

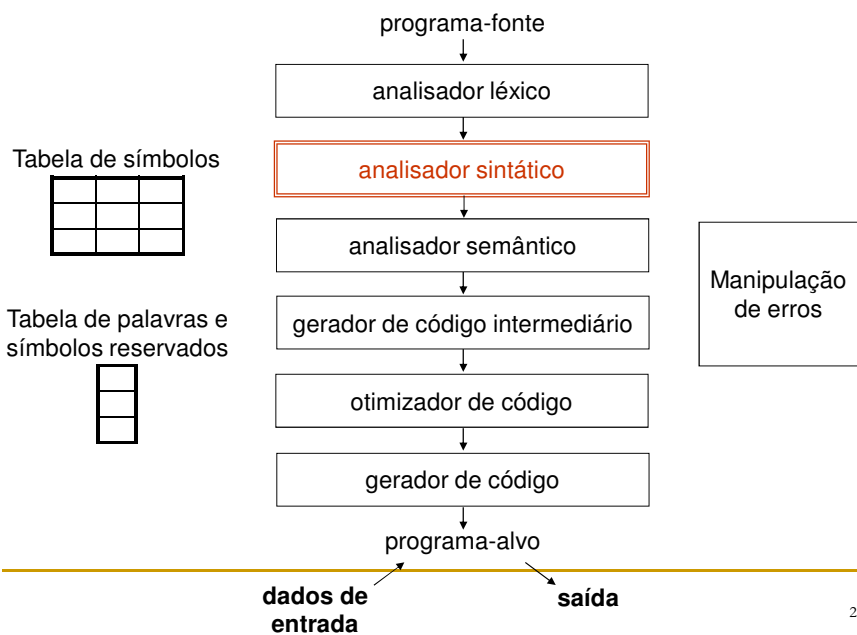
## Análise sintática

Função, interação com o compilador  
 Análise descendente e ascendente  
 Especificação e reconhecimento de cadeias de tokens válidas  
 Implementação  
 Tratamento de erros

Prof. Thiago A. S. Pardo

1

## Estrutura geral de um compilador



2

## Analizador sintático

- Analizador sintático ou *parser*: processo principal do compilador
  - Coordena as outras etapas
- Funções
  - Verificar a boa formação do programa: quais cadeias pertencem à linguagem
    - Sintaxe, gramática
    - Construção da árvore sintática do programa: implícita ou explícita
  - Tratar erros
- Exemplos
  - while ( <exp> ) <comandos>
  - id := <exp>

3

## Análise sintática

### ■ 2 principais tipos

- Top-down ou descendente

- Da raiz para as folhas

<programa>



program p ...

- Bottom-up ou ascendente

- Das folhas para a raiz

<programa>

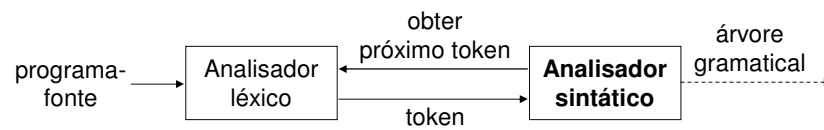


program p ...

4

## Análise sintática

- Para análise eficiente, trabalha-se com uma **subclasse de gramáticas**
  - Suficientemente expressivas para descrever a maioria das linguagens de programação



5

## Análise sintática descendente (ASD)

- Parte-se do símbolo inicial da gramática e tenta-se chegar às folhas
- **2 tipos**
  - ASD com retrocesso
  - ASD preditiva
    - Recursiva
    - Não recursiva

6

## ASD com retrocesso

- Método de tentativa e erro
- Um dos primeiros métodos que surgiram
- Fácil de implementar manualmente
- Características
  - Exploratório: tenta todas as possibilidades
  - Ineficiente
- Funcionamento
  - A cada passo, escolhe uma regra e aplica
  - Se falhar em algum ponto, retrocede e escolhe uma outra regra
  - O processo termina quando a cadeia é reconhecida ou quando as regras se esgotaram e a cadeia não foi reconhecida

7

## ASD com retrocesso

### ■ Exemplo

$$\langle E \rangle ::= \langle T \rangle + \langle E \rangle \mid \langle T \rangle$$

$$\langle T \rangle ::= \langle F \rangle^* \langle T \rangle \mid \langle F \rangle$$

$$\langle F \rangle ::= a \mid b \mid (\langle E \rangle)$$

Reconhecer a cadeia  $a^*b$

8

### ASD com retrocesso

$$\begin{aligned} \langle E \rangle &::= \langle T \rangle + \langle E \rangle \mid \langle T \rangle \\ \langle T \rangle &::= \langle F \rangle * \langle T \rangle \mid \langle F \rangle \\ \langle F \rangle &::= a \mid b \mid (\langle E \rangle) \end{aligned}$$

(1) a\*b

E

↑

(2) a\*b

E

↙ ↘

T+E

↑

(3) a\*b

E

↙ ↘

T+E

↙ ↘

F\*T

↑

(4) a\*b

E

↙ ↘

T+E

↙ ↘

F\*T

↑

a

(5) \*b

E

↙ ↘

T+E

↙ ↘

F\*T

↑

a

(6) b

E

↙ ↘

T+E

↙ ↘

F\*T

↑

a

(7) b

E

↙ ↘

T+E

↙ ↘

F\*T

↙ ↘

a F\*T

↑

(8) b

E

↙ ↘

T+E

↙ ↘

F\*T

↙ ↘

a F\*T

↑

a

7 ←

---

9

### ASD com retrocesso

$$\begin{aligned} \langle E \rangle &::= \langle T \rangle + \langle E \rangle \mid \langle T \rangle \\ \langle T \rangle &::= \langle F \rangle * \langle T \rangle \mid \langle F \rangle \\ \langle F \rangle &::= a \mid b \mid (\langle E \rangle) \end{aligned}$$

(9) b

E

↙ ↘

T+E

↙ ↘

F\*T

↙ ↘

a F\*T

↑

b

(10) λ

E

↙ ↘

T+E

↙ ↘

F\*T

↙ ↘

a F\*T

↑

b

(11) b

E

↙ ↘

T+E

↙ ↘

F\*T

↙ ↘

a F\*T

↑

( E )

6 ← 7 ←

(12) b

E

↙ ↘

T+E

↙ ↘

F\*T

↙ ↘

a F

↑

(13) b

E

↙ ↘

T+E

↙ ↘

F\*T

↙ ↘

a F

↑

a

12 ←

(14) b

E

↙ ↘

T+E

↙ ↘

F\*T

↙ ↘

a F

↑

b

---

10

### ASD com retrocesso

$$\begin{aligned} \langle E \rangle &::= \langle T \rangle + \langle E \rangle \mid \langle T \rangle \\ \langle T \rangle &::= \langle F \rangle^* \langle T \rangle \mid \langle F \rangle \\ \langle F \rangle &::= a \mid b \mid (\langle E \rangle) \end{aligned}$$

(15)

$$\begin{array}{c} \lambda \\ \swarrow \downarrow \searrow \\ E \\ \swarrow \downarrow \searrow \\ T + E \\ \swarrow \downarrow \searrow \\ F^* T \\ \swarrow \downarrow \searrow \\ a \quad F \\ \quad \quad \downarrow \\ \quad \quad b \end{array}$$

12 ←

(16)

$$\begin{array}{c} b \\ \swarrow \downarrow \searrow \\ E \\ \swarrow \downarrow \searrow \\ T + E \\ \swarrow \downarrow \searrow \\ F^* T \\ \swarrow \downarrow \searrow \\ a \quad F \\ \quad \quad \downarrow \\ \quad \quad (\langle E \rangle) \\ \quad \quad \uparrow \end{array}$$

3 ← 6 ← 12 ←

(17)

$$\begin{array}{c} a^*b \\ \swarrow \downarrow \searrow \\ E \\ \swarrow \downarrow \searrow \\ T + E \\ \swarrow \downarrow \searrow \\ F^* T \\ \swarrow \downarrow \searrow \\ b \end{array}$$

3 ←

(18)

$$\begin{array}{c} a^*b \\ \swarrow \downarrow \searrow \\ E \\ \swarrow \downarrow \searrow \\ T + E \\ \swarrow \downarrow \searrow \\ F^* T \\ \swarrow \downarrow \searrow \\ (\langle E \rangle) \\ \uparrow \end{array}$$

2 ← 3 ←

(19)

$$\begin{array}{c} ^*b \\ \swarrow \downarrow \searrow \\ E \\ \swarrow \downarrow \searrow \\ T + E \\ \downarrow \\ F \\ \uparrow \end{array}$$

---

11

### ASD com retrocesso

$$\begin{aligned} \langle E \rangle &::= \langle T \rangle + \langle E \rangle \mid \langle T \rangle \\ \langle T \rangle &::= \langle F \rangle^* \langle T \rangle \mid \langle F \rangle \\ \langle F \rangle &::= a \mid b \mid (\langle E \rangle) \end{aligned}$$

(20)

$$\begin{array}{c} a^*b \\ \swarrow \downarrow \searrow \\ E \\ \swarrow \downarrow \searrow \\ T + E \\ \downarrow \\ F \\ \downarrow \\ a \\ \uparrow \end{array}$$

(21)

$$\begin{array}{c} a^*b \\ \swarrow \downarrow \searrow \\ E \\ \swarrow \downarrow \searrow \\ T + E \\ \downarrow \uparrow \\ F \\ \downarrow \\ a \end{array}$$

19 ←

(22)

$$\begin{array}{c} a^*b \\ \swarrow \downarrow \searrow \\ E \\ \swarrow \downarrow \searrow \\ T + E \\ \downarrow \\ F \\ \downarrow \\ b \\ \uparrow \end{array}$$

19 ←

(23)

$$\begin{array}{c} a^*b \\ \swarrow \downarrow \searrow \\ E \\ \swarrow \downarrow \searrow \\ T + E \\ \downarrow \\ F \\ \swarrow \downarrow \searrow \\ (\langle E \rangle) \\ \uparrow \end{array}$$

1 ← 2 ← 19 ←

(24)

$$\begin{array}{c} a^*b \\ \downarrow \\ E \\ \downarrow \\ T \\ \uparrow \end{array}$$

(25)

$$\begin{array}{c} a^*b \\ \downarrow \\ E \\ \downarrow \\ T \\ \swarrow \downarrow \searrow \\ F^* T \\ \uparrow \end{array}$$

(26)

$$\begin{array}{c} a^*b \\ \downarrow \\ E \\ \downarrow \\ T \\ \swarrow \downarrow \searrow \\ F^* T \\ \downarrow \\ a \\ \uparrow \end{array}$$

---

12

### ASD com retrocesso

$$\begin{aligned} \langle E \rangle &::= \langle T \rangle + \langle E \rangle \mid \langle T \rangle \\ \langle T \rangle &::= \langle F \rangle^* \langle T \rangle \mid \langle F \rangle \\ \langle F \rangle &::= a \mid b \mid (\langle E \rangle) \end{aligned}$$

(27)

\*b

E

|

T

/ \

F\* T

| |

a ↑

(28)

b

E

|

T

/ \

F\* T

| |

a ↑

(29)

b

E

|

T

/ \

F\* T

| / \

a F\* T

| |

↑

(30)

b

E

|

T

/ \

F\* T

| / \

a F\* T

| |

a ↑

(31)

b

E

|

T

/ \

F\* T

| / \

a F\* T

| |

b ↑

(32)

λ

E

|

T

/ \

F\* T

| / \

a F\* T

| |

b ↑

29 ←

29 ←

---

13

### ASD com retrocesso

$$\begin{aligned} \langle E \rangle &::= \langle T \rangle + \langle E \rangle \mid \langle T \rangle \\ \langle T \rangle &::= \langle F \rangle^* \langle T \rangle \mid \langle F \rangle \\ \langle F \rangle &::= a \mid b \mid (\langle E \rangle) \end{aligned}$$

(33)

b

E

|

T

/ \

F\* T

| / \

a F\* T

| |

( E )

↑

(34)

b

E

|

T

/ \

F\* T

| |

a F

↑

(35)

b

E

|

T

/ \

F\* T

| |

a F

↑

(36)

b

E

|

T

/ \

F\* T

| |

a F

↑

(37)

λ

E

|

T

/ \

F\* T

| |

a F

↑

28 ← 29 ←

---

14

## ASD com retrocesso

- O número de derivações pode ser uma função **exponencial** do tamanho da cadeia
- A **recursividade à esquerda não é permitida** nos métodos de ASD
  - O que acontece com a gramática abaixo?

$$\begin{aligned} \langle E \rangle &::= \langle E \rangle + \langle T \rangle \mid \langle T \rangle \\ \langle T \rangle &::= \langle T \rangle * \langle F \rangle \mid \langle F \rangle \\ \langle F \rangle &::= a \mid b \mid (\langle E \rangle) \end{aligned}$$

15

## ASD com retrocesso

- Se gramática recursiva à esquerda, deve-se eliminar essa recursividade
- Exemplo

$$\begin{array}{l} \langle E \rangle ::= \langle E \rangle \text{ op } \langle T \rangle \mid \langle T \rangle \\ \langle T \rangle ::= a \end{array} \quad \longrightarrow \quad \begin{array}{l} \langle E \rangle ::= \langle T \rangle \text{ op } \langle E \rangle \mid \langle T \rangle \\ \langle T \rangle ::= a \end{array}$$

Recursão à esquerda eliminada?  
Produzem a mesma cadeia?

16

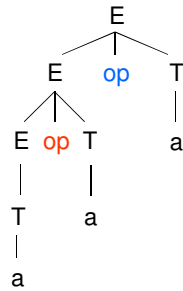


## ASD com retrocesso

$$\langle E \rangle ::= \langle E \rangle \text{ op } \langle T \rangle \mid \langle T \rangle \longrightarrow$$

$$\langle T \rangle ::= a$$

a op a op a

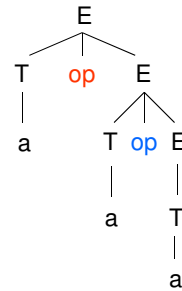


Precedência à esquerda!

$$\langle E \rangle ::= \langle T \rangle \text{ op } \langle E \rangle \mid \langle T \rangle$$

$$\langle T \rangle ::= a$$

a op a op a



Precedência à direita!

17

## ASD

- A análise só é eficiente quando se **eliminam retrocessos**
  - Sabe-se de antemão qual regra aplicar
    - Como garantir isso?

18

## ASD

- A análise só é eficiente quando se **eliminam retrocessos**
    - Sabe-se de antemão qual regra aplicar
  - Além de não serem recursivas à esquerda, as gramáticas devem obedecer **duas restrições**
    - Os lados direitos das produções devem começar por terminais
    - Para um não terminal qualquer, não devem existir duas regras que comecem com um mesmo terminal
- Com isso, olhando o primeiro símbolo da entrada, sabe-se qual regra aplicar

19

## ASD

### ■ Exemplo

$$\langle E \rangle ::= a \mid b \mid * \langle E \rangle \langle E \rangle \mid + \langle E \rangle \langle E \rangle$$

Reconhecer a cadeia +a\*ba

20

**ASD**  $\langle E \rangle ::= a \mid b \mid * \langle E \rangle \langle E \rangle \mid + \langle E \rangle \langle E \rangle$

(1)

+ a \* b a

E

↑

(2)

+ a \* b a

E

+ E E

↑

(3)

a \* b a

E

+ E E

↑

(4)

a \* b a

E

+ E E

  |

  a

  ↑

(5)

\* b a

E

+ E E

  |

  a

  ↑

(6)

\* b a

E

+ E E

  |

  a

  \* E E

  ↑

---

21

**ASD**  $\langle E \rangle ::= a \mid b \mid * \langle E \rangle \langle E \rangle \mid + \langle E \rangle \langle E \rangle$

(7)

b a

E

+ E E

  |

  a

  \* E E

  ↑

(8)

b a

E

+ E E

  |

  a

  \* E E

  |

  b

  ↑

(9)

a

E

+ E E

  |

  a

  \* E E

  |

  b

  ↑

(10)

a

E

+ E E

  |

  a

  \* E E

  |

  b

  |

  a

  ↑

(11)

λ

E

+ E E

  |

  a

  \* E E

  |

  b

  |

  a

  ↑

---

22

## ASD

- Além de não serem recursivas à esquerda, as gramáticas devem obedecer duas restrições
  - O lado direito das produções devem começar por terminais
  - Para um não terminal qualquer, não devem existir duas regras que comecem com um mesmo terminal
  
- Restrição muito severa!
  - Dá para melhorar?

23

## ASD

- Além de não serem recursivas à esquerda, as gramáticas devem obedecer duas restrições
  - O lado direito das produções devem começar por terminais
  - Para um não terminal qualquer, não devem existir duas regras que comecem com um mesmo terminal
  
- Restrição muito severa!
  - Generalização: podem existir não terminais começando os lados direitos das regras de um não terminal, mas seus conjuntos Primeiro devem ser disjuntos

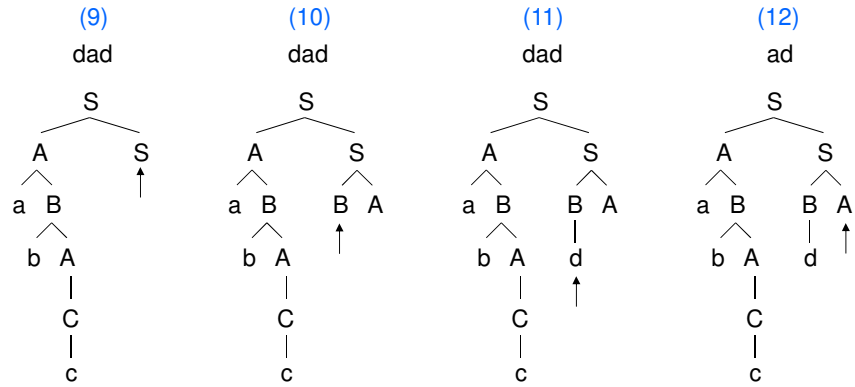
24



# ASD

$\langle S \rangle ::= \langle A \rangle \langle S \rangle \mid \langle B \rangle \langle A \rangle$   
 $\langle A \rangle ::= a \langle B \rangle \mid \langle C \rangle$   
 $\langle B \rangle ::= b \langle A \rangle \mid d$   
 $\langle C \rangle ::= c$

■ Reconhecer cadeia abcdad

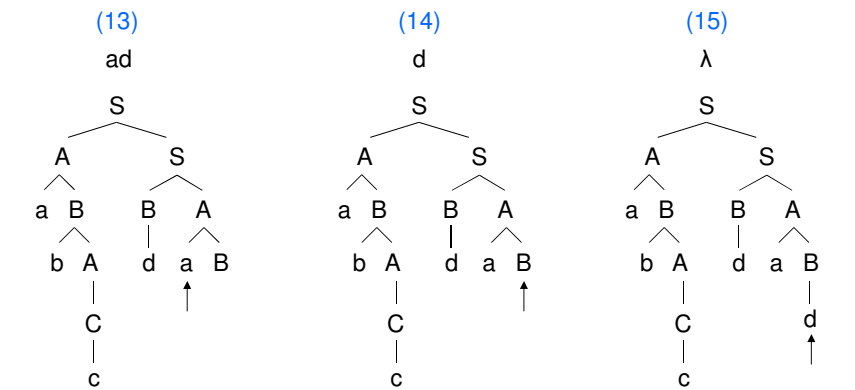


27

# ASD

$\langle S \rangle ::= \langle A \rangle \langle S \rangle \mid \langle B \rangle \langle A \rangle$   
 $\langle A \rangle ::= a \langle B \rangle \mid \langle C \rangle$   
 $\langle B \rangle ::= b \langle A \rangle \mid d$   
 $\langle C \rangle ::= c$

■ Reconhecer cadeia abcdad



28

## ASD

- As gramáticas que
  - Não são recursivas à esquerda
  - Para um não terminal, não possuem regras cujo lado direito comecem com o mesmo terminal

são chamadas gramáticas LL(1)

- *Left to right, Leftmost derivation*
- 1 único símbolo a frente para determinar qual regra aplicar

29

## ASD

- ASD preditiva
- Dois métodos
  - ASD preditiva recursiva
  - ASD preditiva não recursiva

30

## ASD preditiva recursiva

- Um **analisador sintático recursivo** é um conjunto de procedimentos possivelmente recursivos, um para cada não terminal a ser derivado
- Se se dispõe de uma gramática LL(1), pode-se usar tal método
  - Eficiência

31

## ASD preditiva recursiva

- **Exemplo** (note que não é LL(1))

$\langle E \rangle ::= \langle T \rangle + \langle E \rangle \mid \langle T \rangle$

$\langle T \rangle ::= \langle F \rangle^* \langle T \rangle \mid \langle F \rangle$

$\langle F \rangle ::= a \mid b \mid (\langle E \rangle)$

```
procedimento E
begin
  T;
  se (símbolo='+') então
    obter_simbolo;
  E;
end
```

```
procedimento T
begin
  F;
  se (símbolo='*') então
    obter_simbolo;
  T;
end
```

```
procedimento ASD
begin
  obter_simbolo;
  E;
  se fim-de-cadeia OK
  senão ERRO
end
```

```
procedimento F
begin
  se (símbolo='(') então
    obter_simbolo;
  E;
  se (símbolo=')') então
    obter_simbolo
  senão ERRO;
  senão se (símbolo='a')
    ou (símbolo='b')
    então obter_simbolo
  senão ERRO;
end
```

32



## ASD preditiva recursiva

- Exemplo (note que não é LL(1))

$\langle E \rangle ::= \langle T \rangle + \langle E \rangle \mid \langle T \rangle$

$\langle T \rangle ::= \langle F \rangle^* \langle T \rangle \mid \langle F \rangle$

$\langle F \rangle ::= a \mid b \mid (\langle E \rangle)$

**Testar com cadeia a+b**

```

procedimento E
begin
  T;
  se (símbolo='+') então
    obter_simbolo;
  E;
end

```

```

procedimento T
begin
  F;
  se (símbolo='*') então
    obter_simbolo;
  T;
end

```

```

procedimento ASD
begin
  obter_simbolo;
  E;
  se fim-de-cadeia OK
  então ERRO
end

```

```

procedimento F
begin
  se (símbolo='(') então
    obter_simbolo;
    E;
  se (símbolo=')') então
    obter_simbolo
  senão ERRO;
  senão se (símbolo='a')
    ou (símbolo='b')
  então obter_simbolo
  senão ERRO;
end

```

33

## ASD preditiva recursiva

- Método formal para gerar os procedimentos
  - Regras de transformação: mapeamento das regras de um não terminal em grafos sintáticos
  - Regras de tradução: mapeamento dos grafos em procedimentos
- Exemplo

$\langle S \rangle ::= a \langle A \rangle d$

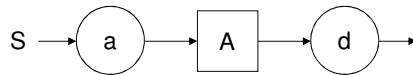
$\langle A \rangle ::= c \langle A \rangle \mid e \langle B \rangle$

$\langle B \rangle ::= f \mid g$

34

## ASD preditiva recursiva

- $\langle S \rangle ::= a \langle A \rangle d$



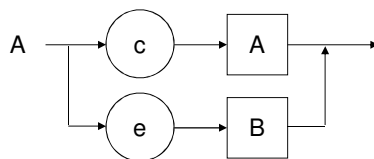
```

procedimento S
begin
  se (simbolo='a') então
    obter_simbolo;
    A;
  se (simbolo='d')
    então obter_simbolo
    senão ERRO;
  senão ERRO;
end
  
```

35

## ASD preditiva recursiva

- $\langle A \rangle ::= c \langle A \rangle \mid e \langle B \rangle$



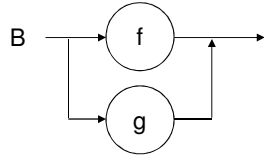
```

procedimento A
begin
  se (simbolo='c') então
    obter_simbolo;
    A;
  senão se (simbolo='e') então
    obter_simbolo
    B;
  senão ERRO;
end
  
```

36

## ASD preditiva recursiva

- $\langle B \rangle ::= f \mid g$



```

procedimento B
begin
  se (simbolo='f') ou (simbolo='g')
  então obter_simbolo
  senão ERRO;
end
  
```

37

## ASD preditiva recursiva

- Programa principal

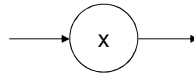
```

procedimento ASD
begin
  obter_simbolo;
  S;
  se (terminou_cadeia)
  então SUCESSO
  senão ERRO
end
  
```

38

## ASD preditiva recursiva

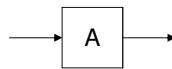
- Regras de transformação
    - Regras gramaticais → grafos sintáticos
1. Toda regra é mapeada em um grafo
  2. Toda ocorrência de um terminal x em uma forma corresponde ao seu reconhecimento na cadeia de entrada e à leitura do próximo símbolo dessa cadeia



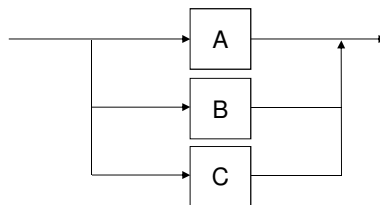
39

## ASD preditiva recursiva

3. Toda ocorrência de um não-terminal A corresponde a análise imediata de A



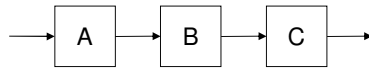
4. Alternativas são representadas como



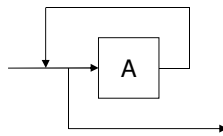
40

## ASD preditiva recursiva

5. Uma seqüência A B C é mapeada em



6. A forma  $\{A\}^*$  ou  $A^*$  é representada por



41

## ASD preditiva recursiva

### ■ Exercício

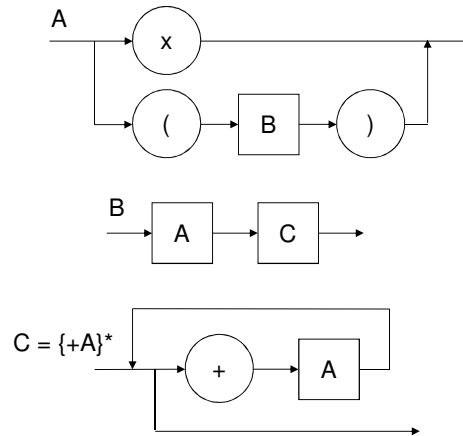
$\langle A \rangle ::= x \mid (\langle B \rangle)$   
 $\langle B \rangle ::= \langle A \rangle \langle C \rangle$   
 $\langle C \rangle ::= +\langle A \rangle \langle C \rangle \mid \lambda$

42

## ASD preditiva recursiva

### ■ Exercício

$\langle A \rangle ::= x \mid (\langle B \rangle)$   
 $\langle B \rangle ::= \langle A \rangle \langle C \rangle$   
 $\langle C \rangle ::= +\langle A \rangle \langle C \rangle \mid \lambda$



43

## ASD preditiva recursiva

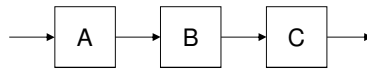
### ■ Regras de tradução

- Grafos sintáticos → procedimentos
1. Reduzir o número de grafos: união de grafos para maior simplicidade e eficiência
    - Bom senso!
  2. Escrever um procedimento para cada grafo

44

## ASD preditiva recursiva

### 3. A seqüência



origina o procedimento

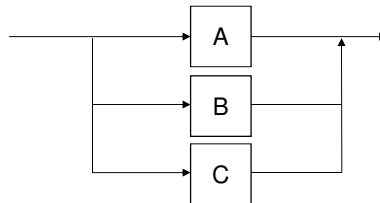
```

begin
    A;
    B;
    C;
end
  
```

45

## ASD preditiva recursiva

### 4. A alternativa



origina o procedimento

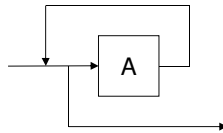
```

begin
    se (símbolo está em Primeiro(A)) então A
    senão se (símbolo está em Primeiro(B)) então B
    senão se (símbolo está em Primeiro(C)) então C
end
  
```

46

## ASD preditiva recursiva

### 5. Uma repetição



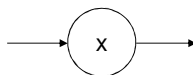
origina o procedimento

```
begin
    enquanto (símbolo está em Primeiro(A)) faça
        A;
end
```

47

## ASD preditiva recursiva

### 6. O terminal



origina

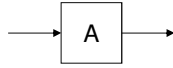
```
begin
    se (símbolo=x)
        então obter_simbolo
        senão ERRO;
end
```

48



## ASD preditiva recursiva

### 7. O não terminal



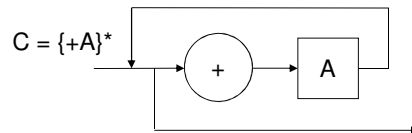
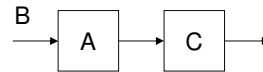
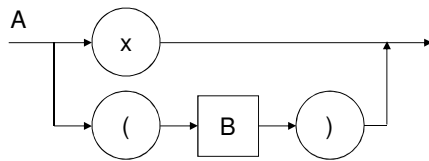
origina

```
begin
  A;
end
```

49

## ASD preditiva recursiva

- Exercício: fazer o(s) procedimento(s) para os grafos sintáticos



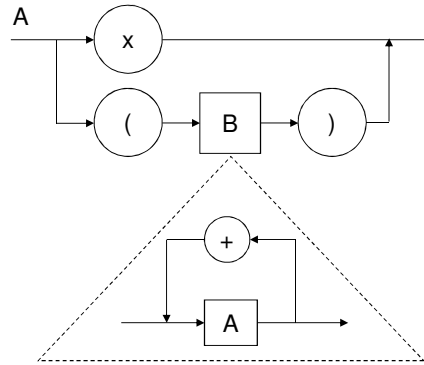
50

## ASD preditiva recursiva

```

procedimento A
begin
  se (símbolo='x') então obter_símbolo
  senão se (símbolo('(') então
    obter_símbolo;
    A;
  enquanto (símbolo='+') faça
    obter_símbolo;
    A;
  se (símbolo=')')
    então obter_símbolo
    senão ERRO;
senão ERRO;
end

```



51

## ASD preditiva

- Operações para re-escrita de gramáticas para processamento pela ASD preditiva
  - Eliminação da recursividade direta à esquerda por re-escrita

$$\langle A \rangle ::= \langle A \rangle \alpha \mid \beta \quad \rightarrow \quad \begin{array}{l} \langle A \rangle ::= \beta \langle A' \rangle \\ \langle A' \rangle ::= \alpha \langle A' \rangle \mid \lambda \end{array}$$

Exemplo:

$$\begin{array}{l} \langle E \rangle ::= \langle E \rangle + \langle T \rangle \mid \langle T \rangle \\ \langle T \rangle ::= \langle T \rangle * \langle F \rangle \mid \langle F \rangle \\ \langle F \rangle ::= (\langle E \rangle) \mid \text{id} \end{array} \quad \rightarrow \quad \begin{array}{l} \langle E \rangle ::= \langle T \rangle \langle E' \rangle \\ \langle E' \rangle ::= + \langle T \rangle \langle E' \rangle \mid \lambda \\ \langle T \rangle ::= \langle F \rangle \langle T' \rangle \\ \langle T' \rangle ::= * \langle F \rangle \langle T' \rangle \mid \lambda \\ \langle F \rangle ::= (\langle E \rangle) \mid \text{id} \end{array}$$

52

## ASD preditiva

- Eliminação da recursividade direta à esquerda por substituição

$$\langle E \rangle ::= \langle E \rangle + \langle T \rangle \mid \langle T \rangle \quad \rightarrow \quad \langle E \rangle ::= \langle T \rangle \{ + \langle T \rangle \}$$

53

## ASD preditiva

- Fatoração à esquerda: eliminar regras com mesmo terminal em seus conjuntos primeiro

$$\langle A \rangle ::= \alpha \beta_1 \mid \alpha \beta_2 \quad \rightarrow \quad \begin{aligned} \langle A \rangle &::= \alpha \langle A' \rangle \\ \langle A' \rangle &::= \beta_1 \mid \beta_2 \end{aligned}$$

Regra: para cada não terminal A, achar o maior prefixo  $\alpha$  em comum a duas ou mais alternativas; se há um prefixo em comum, então substituir a produção  $\langle A \rangle ::= \alpha \beta_1 \mid \dots \mid \alpha \beta_n \mid \gamma$  (em que  $\gamma$  representa as alternativas que não começam por  $\alpha$ ) por

$$\begin{aligned} \langle A \rangle &::= \alpha \langle A' \rangle \mid \gamma \\ \langle A' \rangle &::= \beta_1 \mid \dots \mid \beta_n \end{aligned}$$

54

## ASD preditiva

### Exemplo

$$\begin{aligned} \langle S \rangle &::= \text{if } \langle E \rangle \text{ then } \langle S \rangle \mid \text{if } \langle E \rangle \text{ then } \langle S \rangle \text{ else } \langle S \rangle \mid a \\ \langle E \rangle &::= b \end{aligned}$$


$$\begin{aligned} \langle S \rangle &::= \text{if } \langle E \rangle \text{ then } \langle S \rangle \langle S' \rangle \mid a \\ \langle S' \rangle &::= \text{else } \langle S \rangle \mid \lambda \\ \langle E \rangle &::= b \end{aligned}$$

55

## ASD preditiva

### Atenção: recursividade indireta

$$\begin{aligned} \langle S \rangle &::= \langle A \rangle a \mid b \\ \langle A \rangle &::= \langle A \rangle c \mid \langle S \rangle d \mid \lambda \end{aligned}$$

$$S \rightarrow Aa \rightarrow Sda$$

**RECURSÃO**

56

## ASD preditiva

- Identificação da linguagem reconhecida pela gramática para posterior reescrita da gramática

Gramática G  $\rightarrow$  L(G)  $\rightarrow$  Gramática G1

57

## ASD preditiva

### ■ Exercício

- A gramática seguinte é LL(1)? Se não é, transforme-a

```


<S> ::= i<A>
<A> ::= <E>
<E> ::= <T> + <E> | <T>
<T> ::= <F> * <T> | <F>
<F> ::= <P> <F> | <P>
<P> ::= i | (<E>)
  
```

58

## ASD preditiva

### ■ Exercício

- A gramática seguinte é LL(1)? Se não é, transforme-a

$\begin{aligned} \langle S \rangle &::= i \langle A \rangle \\ \langle A \rangle &::= \langle E \rangle \\ \langle E \rangle &::= \langle T \rangle + \langle E \rangle \mid \langle T \rangle \\ \langle T \rangle &::= \langle F \rangle * \langle T \rangle \mid \langle F \rangle \\ \langle F \rangle &::= \langle P \rangle \langle F \rangle \mid \langle P \rangle \\ \langle P \rangle &::= i \mid \langle E \rangle \end{aligned}$		$\begin{aligned} \langle S \rangle &::= i \langle A \rangle \\ \langle A \rangle &::= \langle E \rangle \\ \langle E \rangle &::= \langle T \rangle \langle E' \rangle \\ \langle E' \rangle &::= + \langle E \rangle \mid \lambda \\ \langle T \rangle &::= \langle F \rangle \langle T' \rangle \\ \langle T' \rangle &::= * \langle T \rangle \mid \lambda \\ \langle F \rangle &::= \langle P \rangle \langle F' \rangle \\ \langle F' \rangle &::= \langle F \rangle \mid \lambda \\ \langle P \rangle &::= i \mid \langle E \rangle \end{aligned}$
--	---	--

59

## Exercício

- Passo 1: a gramática é LL(1)? Se não, transforme-a

$$\begin{aligned} \langle E \rangle &::= \langle T \rangle + \langle E \rangle \mid \langle T \rangle \\ \langle T \rangle &::= a \mid b \end{aligned}$$

60

## Exercício

- Passo 1: a gramática é LL(1)? Se não, transforme-a

$$\begin{array}{l}
 \langle E \rangle ::= \langle T \rangle + \langle E \rangle \mid \langle T \rangle \\
 \langle T \rangle ::= a \mid b
 \end{array}
 \quad \longrightarrow \quad
 \begin{array}{l}
 \langle E \rangle ::= \langle T \rangle \langle E' \rangle \\
 \langle E' \rangle ::= + \langle E \rangle \mid \lambda \\
 \langle T \rangle ::= a \mid b
 \end{array}$$

61

## Exercício

- Passo 2: construa os grafos sintáticos

$$\begin{array}{l}
 \langle E \rangle ::= \langle T \rangle \langle E' \rangle \\
 \langle E' \rangle ::= + \langle E \rangle \mid \lambda \\
 \langle T \rangle ::= a \mid b
 \end{array}$$

62

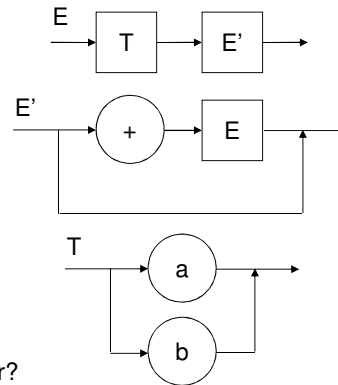
## Exercício

- Passo 2: construa os grafos sintáticos

$\langle E \rangle ::= \langle T \rangle \langle E' \rangle$

$\langle E' \rangle ::= + \langle E \rangle \mid \lambda$

$\langle T \rangle ::= a \mid b$



É possível reduzir?

63

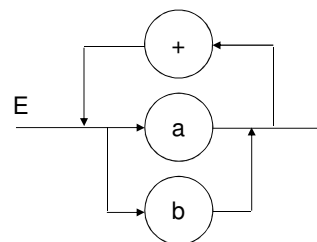
## Exercício

- Passo 2: construa os grafos sintáticos

$\langle E \rangle ::= \langle T \rangle \langle E' \rangle$

$\langle E' \rangle ::= + \langle E \rangle \mid \lambda$

$\langle T \rangle ::= a \mid b$

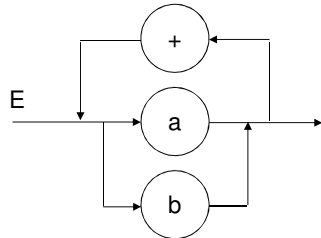


64



## Exercício

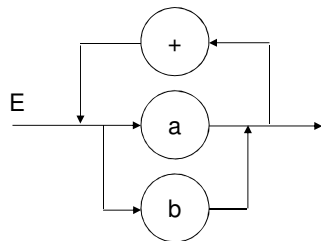
- Passo 3: construa o programa principal e o procedimento para E



65

## Exercício

- Passo 3: construa o programa principal e o procedimento para E



```

procedimento ASD
begin
  obter_próximo();
  E();
  se (terminou_cadeia) então SUCESSO
  senão ERRO;
end

```

```

procedimento E
begin
  se (símbolo='a' ou (símbolo='b')) então
    obter_próximo()
  senão ERRO;
  enquanto (símbolo='+') faça
    obter_próximo();
    se (símbolo='a' ou (símbolo='b')) então
      obter_próximo()
    senão ERRO;
end

```

66

## ASD preditiva

### ■ Exercícios

- A gramática da LALG é LL(1)? Se não é, transforme-a

67

## Exercício

- Construa o(s) grafo(s) (em número reduzido) e o(s) procedimento(s) recursivo(s) para declaração de variáveis na LALG

68