# Gramáticas Livres de Contexto

Prof$^a$ Jerusa Marchi

jerusa.marchi@ufsc.br

Departamento de Informática e Estatística

Universidade Federal de Santa Catarina

# Gramática Livre de Contexto

- Definição Formal:
  - Uma *GLC* é uma quádrupla $G = (N, T, P, S)$ onde:
    - $N$: símbolos não terminais
    - $T$: símbolos terminais ou alfabeto
    - $S$: símbolo inicial
    - $P$: conjunto de regras de produção, tal que

$$P = \{A ::= \delta, \text{ com } A \in N \text{ e } \delta \in (T \cup N)^*\}$$

# Gramática Livre de Contexto

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

$$E ::= E + T \mid E - T \mid T$$
$$T ::= T * F \mid T/F \mid F$$
$$F ::= (E) \mid id$$

# Derivações

- Usa-se o símbolo $\Rightarrow$ para indicar a derivação de palavras a partir da cabeça para o corpo da produção.
- Dada uma palavra de terminais e não-terminais $\alpha A \beta$, onde $A$ é um não-terminal, ou seja $\alpha$ e $\beta$ são palavras em $(N \cup T)^*$ e $A$ está em $N$.
- Se $A\ := \gamma$ é uma produção de $G$, então $\alpha A \beta \Rightarrow \alpha \gamma \beta$
- $\Rightarrow$ significa "deriva em um passo"
- $\stackrel{*}{\Rightarrow}$ significa "deriva em zero ou mais passos"

$$E \stackrel{*}{\Rightarrow} id * (id + id)$$

# Derivações

- **Derivação mais à esquerda** significa substituir o símbolo não-terminal *mais à esquerda* a cada derivação ($\underset{lm}{\Rightarrow}$)
  - Exemplo:

$$E \underset{lm}{\Rightarrow} T \underset{lm}{\Rightarrow} T * F \underset{lm}{\Rightarrow} F * F \underset{lm}{\Rightarrow} id * F \underset{lm}{\Rightarrow} id * (E) \underset{lm}{\Rightarrow} id * (E + T) \underset{lm}{\Rightarrow} id*(T+T) \underset{lm}{\Rightarrow} id*(F+T) \underset{lm}{\Rightarrow} id*(id+T) \underset{lm}{\Rightarrow} id*(id+F) \underset{lm}{\Rightarrow} id*(id+id)$$

- **Derivação mais à direita** significa substituir o símbolo não-terminal *mais à direita* a cada derivação ($\underset{rm}{\Rightarrow}$)
  - Exemplo:

$$E \underset{rm}{\Rightarrow} T \underset{rm}{\Rightarrow} T*F \underset{rm}{\Rightarrow} T*(E) \underset{rm}{\Rightarrow} T*(E+T) \underset{rm}{\Rightarrow} T*(E+F) \underset{rm}{\Rightarrow} T*(E+id) \underset{rm}{\Rightarrow} T*(F+id) \underset{rm}{\Rightarrow} T*(id+id) \underset{rm}{\Rightarrow} F*(id+id) \underset{rm}{\Rightarrow} id*(id+id)$$

# Árvores Gramaticais

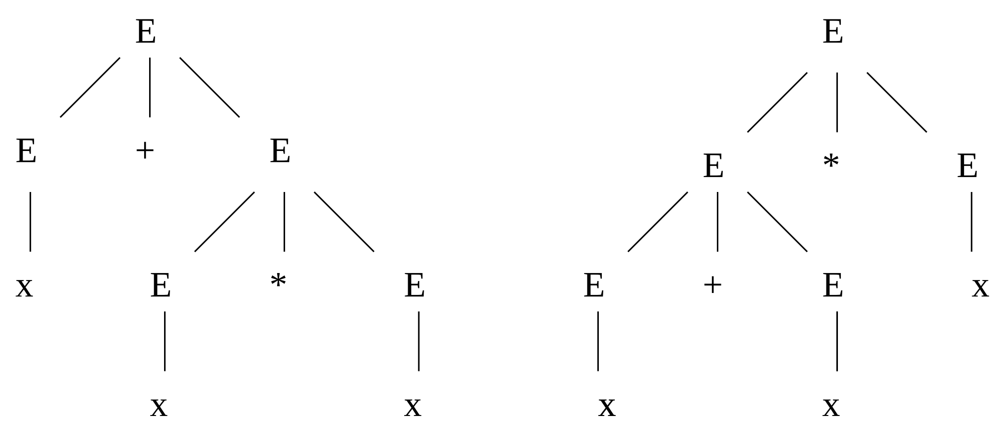
- Uma árvore gramatical é a representação gráfica de uma derivação onde:
  - cada nó interior da árvore é um não-terminal $A$ onde seus filhos são rotulados, da esquerda para a direita, pelos símbolos do lado direito da produção pelos quais $A$ foi substituido na derivação
  - cada nó folha é um não-terminal, um terminal ou $\varepsilon$ (onde então deve ser o único nó filho)

# Gramáticas Ambíguas

- Uma gramática é ambígua se permite construir mais de uma árvore de derivação para a mesma sentença.

$$G = (\{E\}, \{+, *, x\}, P, E)$$

$$x + x * x$$




- Uma GLC é ambígua se existe alguma sentença com mais de uma derivação possível.

# Gramáticas Ambíguas

- Algumas gramáticas são inerentemente ambiguas, ou seja, a definição da linguagem implica uma ambiguidade natural
  - Exemplo: $L(G) = \{a^n b^m c^k | n, m, k \geq 1$ e $n = m$ ou $m = k\}$

$$S ::= abc | aAbC | A'bBc$$
$$A ::= aAb | ab$$
$$A' ::= aA' | a$$
$$B ::= bBc | bc$$
$$C ::= cC | c$$

  - $aabbcc$

# Gramáticas Ambíguas

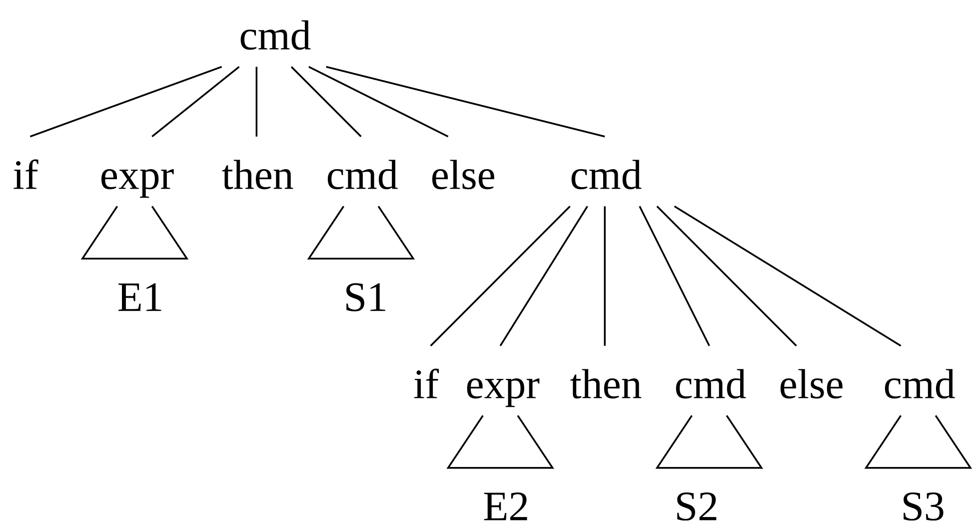
- Problema
  - Reconhecedores exigem derivações unívocas para obter bom desempenho ou mesmo para concluir a análise sintática
  - Nas linguagens de programação parte do significado dos comandos está especificada em sua estrutura sintática (existe semântica na estrutura do programa).
  - Como eliminar a ambiguidade?
    - Para o exemplo anterior, $*$ precede a $+$, então incorporamos essa informação na gramática.

    1. $E ::= E + T$
    2. $T ::= T * F$
    3. $F ::= (E)|$dígito

# Eliminando a Ambiguidade

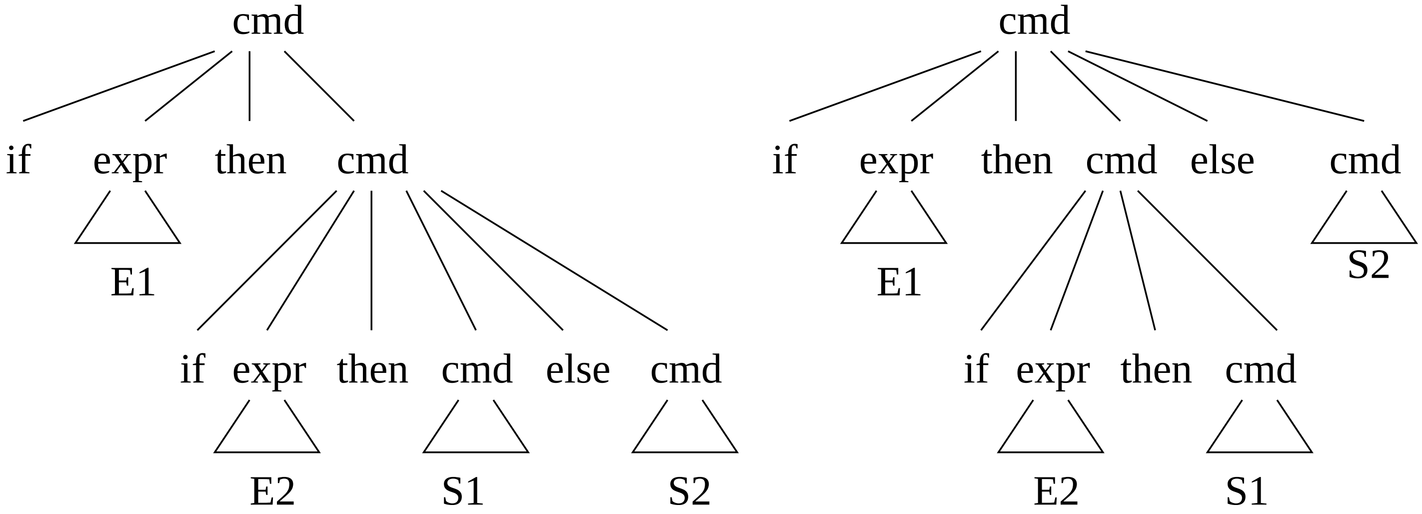
- Outro exemplo:
  1. cmd ::= **if** expr **then** cmd
     | **if** expr **then** cmd **else** cmd
     | **outro**

**if** $E_1$ **then** $S_1$ **else if** $E_2$ **then** $S_2$ **else** $S_3$



# Eliminando a Ambiguidade

**if** $E_1$ **then if** $E_2$ **then** $S_1$ **else** $S_2$



# Eliminando a Ambiguidade

- Regra geral: Associar cada **else** ao **then** anterior mais próximo ainda não associado
- Incorporando a regra geral na gramática:
  1. cmd ::= cmd-associado
     | cmd-não associado
  2. cmd-associado ::= **if** expr **then** cmd-associado **else** cmd-associado
     | **outro**
  3. cmd-não associado ::= **if** expr **then** cmd
     | **if** expr **then** cmd-associado **else** cmd-não associado

# Simplificações em GLC

- É possível simplificar uma GLC sem reduzir o seu poder expressivo. Se L é uma LLC não vazia, então L pode ser gerada por uma GLC G com as seguintes propriedades:
  - Cada não-terminal e cada terminal de $G$ aparecem na derivação de alguma palavra de $L$ (não há símbolos inúteis)
  - Se $\varepsilon \notin L$ então não há necessidade de produções da forma $A ::= \varepsilon$ ($\varepsilon$-Livre)
  - Não há produções da forma $A ::= B$ onde $A$ e $B$ são não-terminais (não há produções unitárias)
  - Não há produções "não determinísticas" na gramática (Fatoração)
  - Não há produções recursivas, ou seja um não terminal não deriva a si próprio direta ou indiretamente

# Simplificações em GLC

- É possível simplificar uma GLC sem reduzir o seu poder expressivo. Se L é uma LLC não vazia, então L pode ser gerada por uma GLC G com as seguintes propriedades:
  - **Cada não-terminal e cada terminal de $G$ aparecem na derivação de alguma palavra de $L$ (não há símbolos inúteis)**
  - Se $\varepsilon \notin L$ então não há necessidade de produções da forma $A ::= \varepsilon$ ($\varepsilon$-Livre)
  - Não há produções da forma $A ::= B$ onde $A$ e $B$ são não-terminais (não há produções unitárias)
  - Não há produções “não determinísticas” na gramática (Fatoração)
  - Não há produções recursivas, ou seja um não terminal não deriva a si próprio direta ou indiretamente

# Simplificações em GLC

- Eliminação de símbolos inúteis
  - Um símbolo $X$ é útil se existe uma derivação

$$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$$

    para algum $w, \alpha$ e $\beta$, onde $w$ é uma cadeia de $T^*$ e $\alpha$ e $\beta$ são cadeias quaisquer de não-terminais e teminais.
  - Há dois tipos de símbolos inúteis:
    - Símbolos improdutivos - não geram nenhuma cadeia de terminais
    - Símbolos inalcançáveis - jamais são gerados a partir de $S$

# Simplificações em GLC

- Eliminação de símbolos improdutivos

  Entrada GLC $G = (N, T, P, S)$

  Saída GLC $G' = (N', T, P', S)$

  $SP := T \cup \{\varepsilon\}$

  Repita

  1. $Q := \{X \mid X \in N$ e $X \notin SP$ e existe pelo menos uma produção $X ::= X_1 X_2 ... X_n$ tal que $X_1, X_2, ..., X_n \in SP\}$
  2. $SP := SP \cup Q$

  Até $Q = \emptyset$

  $N' := SP \cap N$

  Se $S \in SP$ então

  $P' := \{p \mid p \in P$ e todos os símbolos de $p \in SP\}$

  senão "$L(G) = \emptyset$" e $P' := \emptyset$

# Simplificações em GLC

- Eliminação de símbolos improdutivos
  - raciocínio similar a eliminar estados mortos de um autômato
    - marque os símbolos terminais da gramática em todas as produções
    - repita de modo iterativo, até que nenhum novo não-terminal seja marcado
      - marque as cabeças das produções, cujo corpo esteja completamente marcado
      - marque, no corpo das produções, os não-terminais marcados no passo anterior
    - faça a nova gramática ser igual as produções totalmente marcadas

# Simplificações em GLC

- Eliminação de símbolos improdutivos
  - Exemplo:

$$\begin{array}{l|l} S ::= ABB|CAC & S ::= CAC \\ A ::= a & A ::= a \\ B ::= Bc|ABB & C ::= a \\ C ::= bB|a & \end{array}$$

# Simplificações em GLC

- Eliminação de símbolos inalcançáveis
  Entrada GLC $G = (N, T, P, S)$
  Saída GLC $G' = (N', T', P', S)$

  $SA := \{S\}$
  Repita

  1. $M := \{X \mid X \in N \cup T$ e $X \notin SA$ e existe pelo menos uma produção $Y ::= \alpha X \beta$ e $Y \in SA\}$
  2. $SA := SA \cup M$

  Até $M = \emptyset$
  $N' := SA \cap N$
  $T' := SA \cap T$
  $P' := \{p \mid p \in P$ e todos os símbolos de $p \in SA\}$

# Simplificações em GLC

- Eliminação de símbolos inalcançáveis
  - raciocínio similar ao de eliminar os estados inalcançáveis em um autômato
    - marque o símbolo inicial da gramática
    - marque o corpo das produções derivadas diretamente a partir de S
    - repita de modo iterativo, até que nenhum novo símbolo seja marcado
      - marque o corpo das produções cujas cabeças (não-terminais) foram marcadas anteriormente
    - faça a nova gramática ser igual as produções totalmente marcadas

# Simplificações em GLC

- Eliminação de símbolos inalcançáveis
  - Exemplo:

$$
\begin{array}{l|l}
S ::= aS|SB|SS|b & S ::= aS|SB|SS|b \\
A ::= ASB|c & B ::= b \\
B ::= b &
\end{array}
$$

# Simplificações em GLC

- É possível simplificar uma GLC sem reduzir o seu poder expressivo. Se L é uma LLC não vazia, então L pode ser gerada por uma GLC $G$ com as seguintes propriedades:
  - Cada não-terminal e cada terminal de $G$ aparecem na derivação de alguma palavra de $L$ (não há símbolos inúteis)
  - **Se $\varepsilon \notin L$ então não há necessidade de produções da forma $A ::= \varepsilon$ ($\varepsilon$-Livre)**
  - Não há produções da forma $A ::= B$ onde $A$ e $B$ são não-terminais (não há produções unitárias)
  - Não há produções "não determinísticas" na gramática (Fatoração)
  - Não há produções recursivas, ou seja um não terminal não deriva a si próprio direta ou indiretamente

# Simplificações em GLC

- Eliminação de $\varepsilon$-Produções
  - se $\varepsilon \in L$, a única produção utilizando $\varepsilon$ deve ser $S ::= \varepsilon$. Todas as demais podem ser eliminadas
  - uma GLC sem $\varepsilon$-Produções é dita $\varepsilon$-Livre
  - o método para eliminar $\varepsilon$-produções consiste em determinar para cada não-terminal $A$ em $N$, se $A \stackrel{*}{\Rightarrow} \varepsilon$
  - Se isto for verdade, se diz que a variável $A$ é anulável. Pode-se assim substituir cada produção da forma

$$B ::= X_1 X_2 ... X_n$$

    por todas as produções formadas pela retirada de uma ou mais variáveis de $X_i$ anuláveis

# Simplificações em GLC

- Eliminação de $\varepsilon$-não-terminais
  Entrada GLC $G = (N, T, P, S)$
  Saída GLC $G = (N', T, P', S')$$\varepsilon$-livre

  Construa o Conjunto $E$ (conjunto dos $\varepsilon$-não-terminais)
  $P' := \{p \mid p \in P$ e $p$ não é $\varepsilon$-produção$\}$
  Repita

  1. Se $P'$ tem uma produção da forma $A ::= \alpha B \beta$ tal que $B \in E$, $\alpha\beta \in (N \cup T)^*$ e $\alpha\beta \neq \varepsilon$, então inclua a produção $A ::= \alpha\beta$ em P'

  Até que nenhuma nova produção possa ser adicionada a $P'$
  Se $S \in E$ então
      Adicione a $P'$ as produções $S' ::= S|\varepsilon$
      $N' := N \cup \{S'\}$
  Senão $S' := S$ e $N' := N$

# Simplificações em GLC

- Identificação dos $\varepsilon$-não-terminais
  Entrada GLC $G = (N, T, P, S)$
  Saída conjunto E dos $\varepsilon$-não-terminais

  $E := \{\varepsilon\}$
  Repita

  1. $Q := \{X \mid X \in N$ e $X \notin E$ e existe pelo menos uma produção $X ::= Y_1Y_2...Y_n$ tal que $Y_1, Y_2, ..., Y_n \in E\}$
  2. $E := E \cup Q$

  Até $Q = \emptyset$

# Simplificações em GLC

- Eliminação de $\varepsilon$-Produções
  - raciocínio similar ao de eliminar transições $\varepsilon$ em um autômato
    - Se há uma produção na forma $A \rightarrow \varepsilon \mid aA$ construa novas produções, para aquelas onde $A$ aparece, omitindo o $A$.

# Simplificações em GLC

- Eliminação de $\varepsilon$-Produções
  - Exemplo:

$$
\begin{array}{l|l}
S ::= AB \mid Sc & \quad S' ::= S \mid \varepsilon \\
A ::= aA \mid \varepsilon & \quad S ::= AB \mid A \mid B \mid Sc \mid c \\
B ::= bB \mid \varepsilon & \quad A ::= aA \mid a \\
 & \quad B ::= bB \mid b
\end{array}
$$

# Simplificações em GLC

- É possível simplificar uma GLC sem reduzir o seu poder expressivo. Se L é uma LLC não vazia, então L pode ser gerada por uma GLC G com as seguintes propriedades:
  - Cada não-terminal e cada terminal de $G$ aparecem na derivação de alguma palavra de $L$ (não há símbolos inúteis)
  - Se $\varepsilon \notin L$ então não há necessidade de produções da forma $A ::= \varepsilon$ ($\varepsilon$-Livre)
  - **Não há produções da forma $A ::= B$ onde $A$ e $B$ são não-terminais (não há produções unitárias)**
  - Não há produções "não determinísticas" na gramática (Fatoração)
  - Não há produções recursivas, ou seja um não terminal não deriva a si próprio direta ou indiretamente

# Simplificações em GLC

- Eliminação de Produções Unitárias
  - Produção da forma $A ::= \alpha$ onde $\alpha$ é um não terminal
  - Se $A ::= A$ a produção é chamada de produção circular
    - Esse tipo de produção pode ser removida diretamente sem afetar a capacidade de geração da Gramática
  - O algoritmo para eliminar produções unitárias assume que a gramática de entrada não possui produções circulares

# Simplificações em GLC

- Eliminação de Produções Unitárias
  Entrada GLC $G = (N, T, P, S)$ sem produções circulares
  Saída GLC $G' = (N, T, P', S)$

  Para todo $A \in N$ faça:

  1. $N_A := \{B \mid A \overset{*}{\Rightarrow} B$ com $B \in N\} \cup A$

  $P' := \emptyset$
  Para toda produção $B ::= \alpha \in P$ faça:

  1. Se $B ::= \alpha$ não é produção unitária então
     $$P' := P' \cup \{A ::= \alpha \mid B \in N_A\}$$

# Simplificações em GLC

- Eliminação de Produções Unitárias
  - a aplicação deste algoritmo pode gerar uma GLC com símbolos inalcançáveis, uma vez que o símbolo não terminal é substituído por suas produções
    - Se o símbolo aparece somente em produções unitárias, após a execução do algoritmo, ele irá desaparecer do lado direito das produções, tornando-se inalcançável

# Simplificações em GLC

- É possível simplificar uma GLC sem reduzir o seu poder expressivo. Se L é uma LLC não vazia, então L pode ser gerada por uma GLC G com as seguintes propriedades:
  - Cada não-terminal e cada terminal de $G$ aparecem na derivação de alguma palavra de $L$ (não há símbolos inúteis)
  - Se $\varepsilon \notin L$ então não há necessidade de produções da forma $A ::= \varepsilon$ ($\varepsilon$-Livre)
  - Não há produções da forma $A ::= B$ onde $A$ e $B$ são não-terminais (não há produções unitárias)
  - **Não há produções "não determinísticas" na gramática (Fatoração)**
  - Não há produções recursivas, ou seja um não terminal não deriva a si próprio direta ou indiretamente

# Simplificações em GLC

- Fatoração
  - permite eliminar a indecisão sobre qual procução aplicar quando duas ou mais produções iniciam com a mesma forma sentencial
  - uma GLC pode ser não determinística direta ou indiretamente
    - Exemplos:

| | |
|---|---|
| $S ::= aSB \mid aSA$ | $S ::= AD \mid BC$ |
| $A ::= a$ | $A ::= aC \mid cC$ |
| $B ::= b$ | $B ::= aB \mid dD$ |
| | $C ::= eC \mid eA$ |
| | $D ::= fD \mid AB$ |

# Simplificações em GLC

- Fatoração
  - Para fatorar uma GLC deve-se alterar as produções envolvidas no não determinismo da seguinte forma:
    - As produções com não-determinismo direto da forma

$$A ::= \alpha\beta|\alpha\gamma$$

      serão substituídas por:

$$A ::= \alpha A'$$
$$A' ::= \beta|\gamma$$

# Simplificações em GLC

- Fatoração
  - O não-determinismo indireto é retirado transformando-o em não-determinismo direto (através de derivações sucessivas) e posteriormente eliminando-o
    - derivações sucessivas: substituir os não-terminais por suas produções

| | |
|---|---|
| $S ::= AC \mid BC$ | $S ::= aDC \mid cCC \mid aBC \mid dDC$ |
| $A ::= aD \mid cC$ | $A ::= aD \mid cC$ |
| $B ::= aB \mid dD$ | $B ::= aB \mid dD$ |
| $C ::= eC \mid eA$ | $C ::= eC \mid eA$ |
| $D ::= fD \mid CB$ | $D ::= fD \mid CB$ |

# Simplificações em GLC

- Gramática recusiva à esquerda
  - Um não-terminal $A$ em uma GLC é recursivo se $A ::= \alpha A \beta$ para algum $\alpha$ e $\beta \in (N \cup T)^*$
  - Se $\alpha$ é $\varepsilon$ então $A$ é recursivo à esquerda
  - A recursividade pode ser direta ou indireta
  - Exemplo:

$$S ::= Sb \mid Bc \mid Ab$$
$$A ::= Sc \mid ab$$
$$B ::= Scd \mid Bba \mid b$$

# Simplificações em GLC

- Para eliminar as recursões diretas à esquerda nas produções:

$$A ::= A\alpha_1 \mid A\alpha_2 \mid ... \mid A\alpha_n \mid \beta_1 \mid ... \mid \beta_m$$

deve-se substituir estas produções pelas seguintes:

$$A ::= \beta_1 A' \mid \beta_2 A' \mid ... \mid \beta_m A'$$
$$A' ::= \alpha_1 A' \mid \alpha_2 A' \mid ... \mid \alpha_n A' \mid \varepsilon$$

onde $A'$ é um novo não-terminal

# Simplificações em GLC

- Exemplo:

$$S ::= Sa|b$$

$$S ::= bS'$$
$$S' ::= aS' \mid \varepsilon$$

# Simplificações em GLC

- Para eliminar recursões indiretas, aplica o seguinte algoritmo, em uma gramática sem ciclos e sem $\varepsilon$-produções
  Entrada GLC $G = (N, T, P, S)$
  Saída GLC $G' = (N, T, P', S)$

  Coloque os não-terminais de $N$ em alguma ordem $A_1, A_2, \cdots, A_n$
  Para i := 1 até n faça

  1. Para j:= 1 até i-1 faça
     - Se $A_i ::= A_j\alpha \in P$ então
       Remova $A_i ::= A_j$ de $P$
       Se $A_j ::= \beta \in P$ então
       $P' = P' \cup \{A_i ::= \beta\alpha\}$
  2. Elimine as recursões diretas das produções de $P'$ com lado esquerdo $A_i$

# Simplificações em GLC

- Exemplo:

$$\begin{aligned} S &::= Aa \mid b \\ A &::= Ac \mid Sd \mid a \end{aligned} \quad \overset{Indireto}{\Longrightarrow} \quad \begin{aligned} S &::= Aa \mid b \\ A &::= Ac \mid Aad \mid bd \mid a \end{aligned}$$

$$\overset{Direto}{\Longrightarrow} \quad \begin{aligned} S &::= Aa \mid b \\ A &::= bdA' \mid aA' \\ A' &::= cA' \mid adA' \mid \varepsilon \end{aligned}$$

# Simplificações em GLC

- Exemplo:

$$\begin{array}{l|l} & S ::= AaS' \\ S ::= Aa \mid Sb & S' ::= bS' \mid \varepsilon \\ A ::= Sc \mid d & A ::= AaS'c \mid d \end{array}$$

$$\begin{aligned} S &::= AaS' \\ S' &::= bS' \mid \varepsilon \\ A &::= dA' \\ A' &::= aS'cA' \mid \varepsilon \end{aligned}$$