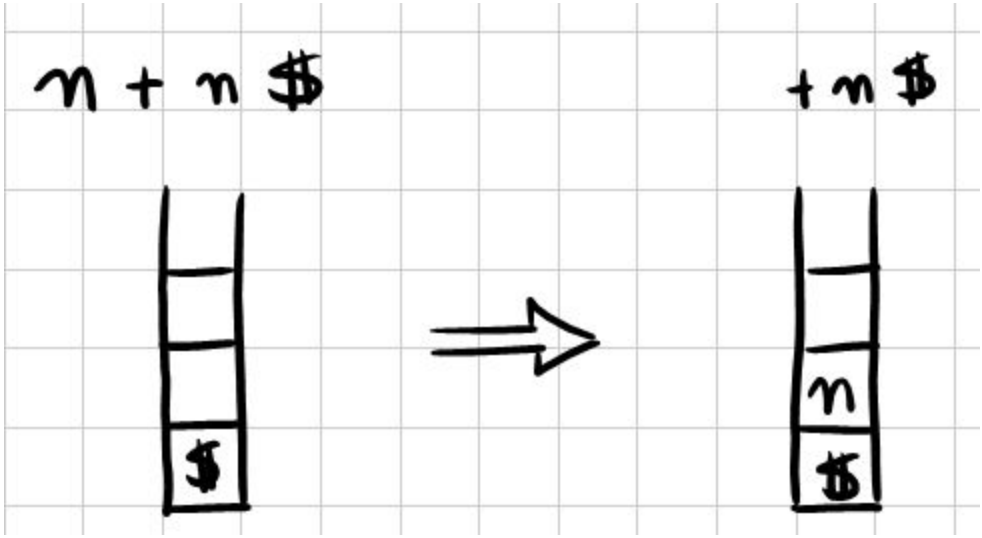

Entrada
a
+
b
.
.
.
$
Pilha
Z
Y
X
$
Algoritmo de análise sintática ascendente
Saída
Inferências recursivas, ou erro
Tabela ASA

Introdução

- Na análise sintática ascendente, para reconhecer uma cadeia de entrada:
  - **Empilha**
    - Os símbolos da cadeia de entrada
  - **Reduz**
    - O lado direito de uma produção no topo da pilha, substituindo-o pelo lado esquerdo da produção
- Os passos 1 e 2 são repetidos até que
  - ACEITA – símbolos da cadeia de entrada foram consumidos e pilha possui apenas o símbolo inicial
    - OU
  - ERRO – o processo foi interrompido antes de chegar ao final

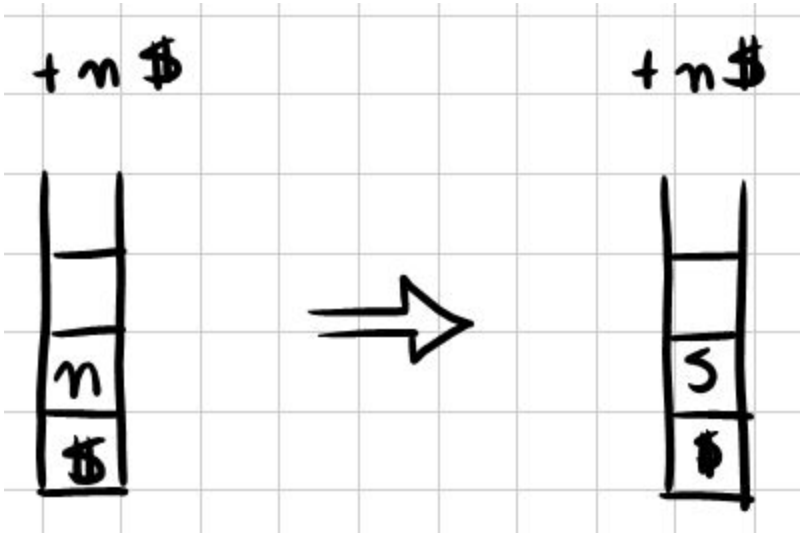
Análise sintática ascendente

- Empilhamento
  - Consiste em remover um símbolo da entrada e adicioná-lo ao topo da pilha
- Ex:
  - Gramática = S → S + n | n
  - Entrada = n + n



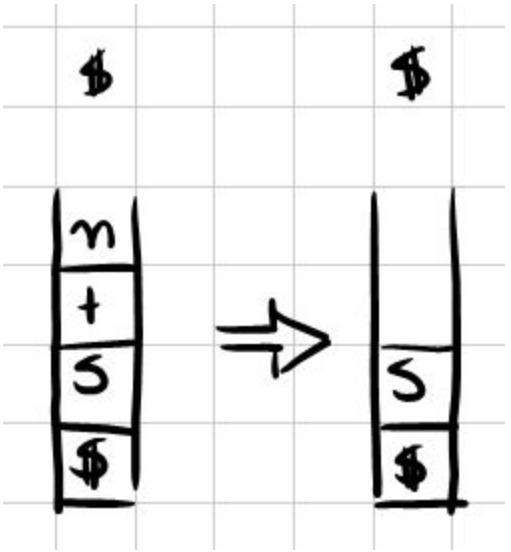
Análise sintática ascendente

- Redução
  - Consiste em substituir símbolos no topo da pilha por um único símbolo
    - Não consome a entrada
- Ex:
  - Gramática = S → S + n | n
  - Entrada = n + n



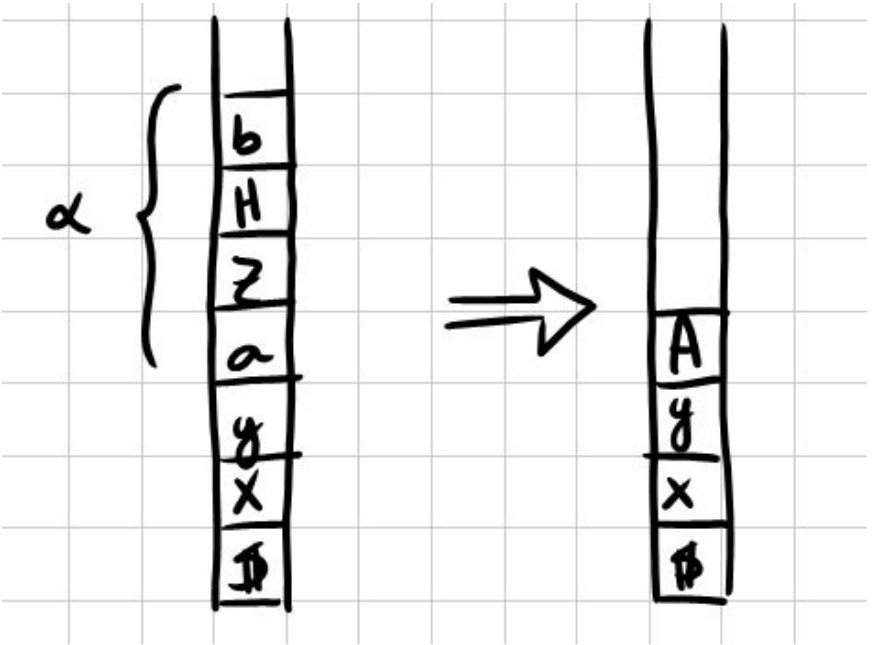
Análise sintática ascendente

- Redução
  - Consiste em substituir símbolos no topo da pilha por um único símbolo
    - Não consome a entrada
- Ex:
  - Gramática = S → S + n | n
  - Entrada = n + n



Análise sintática ascendente

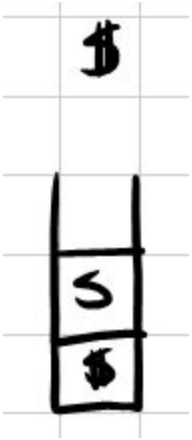
- Quando empilhar/reduzir?
- Conceito de “gancho” (handle)
  - Para cada produção A → α
  - α é um “gancho”
    - Quando α aparecer no topo da pilha, posso substituir por A
  - Ex: A → aZHb
    - α = aZHb



Análise sintática ascendente

- Aceita
  - Quando consumir toda a entrada
  - Pilha contém somente o símbolo inicial

- Ex:
  - Gramática = S → S + n | n
  - Entrada = n + n



Análise sintática ascendente

- Exemplo
  - Gramática: S → S + n | n
  - Entrada: n + n

Agora escrevemos a pilha da esquerda para a direita, para facilitar

Apareceu um gancho aqui

E aqui também

Pilha	Entrada	Ação
$	**n**+n$	empilha n
$**n**	**+**n$	reduz S→n
$S	**+**n$	empilha +
$S+	**n**$	empilha n
$**S+n**	**$**	reduz S→S+n
$S	**$**	aceita

Análise sintática ascendente

- Desafio
  - Detectar o aparecimento do “gancho” na pilha
  - Exige olhar um ou mais símbolos da pilha
  - E também olhar símbolos à frente na entrada
    - Normalmente, busca-se olhar somente um símbolo à frente, por uma questão de eficiência

Análise sintática ascendente

- Analisadores sintáticos ascendentes (ASA)
  - 2 tipos
    - Diferença está na forma com que detectam o aparecimento do “gancho”

- Analisador de precedência de operadores
  - Opera sobre a classe das **gramáticas de operadores**
  - Guiado por uma **tabela de precedência**
  - **Não veremos nesta disciplina!**

Análise sintática ascendente

- Analisador LR (k)
  - **L**eft to right
    - Lê a sentença em análise da esquerda para a direita
  - **R**ightmost derivation
    - Produz uma derivação mais à direita
      - Inferência recursiva
  - Considerando-se k símbolos na cadeia de entrada

Fim