Universidade Federal de Uberlândia
Faculdade de Computação

Análise Sintática (Método Preditivo Baseado em Descida Recursiva)

Curso de Bacharelado em Ciência da Computação
GBC071 - Construção de Compiladores
Prof. Luiz Gustavo Almeida Martins

Descida Recursiva

- Consiste em um **conjunto de procedimentos**
  - **Um procedimento para cada símbolo não-terminal**
    - Reconhece a **parte da entrada derivada dele**
    - Execução começa pelo procedimento do símbolo inicial *S*
  - **Pilha implícita** implementada pelas **chamadas recursivas**

- **Método geral pode exigir retrocesso**
  - **Algoritmo não-determinístico**
    - Produção a ser aplicada é escolhida de forma arbitrária
  - Implementa uma espécie de **busca em profundidade**
    - *Backtrack* no reconhecimento e repetidas leituras sobre a entrada
  - Recursividade à esquerda pode provocar **loop infinito**
  - **Não é usado com frequência**

Descida Recursiva com Retrocesso

- **Algoritmo:** procedimento para o símbolo não-terminal *A*

Início

proxToken ← lex(*W*) // Pega token da cadeia de entrada *W*

Escolha **uma** produção-*A* ($A \rightarrow X_1\ X_2 \ldots X_k$) adequada para proxToken

para (cada símbolo X_i da produção-*A*) **faça**

se (X_i é não terminal) **então**

Ativa o procedimento para X_i

se (X_i é terminal = proxToken) **então**

proxToken ← lex(*W*) // Avança na cadeia

senão

fim_se

fim_para

Fim

se (∃ outras produções-A) **então**

Desfaz procedimentos X_1 até X_{i-1}

Escolha próxima produção-*A*

senão

Trata erro

fim_se

Versão não-determinística

Descida Recursiva com Retrocesso

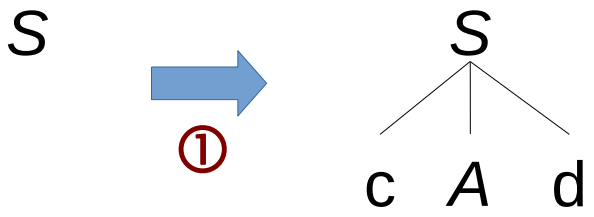
- **Ex:** considere a cadeia **cad** e a gramática *G*:
  - $S \rightarrow cAd$ ①
    $A \rightarrow ab$ ② | a ③

S

Entrada: cad
⇧

Descida Recursiva com Retrocesso

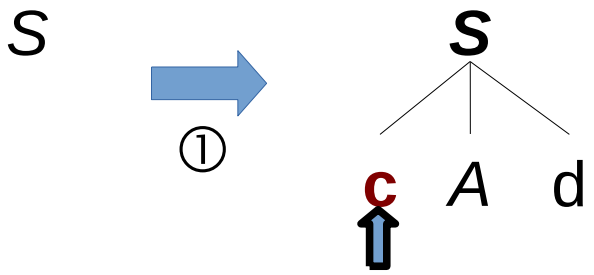
- **Ex:** considere a cadeia **cad** e a gramática *G*:
  - $S \rightarrow cAd$ ①
  - $A \rightarrow ab \mid a$ ② ③



Entrada: cad
⇧

Descida Recursiva com Retrocesso

- **Ex:** considere a cadeia **cad** e a gramática *G*:
  - $S \rightarrow cAd$ ①
  - $A \rightarrow ab \mid a$ ② ③

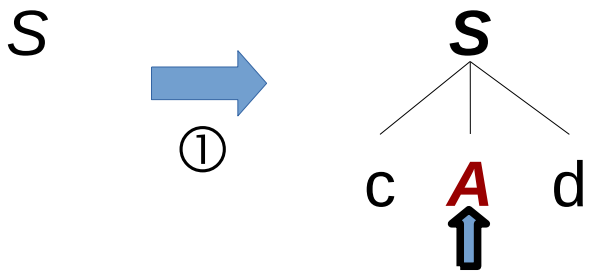


Entrada: cad

casou

Descida Recursiva com Retrocesso

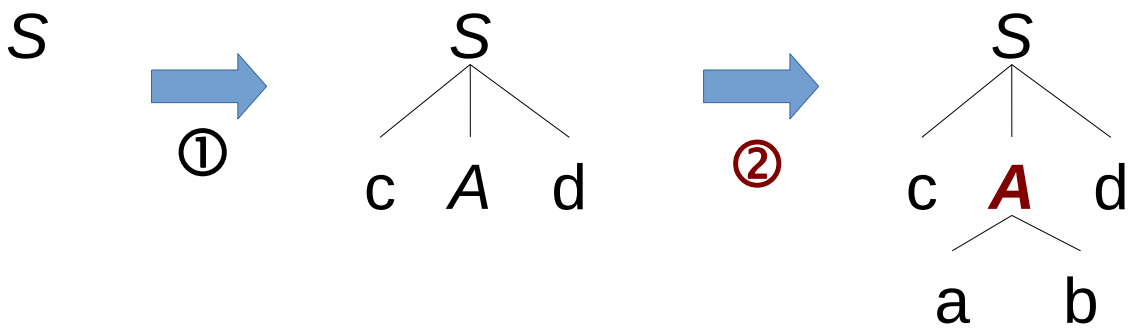
- **Ex:** considere a cadeia **cad** e a gramática *G*:
  - $S \rightarrow cAd$ ①
  - $A \rightarrow ab \mid a$ ② ③



Entrada: cad

Descida Recursiva com Retrocesso

- **Ex:** considere a cadeia **cad** e a gramática *G*:
  - $S \rightarrow cAd$ ①
  - $A \rightarrow ab \mid a$ ② ③

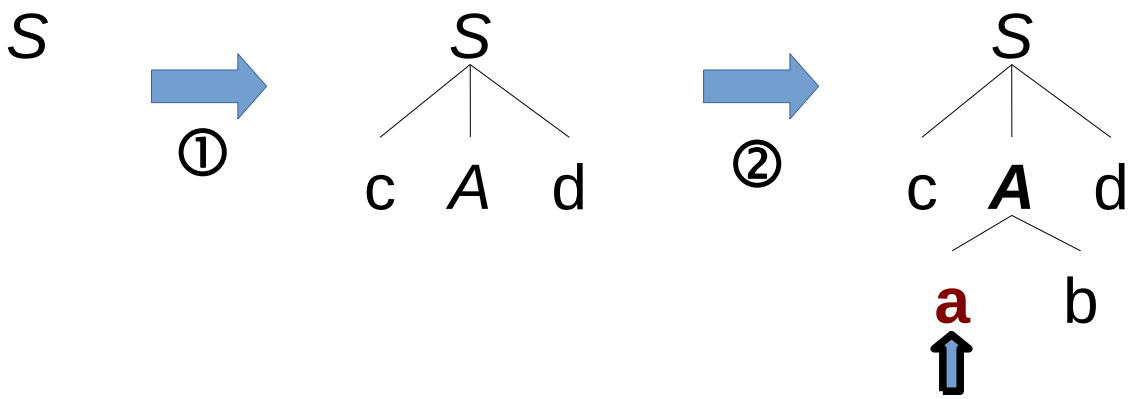


Entrada: cad

⇧

Descida Recursiva com Retrocesso

- **Ex:** considere a cadeia **cad** e a gramática *G*:
  - $S \rightarrow cAd$ ①
  - $A \rightarrow ab \mid a$ ② ③

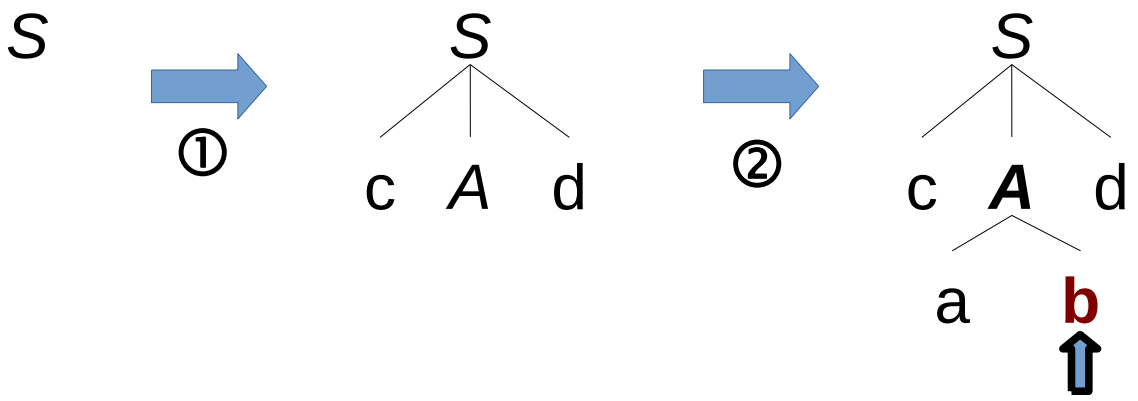


Entrada: cad

casou

Descida Recursiva com Retrocesso

- **Ex:** considere a cadeia **cad** e a gramática *G*:
  - $S \rightarrow cAd$ ①
  - $A \rightarrow ab \mid a$ ② ③

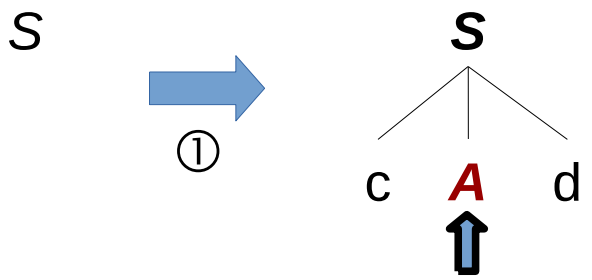


Entrada: cad

não casou

Descida Recursiva com Retrocesso

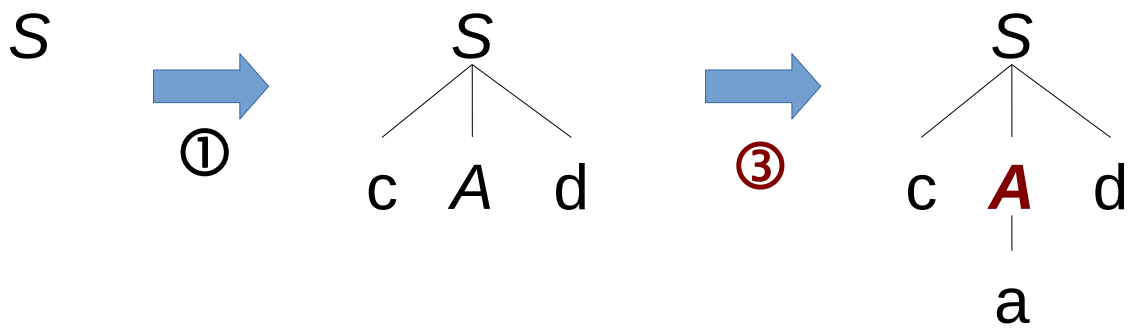
- **Ex:** considere a cadeia **cad** e a gramática *G*:
  - $S \rightarrow cAd$ ① $A \rightarrow ab$ ② | a ③



Entrada: cad

Descida Recursiva com Retrocesso

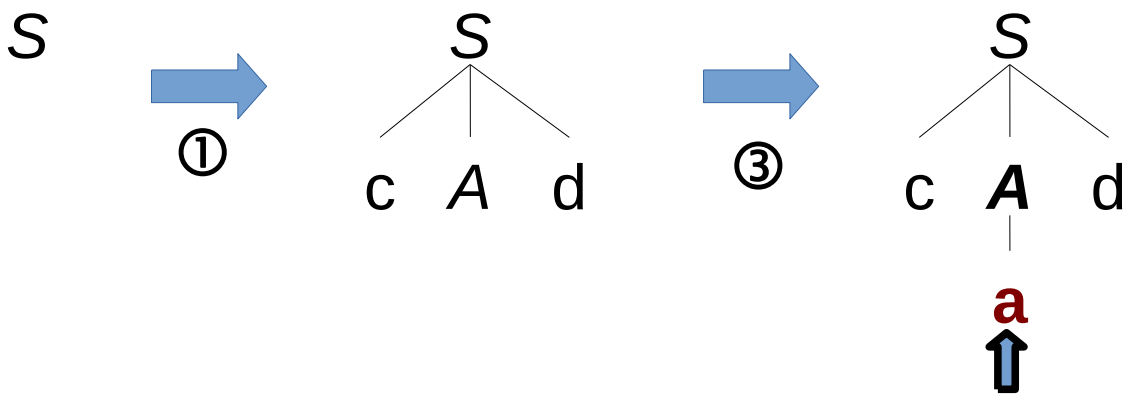
- **Ex:** considere a cadeia **cad** e a gramática *G*:
  - $S \rightarrow cAd$ ①
  - $A \rightarrow ab \mid a$ ② ③



Entrada: cad

Descida Recursiva com Retrocesso

- **Ex:** considere a cadeia **cad** e a gramática *G*:
  - $S \rightarrow cAd$ ①
  - $A \rightarrow ab \mid a$ ② ③

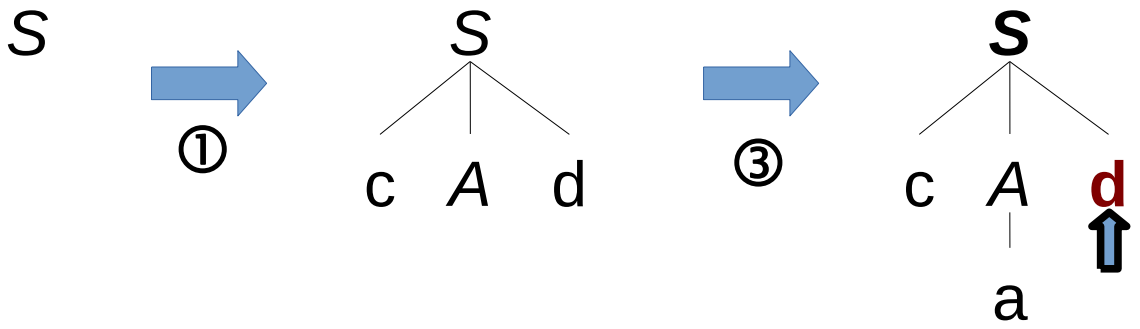


Entrada: cad

casou

Descida Recursiva com Retrocesso

- **Ex:** considere a cadeia **cad** e a gramática *G*:
  - $S \rightarrow cAd$ ① $A \rightarrow ab$ ② $\mid a$ ③



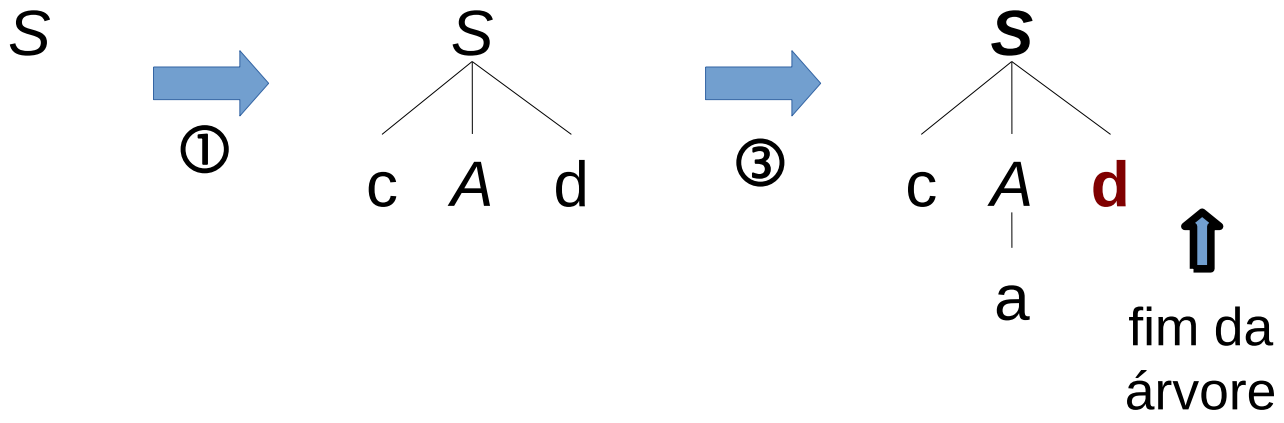
Entrada: cad

casou

Descida Recursiva com Retrocesso

- **Ex:** considere a cadeia **cad** e a gramática *G*:
  - $S \rightarrow cAd$ ① $A \rightarrow ab \mid a$ ② ③

cadeia aceita



Entrada: cad

Descida Recursiva sem Retrocesso

- **Eliminação do retrocesso** aumenta eficiência
  - Fácil de **implementar manualmente**
  - Produz **código mais legível** que facilita a manutenção
- Adota **gramáticas determinísticas**
  - **Sem recursão à esquerda**
  - **Ambiguidade controlada**
- **Método formal** de construção dos procedimentos envolve **regras de transformação** e **de tradução**

Descida Recursiva sem Retrocesso

- **Algoritmo:** procedimento para o símbolo não-terminal *A*

Início

proxToken ← lex(*W*) // Pega token da cadeia de entrada *W*

Escolha **uma** produção-*A* ($A \rightarrow X_1 X_2 \ldots X_k$) adequada para proxToken

para (cada símbolo X_i da produção-*A*) **faça**

se (X_i é não terminal) **então**

Ativa o procedimento para X_i

se (X_i é terminal = proxToken) **então**

proxToken ← lex(*W*) // Avança na cadeia

senão

Trata erro

fim_se

fim_para

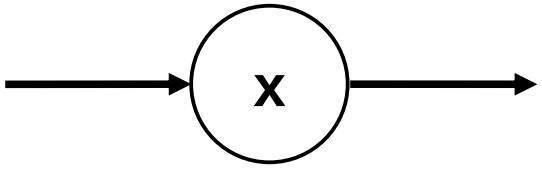
Fim

Versão determinística

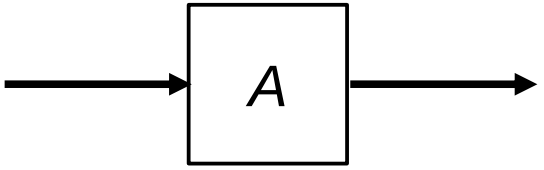
Regras de Transformação

- Mapeamento das produções em **grafos sintáticos**
  - **Cada produção da gramática é mapeada em um grafo**

- Símbolos terminais e não terminais são diferenciados pela **forma geométrica do nó no grafo**
  - Representa a diferença no seu tratamento (**ações**)



Símbolo terminal

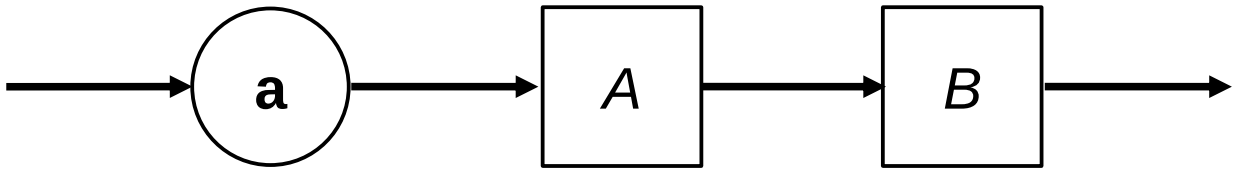


Símbolo não terminal

Regras de Transformação

- **Concatenação:** símbolos são combinados de modo a representar a produção mapeada

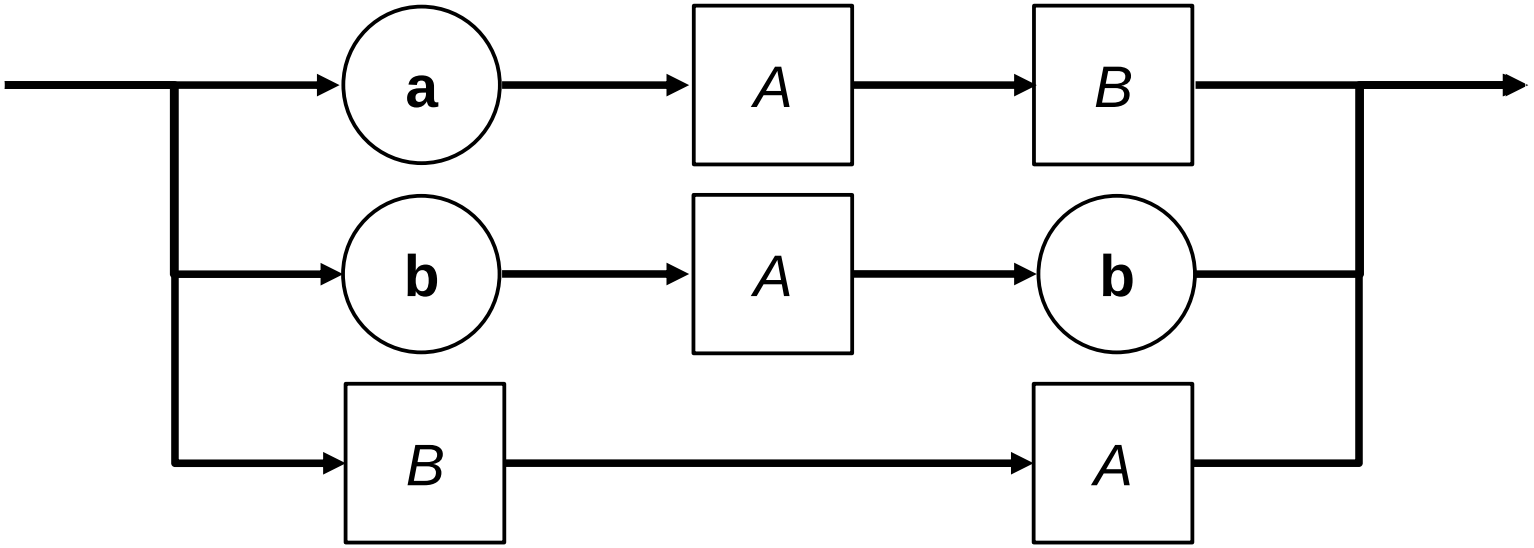
  - **Ex:** $P \rightarrow \mathbf{a}AB$



Grafo de *P*

Regras de Transformação

- **União:** alternativas de produção para um mesmo não terminal podem ser combinadas em um único grafo
  - Reduzir o número de grafos simplifica e aumenta a eficiência da implementação
  - **Ex:** $P \rightarrow \mathbf{a}AB \mid \mathbf{b}A\mathbf{b} \mid BA$

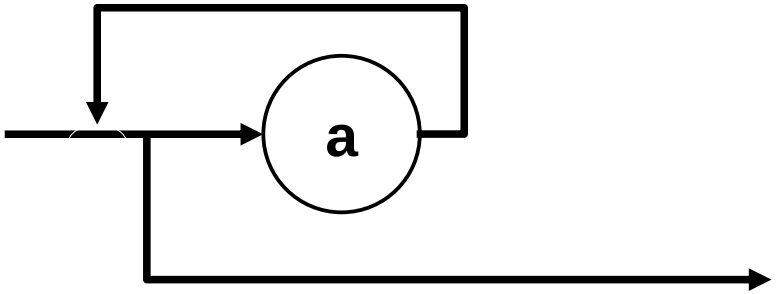


Grafo de *P*

Regras de Transformação

- **Fecho:** produções recursivas são representadas como repetições
  - **Ex:** $P \rightarrow \mathbf{a}P \mid \boldsymbol{\varepsilon}$ (equivalente a **a* ou {a}***)

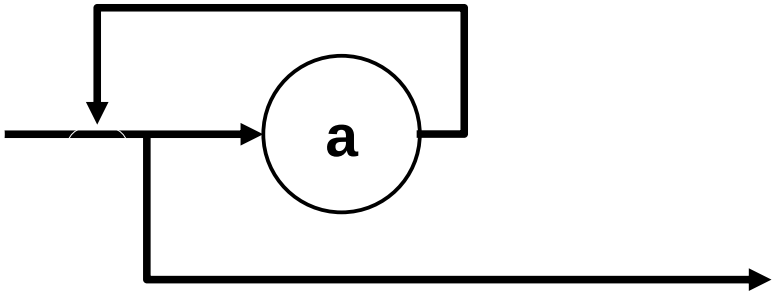
Grafo de *P*



Regras de Transformação

- **Fecho:** produções recursivas são representadas como repetições
  - **Ex:** $P \rightarrow \mathbf{a}P \mid \boldsymbol{\varepsilon}$ (equivalente a **a* ou {a}***)

Grafo de *P*



Representa a **produção-ε**

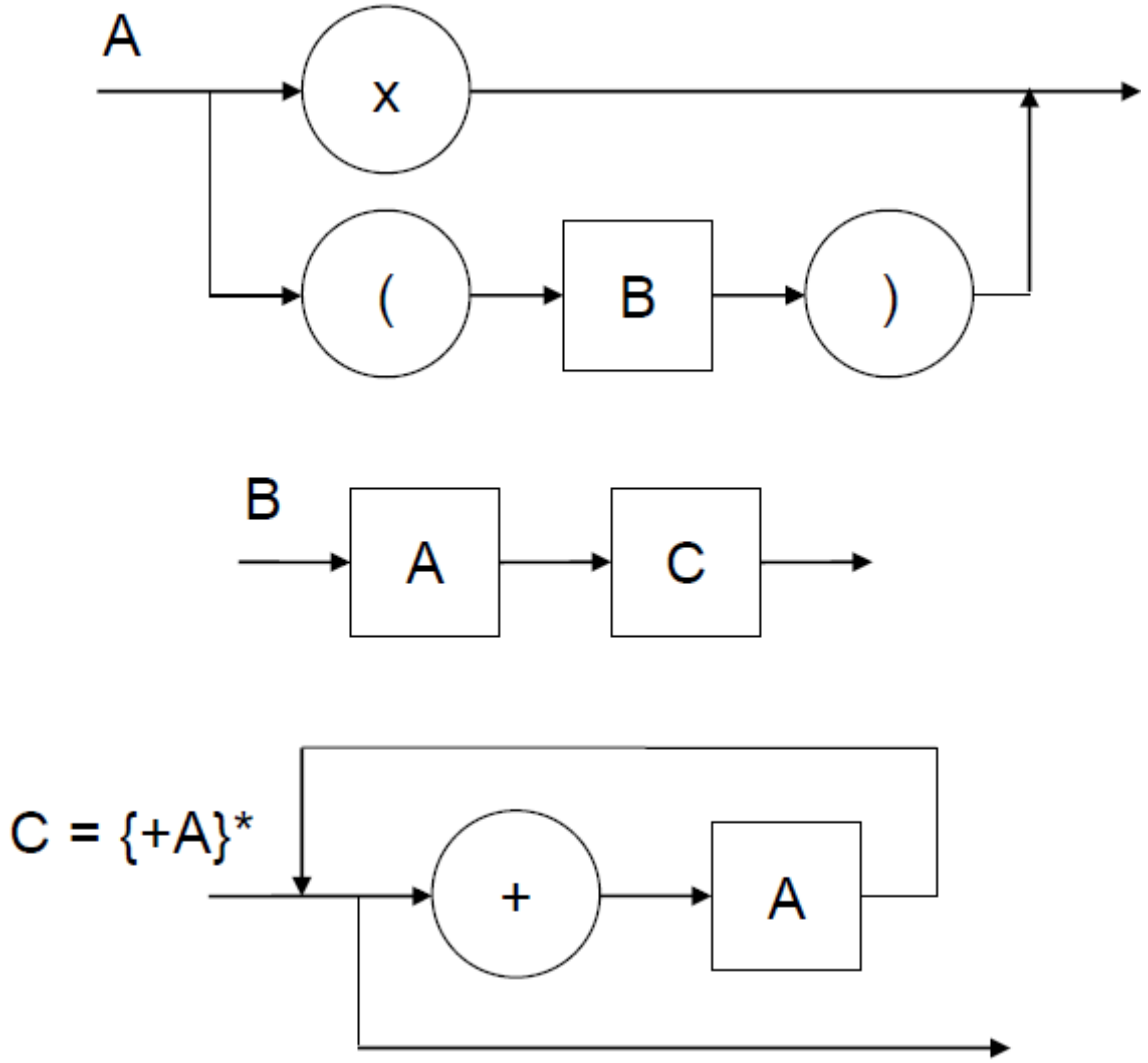
Regras de Transformação

- **Ex:** construa os grafos para as produções da gramática *G*:
  - $A \rightarrow \mathbf{x} \mid \mathbf{(}\ B\ \mathbf{)}$
  - $B \rightarrow AC$
  - $C \rightarrow \mathbf{+}AC \mid \boldsymbol{\varepsilon}$

- (Fonte: [Aluisio, 2011])

Regras de Transformação

- **Ex:** construa os grafos para as produções da gramática *G*:
  - $A \rightarrow \mathbf{x} \mid \mathbf{(}\ B\ \mathbf{)}$
  - $B \rightarrow AC$
  - $C \rightarrow \mathbf{+}AC \mid \boldsymbol{\varepsilon}$



- (Fonte: [Aluisio, 2011])

Regras de Tradução

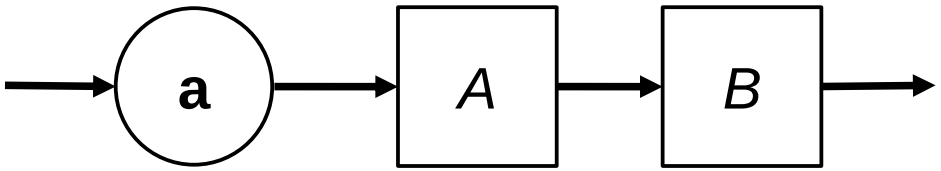
- Mapeamento dos grafos em procedimentos
  - **Um procedimento para cada grafo**

- Ocorrências de um **terminal x** envolvem **seu reconhecimento na cadeia de entrada e a leitura do próximo *token***
  - ***se*** *(proxToken = x)* ***então*** *proxToken ← lex(W)*
  - ***senão*** *trata erro*

- Ocorrências de um **não terminal *A*** envolvem a **chamada do procedimento de *A*** (**análise imediata**)
  - *Procedimento_A*

Regras de Tradução

- **Concatenação:** símbolos são implementados na sequência
  - **Ex:** $P \rightarrow \mathbf{a}AB$



Grafo de *P*

Procedimento_*P*

se (proxToken = **a**) **então**
proxToken ← lex(*W*)

Procedimento_*A*

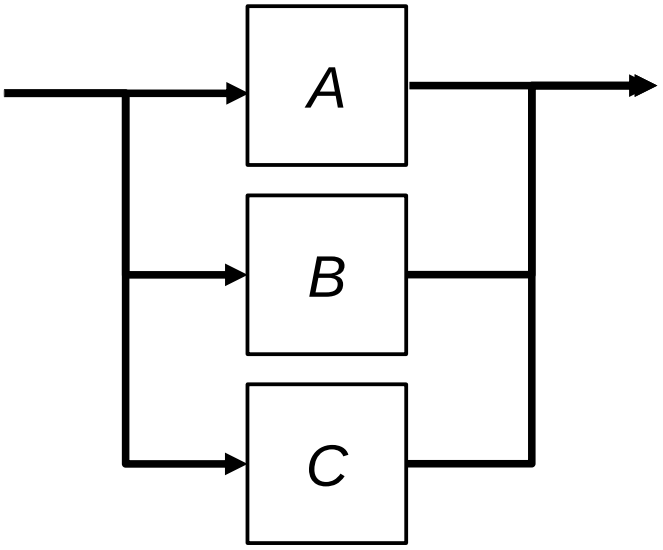
Procedimento_*B*

senão
Trata Erro
fim_se

Fim

Regras de Tradução

- **União:** implementada através de **comandos de seleção** (**tomada de decisão**)
  - **Ex:** $P \rightarrow A \mid B \mid C$



Grafo de *P*

Procedimento_*P*

se (proxToken está em **FIRST(*A*)**) **então**
Procedimento_*A*

senão se (proxToken está em **FIRST(*B*)**) **então**
Procedimento_*B*

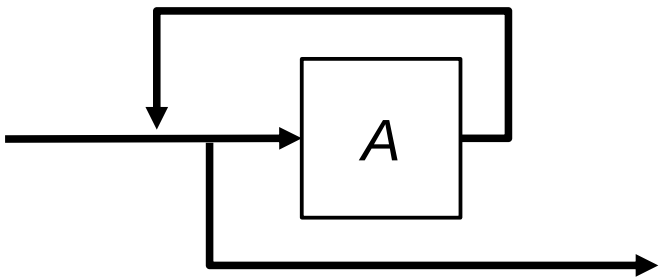
senão se (proxToken está em **FIRST(*C*)**) **então**
Procedimento_*C*

senão
Trata Erro
fim_se

Fim

Regras de Tradução

- **Fecho:** implementada através de **comando de repetição** (**loop**)
  - **Ex:** $P \rightarrow AP \mid \varepsilon$



Grafo de *P*

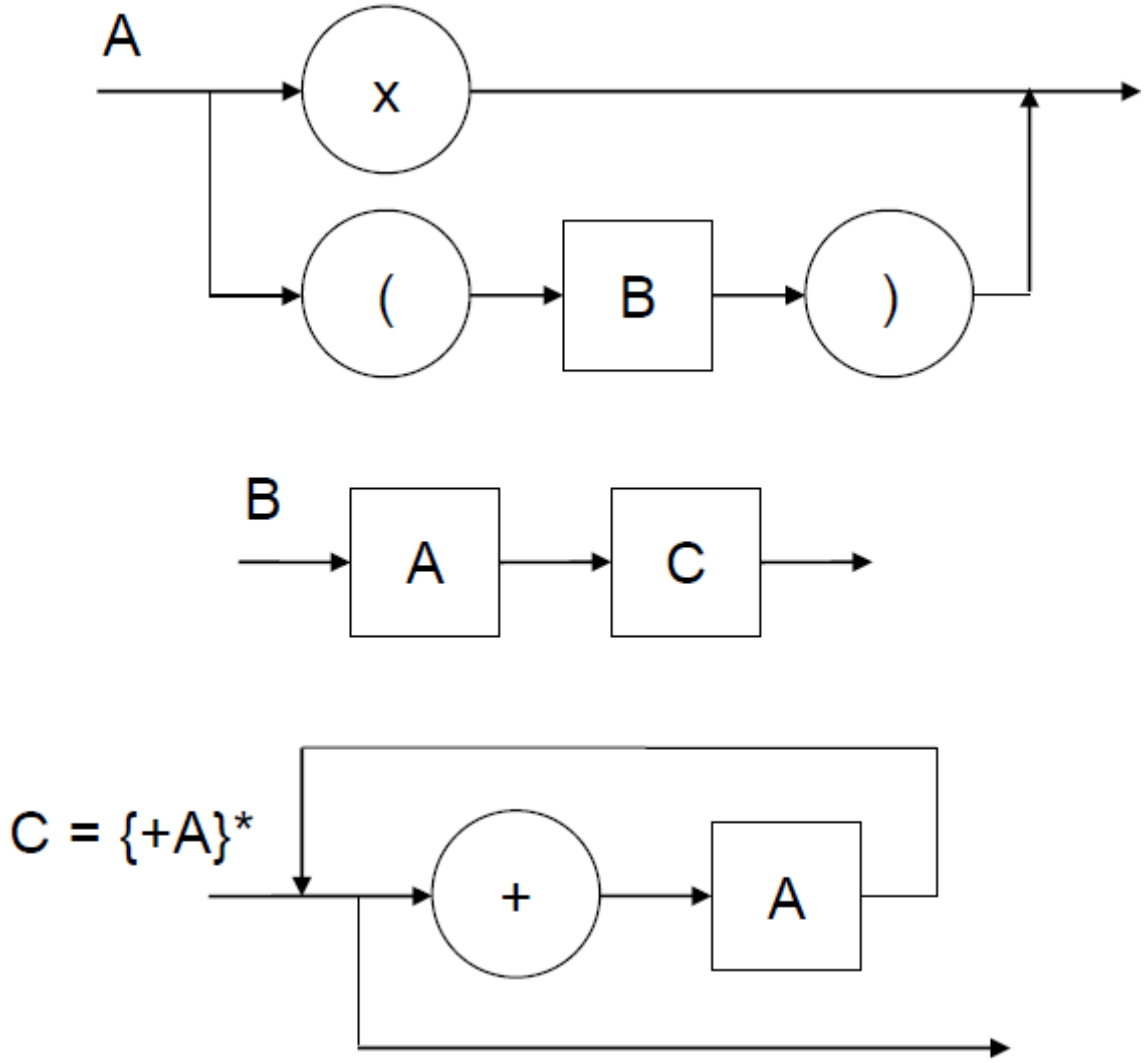
Procedimento_*P*

enquanto (proxToken está em **FIRST(*A*)**) **faça**
Procedimento_*A*

Fim

Regras de Tradução

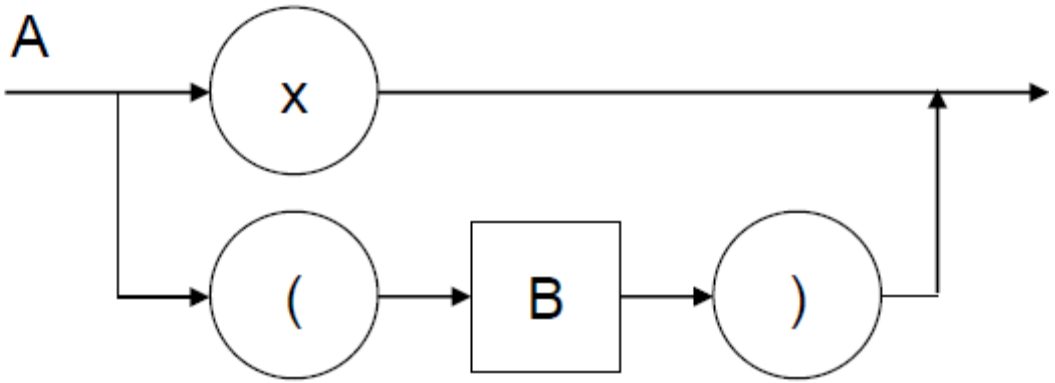
- **Ex:** construa os procedimentos para os grafos das produções de *G*:
  - $A \rightarrow \mathbf{x} \mid \mathbf{(}\ B\ \mathbf{)}$
  - $B \rightarrow AC$
  - $C \rightarrow \mathbf{+}AC \mid \boldsymbol{\varepsilon}$



- (Fonte: [Aluisio, 2011])

Regras de Tradução

- **Ex:** construa os procedimentos para os grafos das produções de *G*:
  - $A \rightarrow \mathbf{x} \mid \mathbf{(}\ B\ \mathbf{)}$



- (Fonte: [Aluisio, 2011])

Procedimento_*A*

se (proxToken = "**x**") **então**
    proxToken ← lex(*W*)

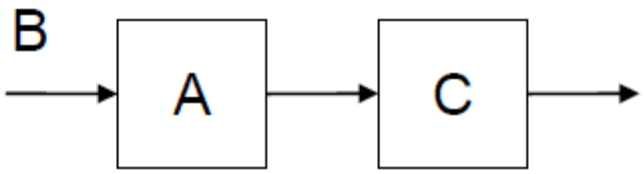
senão se (proxToken = "**(**") **então**
    proxToken ← lex(*W*)
    procedimento_*B*
    se (proxToken = "**)**") **então**
        proxToken ← lex(*W*)
    senão
        mostra("Erro:) *esperado*")

senão
    mostra("Erro: x *ou (esperados*")
fim_se

Fim

Regras de Tradução

- **Ex:** construa os procedimentos para os grafos das produções de *G*:
  - $B \rightarrow AC$



Procedimento_*B*

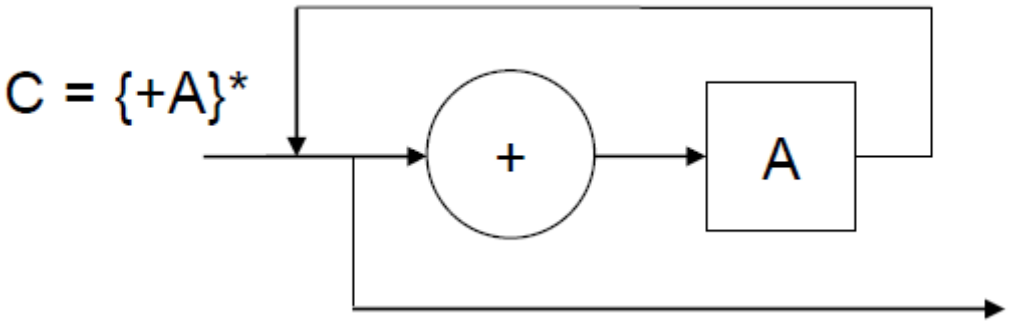
procedimento_*A*
procedimento_*C*

Fim

- (Fonte: [Aluisio, 2011])

Regras de Tradução

- **Ex:** construa os procedimentos para os grafos das produções de *G*:
  - $C \rightarrow +AC \mid \varepsilon$



Procedimento_*C*

enquanto (proxToken = "**+**") **faça**
proxToken ← lex(*W*)

procedimento_*A*

fim_enquanto

Fim

- (Fonte: [Aluisio, 2011])

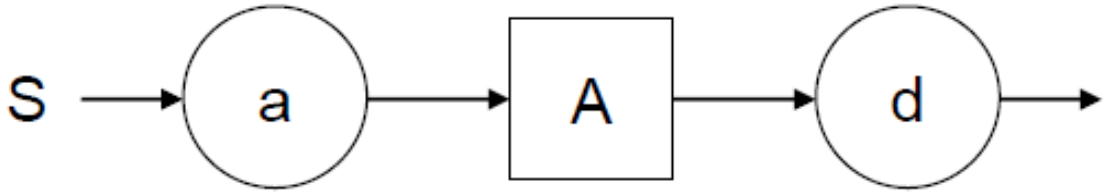
Exercício

- Construa o analisador sintático preditivo baseado em descida recursiva para as produções de uma gramática *G*:
  - $S \rightarrow \mathbf{a}A\mathbf{d}$
  - $A \rightarrow \mathbf{c}A \mid \mathbf{e}B$
  - $B \rightarrow \mathbf{f} \mid \mathbf{g}$

- (Fonte: [Aluisio, 2011])

Exercício

- Construa o analisador sintático preditivo baseado em descida recursiva para as produções de uma gramática *G*:
  - $S \rightarrow \mathbf{a}A\mathbf{d}$
  - $A \rightarrow \mathbf{c}A \mid \mathbf{e}B$
  - $B \rightarrow \mathbf{f} \mid \mathbf{g}$

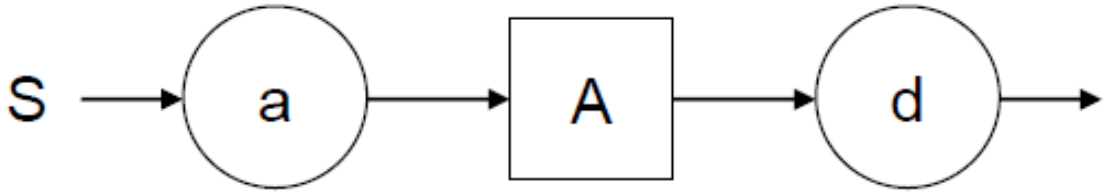


Grafo Sintático

- (Fonte: [Aluisio, 2011])

Exercício

- Construa o analisador sintático preditivo baseado em descida recursiva para as produções de uma gramática *G*:
  - $S \rightarrow \mathbf{a}A\mathbf{d}$
  - $A \rightarrow \mathbf{c}A \mid \mathbf{e}B$
  - $B \rightarrow \mathbf{f} \mid \mathbf{g}$



Grafo Sintático

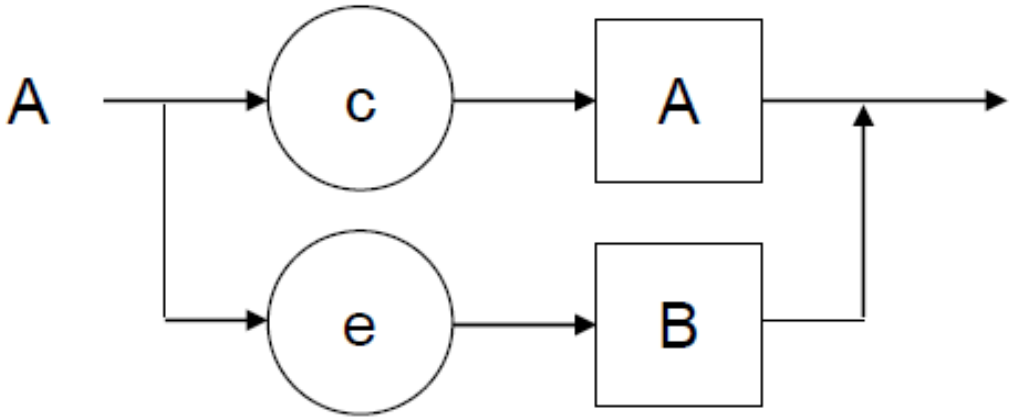
Procedimento_*S*

```
se (proxToken = "a") então
    proxToken ← lex(W)

    procedimento_A

    se (proxToken = "d") então
        proxToken ← lex(W)
    senão
        mostra("Erro: d esperado")
    fim_se

senão
    mostra("Erro: a esperado")
fim_se

Fim
```

- (Fonte: [Aluisio, 2011])

Exercício

- Construa o analisador sintático preditivo baseado em descida recursiva para as produções de uma gramática *G*:
  - $S \rightarrow \mathbf{a}A\mathbf{d}$
  - $A \rightarrow \mathbf{c}A \mid \mathbf{e}B$
  - $B \rightarrow \mathbf{f} \mid \mathbf{g}$

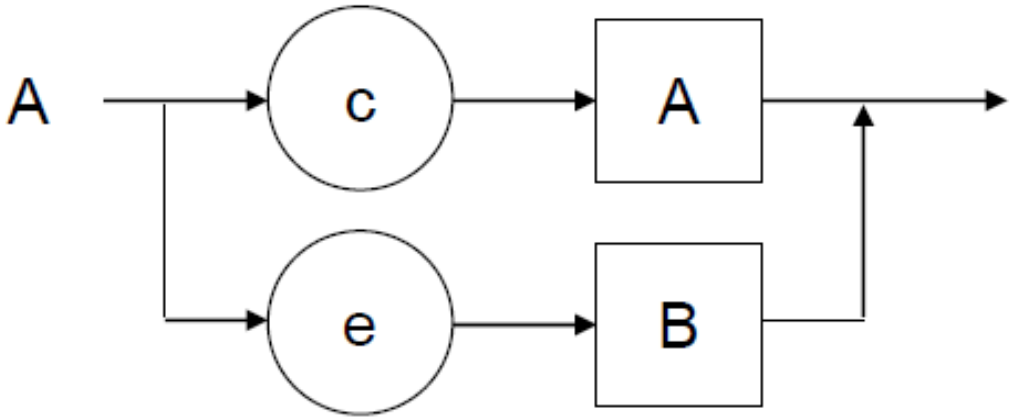


Grafo Sintático

- (Fonte: [Aluisio, 2011])

Exercício

- Construa o analisador sintático preditivo baseado em descida recursiva para as produções de uma gramática *G*:
  - $S \rightarrow \mathbf{a}A\mathbf{d}$
  - $A \rightarrow \mathbf{c}A \mid \mathbf{e}B$
  - $B \rightarrow \mathbf{f} \mid \mathbf{g}$



Grafo Sintático

Procedimento_*A*

se (proxToken = "**c**") **então**
proxToken ← lex(*W*)

procedimento_*A*

senão se (proxToken = "**e**") **então**
proxToken ← lex(*W*)

procedimento_*B*

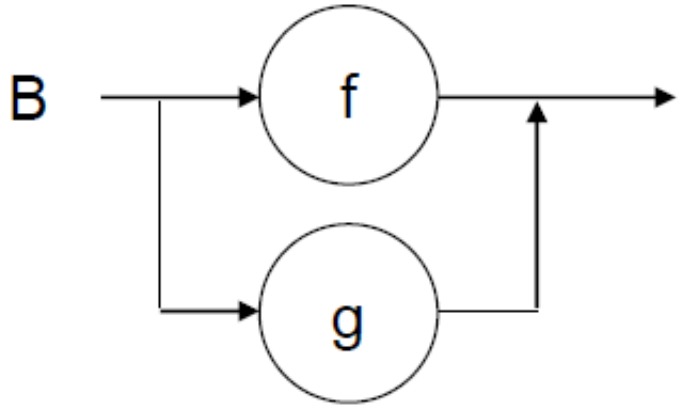
senão
mostra("Erro: *c ou e esperados*")
fim_se

Fim

- (Fonte: [Aluisio, 2011])

Exercício

- Construa o analisador sintático preditivo baseado em descida recursiva para as produções de uma gramática *G*:
  - $S \rightarrow \mathbf{a}A\mathbf{d}$
  - $A \rightarrow \mathbf{c}A \mid \mathbf{e}B$
  - $B \rightarrow \mathbf{f} \mid \mathbf{g}$

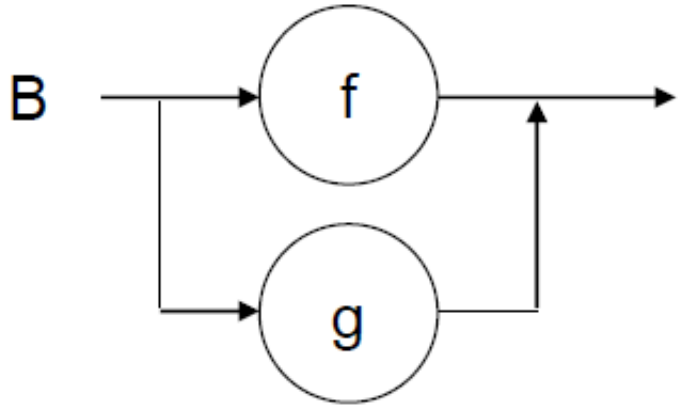


Grafo Sintático

- (Fonte: [Aluisio, 2011])

Exercício

- Construa o analisador sintático preditivo baseado em descida recursiva para as produções de uma gramática *G*:
  - $S \rightarrow \mathbf{a}A\mathbf{d}$
  - $A \rightarrow \mathbf{c}A \mid \mathbf{e}B$
  - $B \rightarrow \mathbf{f} \mid \mathbf{g}$



Grafo Sintático

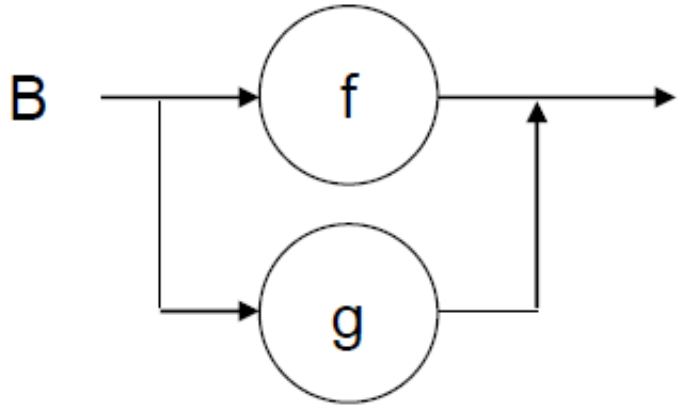
Procedimento_*B*

```
se (proxToken = "f" ou "g") então
    proxToken ← lex(W)
senão
    mostra("Erro: f ou g esperados")
fim_se
Fim
```

- (Fonte: [Aluisio, 2011])

Exercício

- Construa o analisador sintático preditivo baseado em descida recursiva para as produções de uma gramática *G*:
  - $S \rightarrow \mathbf{a}A\mathbf{d}$
  - $A \rightarrow \mathbf{c}A \mid \mathbf{e}B$
  - $B \rightarrow \mathbf{f} \mid \mathbf{g}$



Grafo Sintático

- (Fonte: [Aluisio, 2011])

Procedimento_*B*

se (proxToken = "**f**" ou "**g**") **então**
proxToken ← lex(*W*)
senão
mostra("Erro: *f ou g esperados*")
fim_se
Fim

ASP_Recursivo
proxToken ← lex(*W*)

procedimento_S

se (proxToken = "**$**") **então**
mostra("*Cadeia aceita*")
senão
mostra("*Cadeia rejeitada*")
fim_se
Fim

Referências Bibliográficas

- Aho, A.V.; Lam, M.S.; Sethi, R.; Ullman, J.D. Compiladores: Princípios, técnicas e ferramentas, 2 ed., Pearson, 2008
- Alexandre, E.S.M. Livro de Introdução a Compiladores, UFPB, 2014
- Aluisio, S. material da disciplina “Teoria da Computação e Compiladores”, ICMC/USP, 2011
- Dubach, C. material da disciplina “*Compiling Techniques*”, University of Edinburgh, 2018
- Freitas, R. L. notas de aula - Compiladores, PUC Campinas, 2000
- Menezes, P.B. Linguagens Formais e Autômatos, 6 ed., Bookman, 2011
- Ricarte, I. Introdução à Compilação, Elsevier, 2008