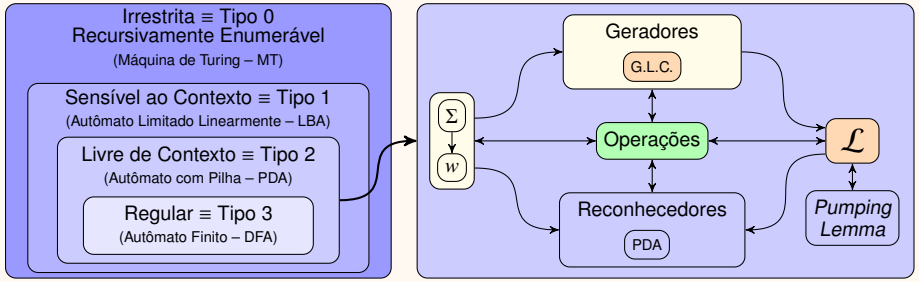
Linguagens Formais e Autômatos

Humberto Longo

Instituto de Informática
Universidade Federal de Goiás

Bacharelado em Ciência da Computação, 2021/1

Roteiro



Transformações em GLC's

- Objetivo das transformações é tornar uma GLC mais simples ou prepará-la para alguma aplicação posterior.
- Qualquer que seja a transformação efetuada em uma gramática, a linguagem gerada deve ser sempre a mesma.
- Transformações:
  1. ‣ Eliminação de recursão na variável inicial.
  2. ‣ Contração.
  3. ‣ Eliminação de derivações vazias.
  4. ‣ Eliminação de símbolos inúteis.
  5. ‣ Eliminação de regras de derivação simples (unitárias).
  6. ‣ Eliminação de recursão à esquerda (e à direita).
  7. ‣ Fatoração.

Eliminação de recursão na variável inicial

Lema 1.44

- *Seja $G = (V, \Sigma, P, S)$ uma gramática livre de contexto. Existe uma gramática $G' = (V', \Sigma, P', S')$, que satisfaz:*
  1. $\mathcal{L}(G) = \mathcal{L}(G')$,
  2. *As regras de derivação em P' são da forma $A \to w$, onde $A \in V'$ e $w \in ((V' - \{S'\}) \cup \Sigma)^*$.*

Demonstração.

- Se a variável S não ocorre no lado direito de nenhuma regra de derivação, então $G = G'$.
- Se S é uma variável recursiva, então $G' = (V \cup \{S'\}, \Sigma, P \cup \{S' \to S\}, S')$.

□

Eliminação de recursão na variável inicial

Lema 1.44

- *Seja $G = (V, \Sigma, P, S)$ uma gramática livre de contexto. Existe uma gramática $G' = (V', \Sigma, P', S')$, que satisfaz:*
  1. $\mathcal{L}(G) = \mathcal{L}(G')$,
  2. *As regras de derivação em P' são da forma $A \to w$, onde $A \in V'$ e $w \in ((V' - \{S'\}) \cup \Sigma)^*$.*

Demonstração.

- $\mathcal{L}(G) = \mathcal{L}(G')$
  - $S \overset{*}{\underset{G}{\Longrightarrow}} u \equiv S' \underset{G'}{\Longrightarrow} S \overset{*}{\underset{G'}{\Longrightarrow}} u.$

□

Eliminação de recursão na variável inicial

Exemplo 1.45

$$G: \quad P = \begin{cases} S \to aS \mid AB \mid AC, \\ A \to aA \mid \varepsilon, \\ B \to bB \mid bS, \\ C \to cC \mid \varepsilon \end{cases} \qquad G': \quad P' = \begin{cases} S' \to S, \\ S \to aS \mid AB \mid AC, \\ A \to aA \mid \varepsilon, \\ B \to bB \mid bS, \\ C \to cC \mid \varepsilon \end{cases}$$

▸ Transformações em GLC's

Contração

Lema 1.46

- *Seja $G = (V, \Sigma, P, S)$ uma GLC. Se $A \overset{*}{\underset{G}{\Longrightarrow}} w$, então a gramática $G' = (V, \Sigma, P \cup \{A \to w\}, S)$ é equivalente a G ($\mathcal{L}(G) = \mathcal{L}(G')$).*

Demonstração.

$\Rightarrow$ $\mathcal{L}(G) \subseteq \mathcal{L}(G')$ uma vez que toda regra de derivação de G também pertence à G'.

$\Leftarrow$ $\mathcal{L}(G') \subseteq \mathcal{L}(G)$, pois a aplicação da regra $A \to w$ em uma derivação em G' pode ser simulada em G pela derivação $A \overset{*}{\underset{G}{\Longrightarrow}} w$.

□

▸ Transformações em GLC's

Eliminação de derivações vazias

Definição 1.47 (ε-regra)

- Regra de derivação cujo lado direito contém somente a cadeia vazia, ou seja, $A \to \varepsilon$.

Definição 1.48 (ε-Variáveis)

- Conjunto de variáveis que derivam, direta ou indiretamente, a cadeia vazia: $V_\varepsilon = \{A \in V \mid A \overset{*}{\Longrightarrow} \varepsilon\}$.

Definição 1.49 (GLC ε-livre)

- GLC que não possui ε-regras ou possui $S \to \varepsilon$ como uma única ε-regra, onde S é o símbolo inicial da gramática e S não aparece do lado direito de nenhuma regra de derivação.

Eliminação de derivações vazias

Exemplo 1.50

- $G = (\{S, B\}, \{a, b\}, P, S)$, onde $P = \begin{Bmatrix} S \to SaB \mid aB, \\ B \to bB \mid \varepsilon \end{Bmatrix}$
- $\mathcal{L}(G) = (a^+b^*)^+$.
- Derivação à esquerda de $aaaa$ gera quatro B's, os quais são removidos com a regra $B \to \varepsilon$.
- Gramática equivalente sem ε-regras:
  - $G_1 = (\{S, B\}, \{a, b\}, P, S)$, onde $P = \begin{Bmatrix} S \to SaB \mid Sa \mid aB \mid a, \\ B \to bB \mid b \end{Bmatrix}$

Eliminação de derivações vazias

Algoritmo 1: Busca ε-variáveis

Entrada: $GLC\ (V, \Sigma, P, S)$.
Saída: Conjunto de ε-variáveis.

```
1 V_ε ← {A | (A → ε) ∈ P};
2 repita
3     AUX ← V_ε;
4     para cada (A ∈ V) faça
5         se (∃ A → w e w ∈ AUX*) então
6             V_ε ← V_ε ∪ {A};
7 até (AUX = V_ε);
8 retorna (V_ε);
```

Eliminação de derivações vazias

Exemplo 1.51

- $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$, onde $P = \begin{Bmatrix} S \to ACA, \\ A \to aAa \mid B \mid C, \\ B \to bB \mid b, \\ C \to cC \mid \varepsilon \end{Bmatrix}$

Iteração	V_ε	AUX
0	$\{C\}$	
1	$\{A, C\}$	$\{C\}$
2	$\{S, A, C\}$	$\{A, C\}$
3	$\{S, A, C\}$	$\{S, A, C\}$

Eliminação de derivações vazias

Lema 1.52

- *Dada uma gramática $G = (V, \Sigma, P, S)$, o Algoritmo 1 gera o conjunto de ε-variáveis de G.*

Demonstração.

1. Toda variável em V_ε deriva a cadeia vazia:
   - Indução no número de iterações do algoritmo.
   - Hipótese Indutiva:
     Após n iterações todas as variáveis em V_ε são ε-variáveis.
   - Passo Indutivo:
     Provar que qualquer variável adicionada no passo $n + 1$ também é uma ε-variável.

□

Eliminação de derivações vazias

Lema 1.52

- *Dada uma gramática $G = (V, \Sigma, P, S)$, o Algoritmo 1 gera o conjunto de ε-variáveis de G.*

Demonstração.

- Passo Indutivo:
  Se A é essa variável, então existe a regra de derivação $A \to A_1 A_2 \dots A_k$, em que cada $A_i \in AUX$ na iteração $n+1$.

  Por hipótese de indução, $A_i \overset{*}{\Longrightarrow} \varepsilon$ para $i = 1, 2, \dots, k$. Portanto:

$$A \Longrightarrow A_1 A_2 \dots A_k \overset{*}{\Longrightarrow} A_2 \dots A_k \overset{*}{\Longrightarrow} A_3 \dots A_k \overset{*}{\Longrightarrow} A_k \overset{*}{\Longrightarrow} \varepsilon.$$

□

Eliminação de derivações vazias

Lema 1.52

- *Dada uma gramática $G = (V, \Sigma, P, S)$, o Algoritmo 1 gera o conjunto de ε-variáveis de G.*

Demonstração.

2. Toda ε-variável é inserida no conjunto V_ε:
   - Indução no comprimento da derivação mínima de ε a partir de uma variável A.
   - Base:
     Se $A \overset{1}{\Longrightarrow} \varepsilon$, então A é inserido em V_ε no passo 1.
   - Hipótese Indutiva:
     Toda variável, cuja derivação mínima de ε é de comprimento máximo n, é inserida no conjunto V_ε até a iteração n.

□

Eliminação de derivações vazias

Lema 1.52

- *Dada uma gramática $G = (V, \Sigma, P, S)$, o Algoritmo 1 gera o conjunto de ε-variáveis de G.*

Demonstração.

- Passo Indutivo:
  Seja A uma variável que deriva ε em $n+1$ passos:

$$A \Longrightarrow A_1 A_2 \dots A_k \overset{n}{\Longrightarrow} \varepsilon.$$

  Por H.I., cada A_i deriva ε com derivação mínima de comprimento n ou menor e é inserida em V_ε antes da iteração $n+1$.

□

Eliminação de derivações vazias

Lema 1.52

- *Dada uma gramática $G = (V, \Sigma, P, S)$, o Algoritmo 1 gera o conjunto de ε-variáveis de G.*

Demonstração.

- Passo Indutivo:
  Se $m \leqslant n$ é a primeira iteração na qual todos os A_i's já foram inseridos em V_ε, então na iteração $m+1$ a regra $A \to A_1 A_2 \dots A_k$ força a variável A a ser inserida em V_ε.

□

Eliminação de derivações vazias

Definição 1.53

- Uma gramática sem ε-variáveis é não contraível, já que nenhuma regra de derivação pode diminuir o tamanho de qualquer forma sentencial.
- Se $G = (V, \Sigma, P, S)$ é uma GLC e $\varepsilon \in \mathcal{L}(G)$, então não existe uma gramática não contraível equivalente.
- Se $S \to \varepsilon$ é única ε-regra de G, então todas as derivações em G (com exceção de $S \Rightarrow \varepsilon$) são não contraíveis.
  - Gramática essencialmente não contraível.

Eliminação de derivações vazias

Exemplo 1.54

- $G_1: \begin{Bmatrix} S \to ASa, \\ A \to aA \mid \varepsilon \end{Bmatrix} \equiv G_2: \begin{Bmatrix} S \to ASa \mid Sa, \\ A \to aA \mid a \end{Bmatrix}$
- $G_3: \begin{Bmatrix} S \to ASAa, \\ A \to aA \mid \varepsilon \end{Bmatrix} \equiv G_4: \begin{Bmatrix} S \to ASAa \mid SAa \mid ASa \mid Sa, \\ A \to aA \mid a \end{Bmatrix}$

Eliminação de derivações vazias

Teorema 1.55

- *Seja $G = (V, \Sigma, P, S)$ uma GLC. Existe um algoritmo para construir uma gramática $G_L = (V_L, \Sigma, P_L, S_L)$ que satisfaz:*
  1. $\mathcal{L}(G) = \mathcal{L}(G_L)$,
  2. *S_L não é variável recursiva, e*
  3. *$(A \to \varepsilon) \in P$ se, e somente se, $\varepsilon \in \mathcal{L}(G)$ e $A = S_L$.*

Eliminação de derivações vazias

Demonstração.

2. S_L não é variável recursiva.
   - É suficiente usar a técnica apresentada no Lema 1.44. $V_L = V$ ou $V_L = V \cup \{S_L\}$ (se necessário mudar a variável inicial).
3. $(A \to \varepsilon) \in P$ se, e somente se, $\varepsilon \in \mathcal{L}(G)$ e $A = S_L$.
   - Regras de derivação de G_L:
     - 3.1 Se $\varepsilon \in \mathcal{L}(G)$ então $(S_L \to \varepsilon) \in P_L$.
     - 3.2 Se $(A \to w) \in P$ e $w = w_1 A_1 w_2 A_2 \ldots w_k A_k w_{k+1}$, onde $A_1, A_2, \ldots, A_k$ são ε-variáveis, então $(A \to w_1 w_2 \ldots w_k w_{k+1}) \in P_L$.
     - 3.3 $(A \to \varepsilon) \in P_L$ somente se $\varepsilon \in \mathcal{L}(G)$ e $A = S_L$.

□

Eliminação de derivações vazias

Demonstração.

1. $\mathcal{L}(G_L) \subseteq \mathcal{L}(G)$.
   - As derivações em G_L usam as regras de G e aquelas criadas pela condição 3 do teorema, as quais são deriváveis em G.
2. $\mathcal{L}(G) \subseteq \mathcal{L}(G_L)$.
   - Mostrar que toda cadeia não vazia, derivável em G a partir de uma variável A, também é derivável em G_L a partir de A.
   - Se $A \underset{G}{\overset{n}{\Longrightarrow}} w$, com $w \in \Sigma^+$, então $A \underset{G_L}{\overset{*}{\Longrightarrow}} w$.
     - Indução em n.

□

Eliminação de derivações vazias

Demonstração.

Se $A \underset{G}{\overset{n}{\Longrightarrow}} w$, com $w \in \Sigma^+$, então $A \underset{G_L}{\overset{*}{\Longrightarrow}} w$.

- **Base:**
  Se $n = 1$, então $(A \to w) \in P$ e, como $w \neq \varepsilon$, $(A \to w) \in P_L$.
- **Hipótese Indutiva:**
  Toda cadeia de terminais derivável a partir de A em G, em no máximo n passos, também é derivável em G_L.

□

Eliminação de derivações vazias

Demonstração.

Se $A \underset{G}{\overset{n}{\Longrightarrow}} w$, com $w \in \Sigma^+$, então $A \underset{G_L}{\overset{*}{\Longrightarrow}} w$.

- **Passo Indutivo:**
  Seja $A \underset{G}{\overset{n+1}{\Longrightarrow}} w$ uma derivação de uma cadeia de terminais. Então:

$$A \Longrightarrow w_1 A_1 w_2 A_2 \ldots w_k A_k w_{k+1} \underset{G}{\overset{n}{\Longrightarrow}} w,$$

onde $A_i \in V$ e $w_i \in \Sigma^*$. Logo, w pode ser reescrito como:

$$w = w_1 p_1 w_2 p_2 \ldots w_k p_k w_{k+1},$$

onde A_i deriva p_i em no máximo n passos.

□

Eliminação de derivações vazias

Demonstração.

Se $A \underset{G}{\overset{n}{\Longrightarrow}} w$, com $w \in \Sigma^+$, então $A \underset{G_L}{\overset{*}{\Longrightarrow}} w$.

- **Passo Indutivo:**
  Por H.I. $A_i \underset{G_L}{\overset{*}{\Longrightarrow}} p_i$, para cada $p_i \in \Sigma^+$. Se $p_i = \varepsilon$, então algum A_j é uma ε-variável e uma nova regra de derivação é gerada a partir de

$$A \to w_1 A_1 w_2 A_2 \ldots w_k A_k w_{k+1},$$

na qual cada variável que deriva a cadeia vazia é excluída.

Para se derivar w em G_L, primeiro aplica-se essa nova regra e então deriva-se cada p_i usando-se as derivações fornecidas pela H.I.

□

Eliminação de derivações vazias

Exemplo 1.56

- $G: \begin{cases} S \to ACA, \\ A \to aAa \mid B \mid C, \\ B \to bB \mid b, \\ C \to cC \mid \varepsilon \end{cases}$ $\quad$ $G_L: \begin{cases} S \to ACA \mid CA \mid AA \mid AC \mid A \mid C \mid \varepsilon, \\ A \to aAa \mid aa \mid B \mid C, \\ B \to bB \mid b, \\ C \to cC \mid c \end{cases}$
  - $V_\varepsilon = \{S, A, C\}$
  - Derivação da cadeia vazia:
    $G: \ S \Rightarrow ACA \Rightarrow CCA \Rightarrow CA \Rightarrow A \Rightarrow C \Rightarrow \varepsilon$
    $G_L: \ S \Rightarrow \varepsilon$
  - Derivação da cadeia aba:
    $G: \ S \Rightarrow ACA \Rightarrow aAaCA \Rightarrow aBaCA \Rightarrow abaCA \Rightarrow abaA \Rightarrow abaC \Rightarrow aba$
    $G_L: \ S \Rightarrow A \Rightarrow aAa \Rightarrow aBa \Rightarrow aba$

Eliminação de derivações vazias

Exemplo 1.57

- $G: \begin{cases} S \to ABC, \\ A \to aA \mid \varepsilon, \\ B \to bB \mid \varepsilon, \\ C \to cC \mid \varepsilon \end{cases}$
  - $\mathcal{L}(G) = a^*b^*c^*$
  - $V_\varepsilon = \{S, A, B, C\}$
  - $G_L: \begin{cases} S \to ABC \mid AB \mid BC \mid AC \mid A \mid B \mid C \mid \varepsilon, \\ A \to aA \mid a, \\ B \to bB \mid b, \\ C \to cC \mid c \end{cases}$

Transformações em GLC's

Eliminação de derivações simples

Definição 1.58

- Regras de derivação simples são da forma $A \to B$, onde $A, B \in V$.

Observações

- A aplicação de uma regra $A \to B$ não aumenta a forma sentencial derivada ou acrescenta terminais à mesma.
- A remoção de regras simples requer a adição de novas regras que permitam a geração das mesmas cadeias.
- Uma regra simples apenas renomeia uma variável.

Eliminação de derivações simples

Exemplo 1.59

- Considere as regras $\begin{cases} A \to aA \mid a \mid B, \\ B \to bB \mid b \mid C \end{cases}$
- A regra $A \to B$ indica que qualquer cadeia derivável a partir de B também é derivável a partir de A.
- Eliminação de $A \to B$: acrescentar $A \to w$ para cada regra $B \to w$:

$$\begin{cases} A \to aA \mid a \mid bB \mid b \mid C, \\ B \to bB \mid b \mid C \end{cases}$$

Eliminação de derivações simples

Definição 1.60

- Uma derivação $A \overset{*}{\Longrightarrow} C$ constituída apenas de regras de derivação simples é chamada de cadeia de derivação.

Dúvida?

- Como determinar as variáveis que podem ser derivadas, com uma cadeia de derivação, a partir de uma dada variável?

Eliminação de derivações simples

Algoritmo 2: Cadeia de derivação

Entrada: $GLC\ (V, \Sigma, P, S)$ essencialmente não contraível e $A \in V$.
Saída: Conjunto de variáveis de cadeias de derivação de A.

```
1  CADEIA(A) ← {A};
2  AUX ← ∅;
3  repita
4  |   NOVAS ← CADEIA(A) − AUX;
5  |   AUX ← CADEIA(A);
6  |   para cada (B ∈ NOVAS) faça
7  |   |   para cada (B → C) faça
8  |   |   |   CADEIA(A) ← CADEIA(A) ∪ {C};
9  até (AUX = CADEIA(A));
10 retorna (CADEIA(A));
```

Eliminação de derivações simples

Lema 1.61

- *Dada uma GLC essencialmente não contraível, o Algoritmo 2 gera o conjunto de variáveis que podem ser derivadas, com cadeias de derivação, a partir de uma dada variável.*

Demonstração.

Exercício □

Eliminação de derivações simples

Teorema 1.62

- *Dada a $GLC\ G = (V, \Sigma, P, S)$ essencialmente não contraível, existe um algoritmo que constrói a $GLC\ G_C = (V_C, \Sigma, P_C, S)$ tal que:*
  1. $\mathcal{L}(G_C) = \mathcal{L}(G)$,
  2. *G_C não contém regras de derivação simples.*

Demonstração.

2. G_C não contém regras de derivação simples.
   - $(A \to w) \in P_C$ se existe $B \in V$ e w tais que:
     - 2.1 $B \in CADEIA(A)$,
     - 2.2 $(B \to w) \in P$,
     - 2.3 $w \notin V$.

□

Eliminação de derivações simples

Teorema 1.62

- *Dada a GLC $G = (V, \Sigma, P, S)$ essencialmente não contraível, existe um algoritmo que constrói a GLC $G_C = (V_C, \Sigma, P_C, S)$ tal que:*
  1. $\mathcal{L}(G_C) = \mathcal{L}(G)$,
  2. *G_C não contém regras de derivação simples.*

Demonstração.

2. G' não contém regras de derivação simples.
   - As regras de derivação de A em G_C são construídas a partir dos conjuntos $CADEIA(A)$ e P.
   - A terceira condição garante que P_C não contém regras de derivação simples.

□

Eliminação de derivações simples

Teorema 1.62

- *Dada a GLC $G = (V, \Sigma, P, S)$ essencialmente não contraível, existe um algoritmo que constrói a GLC $G_C = (V_C, \Sigma, P_C, S)$ tal que:*
  1. $\mathcal{L}(G_C) = \mathcal{L}(G)$,
  2. *G_C não contém regras de derivação simples.*

Demonstração.

1. $\mathcal{L}(G_C) \subseteq \mathcal{L}(G)$:
   - O Lema 1.46 garante que toda cadeia derivável em G_C também é derivável em G.

□

Eliminação de derivações simples

Teorema 1.62

- *Dada a GLC $G = (V, \Sigma, P, S)$ essencialmente não contraível, existe um algoritmo que constrói a GLC $G_C = (V_C, \Sigma, P_C, S)$ tal que:*
  1. $\mathcal{L}(G_C) = \mathcal{L}(G)$,
  2. *G_C não contém regras de derivação simples.*

Demonstração.

2. $\mathcal{L}(G) \subseteq \mathcal{L}(G_C)$:
   - Seja derivação $S \underset{G}{\overset{*}{\Longrightarrow}} uAv \underset{G}{\overset{*}{\Longrightarrow}} uBv \underset{G}{\Longrightarrow} upv \underset{G}{\overset{*}{\Longrightarrow}} w$, onde $w \in \mathcal{L}(G)$ e $A \underset{G}{\overset{*}{\Longrightarrow}} B$ é uma seqüência maximal de regras de derivação simples.

□

Eliminação de derivações simples

Teorema 1.62

- *Dada a GLC $G = (V, \Sigma, P, S)$ essencialmente não contraível, existe um algoritmo que constrói a GLC $G_C = (V_C, \Sigma, P_C, S)$ tal que:*
  1. $\mathcal{L}(G_C) = \mathcal{L}(G)$,
  2. *G_C não contém regras de derivação simples.*

Demonstração.

1. $\mathcal{L}(G) \subseteq \mathcal{L}(G_C)$:
   - A regra $A \to p$ pode substituir a cadeia $A \underset{G}{\overset{*}{\Longrightarrow}} B$.
   - O uso sucessivo desta técnica produz uma derivação válida de w em G_C.

□

Eliminação de derivações simples

Exemplo 1.63

- $G: \begin{cases} S \to aSb \mid A, \\ A \to aA \mid B, \\ B \to bBc \mid bc \end{cases}$

Variável	*Cadeia*
S	$\{S, A, B\}$
A	$\{A, B\}$
B	$\{B\}$

- $G_C: \begin{cases} S \to aSb \mid aA \mid bBc \mid bc, \\ A \to aA \mid bBc \mid bc, \\ B \to bBc \mid bc \end{cases}$

Eliminação de derivações simples

Exemplo 1.64

- $G: \begin{cases} S \to ACA, \\ A \to aAa \mid B \mid C, \\ B \to bB \mid b, \\ C \to cC \mid \varepsilon \end{cases}$

- $G_L: \begin{cases} S \to ACA \mid CA \mid AA \mid AC \mid A \mid C \mid \varepsilon, \\ A \to aAa \mid aa \mid B \mid C, \\ B \to bB \mid b, \\ C \to cC \mid c \end{cases}$

Variável	*Cadeia*
S	$\{S, A, C, B\}$
A	$\{A, B, C\}$
B	$\{B\}$
C	$\{C\}$

- $G_C: \begin{cases} S \to ACA \mid CA \mid AA \mid AC \mid aAa \mid aa \mid bB \mid b \mid cC \mid c \mid \varepsilon, \\ A \to aAa \mid aa \mid bB \mid b \mid cC \mid c, \\ B \to bB \mid b \\ C \to cC \mid c \end{cases}$

Eliminação de derivações simples

- A remoção de regras de derivação simples aumenta o número de regras na gramática, mas reduz o comprimento das derivações.
- A eliminação de regras de derivação simples de uma gramática essencialmente não contraível preserva esta propriedade.

Definição 1.65

- Dada uma $GLC\ (V, \Sigma, P, S)$ essencialmente não contraível e sem regras de derivação simples, então P é formado por regras do tipo:
  1. $S \to \varepsilon$
  2. $A \to a$
  3. $A \to w$,

  onde $w \in (V \cup \Sigma)^+$ é de comprimento pelo menos 2.

Transformações em *GLC*'s

Eliminação de símbolos inúteis

Definição 1.66

- Dada uma $GLC\ (V, \Sigma, P, S)$, um símbolo $x \in (V \cup \Sigma)$ é útil se existe uma derivação

$$S \overset{*}{\underset{G}{\Longrightarrow}} uxv \overset{*}{\underset{G}{\Longrightarrow}} w,$$

  onde $u, v \in (V \cup \Sigma)^*$ e $w \in \Sigma^*$.

- Um símbolo terminal é útil se ocorre em uma cadeia da linguagem de G.
- Uma variável é definida como útil se ocorre em uma derivação que começa na variável inicial e gera uma cadeia de terminais.
  - Variável deve ocorrer em uma forma sentencial da gramática.
  - Variável deve derivar uma cadeia de terminais.

Eliminação de símbolos inúteis

Exemplo 1.67

- $G_1 : \begin{cases} S \to aS \mid A, \\ A \to a \mid bB, \\ B \to b \mid dD, \\ C \to cC \mid c, \\ D \to dD \end{cases}$
- $\mathcal{L}(G) = a^*(a \cup bb)$

- $G_1 : \begin{cases} S \to AC \mid BS \mid B, \\ A \to aA \mid aF, \\ B \to CF \mid b, \\ C \to cC \mid D, \\ D \to aD \mid BD \mid C, \\ E \to aA \mid BSA, \\ F \to bB \mid b \end{cases}$
- $\mathcal{L}(G) = b^+$

Eliminação de símbolos inúteis

Definição 1.68 (Símbolo estéril)

- Não gera qualquer cadeia de terminais de uma sentença.

Definição 1.69 (Símbolo inalcançável)

- Não aparece em nenhuma forma sentencial da gramática.

Definição 1.70 (Simbolos alcançáveis)

- Símbolos deriváveis a partir do símbolo inicial da gramática.

Definição 1.71 (Um símbolo inútil)

- Símbolo estéril ou inalcançável.

Definição 1.72

- V_{Σ^+}: Conjunto de variáveis que derivam cadeias de terminais.

Eliminação de símbolos inúteis

Algoritmo 3: Busca V_{Σ^+}

Entrada: GLC (V, Σ, P, S).
Saída: Conjunto V_{Σ^+}.

```
1 V_Σ+ ← {A | (A → w) ∈ P, com w ∈ Σ*};
2 repita
3 |   AUX ← V_Σ+;
4 |   para cada (A ∈ V) faça
5 |   |   se (∃ A → w e w ∈ (AUX ∪ Σ)*) então
6 |   |   |   V_Σ+ ← V_Σ+ ∪ {A};
7 até (AUX = V_Σ+);
8 retorna (V_Σ+);
```

Eliminação de símbolos inúteis

Teorema 1.73

- *Dada uma GLC $G = (V, \Sigma, P, S)$, existe um algoritmo para construir uma GLC $G_T = (V_T, \Sigma_T, P_T, S)$ que satisfaz:*
  1. $\mathcal{L}(G_T) = \mathcal{L}(G)$,
  2. *Toda variável em V_T deriva uma cadeia de terminais em G_T.*

Demonstração.

- $V_T = V_{\Sigma^+}$.
- $P_T = \{A \to w \mid (A \to w) \in P,\ A \in V_{\Sigma^+} \text{ e } w \in (V_{\Sigma^+} \cup \Sigma)^*\}$.
- $\Sigma_T = \{a \in \Sigma \mid (A \to uav) \in P,\ A \in V_T \text{ e } u, v \in (V_T \cup \Sigma)^*\}$.

□

Eliminação de símbolos inúteis

Teorema 1.73

- *Dada uma GLC $G = (V, \Sigma, P, S)$, existe um algoritmo para construir uma GLC $G_T = (V_T, \Sigma_T, P_T, S)$ que satisfaz:*
  1. $\mathcal{L}(G_T) = \mathcal{L}(G)$,
  2. *Toda variável em V_T deriva uma cadeia de terminais em G_T.*

Demonstração.

- $\mathcal{L}(G_T) \subseteq \mathcal{L}(G)$:
  Como $P_T \subseteq P$, qualquer derivação em G_T também é derivação em G.

□

Eliminação de símbolos inúteis

Teorema 1.73

- *Dada uma GLC $G = (V, \Sigma, P, S)$, existe um algoritmo para construir uma GLC $G_T = (V_T, \Sigma_T, P_T, S)$ que satisfaz:*
  1. $\mathcal{L}(G_T) = \mathcal{L}(G)$,
  2. *Toda variável em V_T deriva uma cadeia de terminais em G_T.*

Demonstração.

- $\mathcal{L}(G) \subseteq \mathcal{L}(G_T)$:
  A remoção de regras de derivação que contêm variáveis em $V - V_{\Sigma^*}$ não afeta o conjunto de cadeias de terminais geradas.

□

Eliminação de símbolos inúteis

Teorema 1.73

- *Dada uma GLC $G = (V, \Sigma, P, S)$, existe um algoritmo para construir uma GLC $G_T = (V_T, \Sigma_T, P_T, S)$ que satisfaz:*
  1. $\mathcal{L}(G_T) = \mathcal{L}(G)$,
  2. *Toda variável em V_T deriva uma cadeia de terminais em G_T.*

Demonstração.

- $\mathcal{L}(G) \subseteq \mathcal{L}(G_T)$:
  Seja $S \overset{*}{\underset{G}{\Longrightarrow}} w$. Se esta derivação não é válida em G_T, então uma variável de $V - V_{\Sigma^+}$ ocorre numa forma sentencial intermediária, a qual não gera uma cadeia de terminais.

□

Eliminação de símbolos inúteis

Teorema 1.73

- *Dada uma GLC $G = (V, \Sigma, P, S)$, existe um algoritmo para construir uma GLC $G_T = (V_T, \Sigma_T, P_T, S)$ que satisfaz:*
  1. $\mathcal{L}(G_T) = \mathcal{L}(G)$,
  2. *Toda variável em V_T deriva uma cadeia de terminais em G_T.*

Demonstração.

- $\mathcal{L}(G) \subseteq \mathcal{L}(G_T)$:
  Logo, todas as regras de derivação usadas estão em P_T e $w \in \mathcal{L}(G)$.

□

Eliminação de símbolos inúteis

Exemplo 1.74

- $G_1 : \begin{cases} S \to AC \mid BS \mid B, \\ A \to aA \mid aF, \\ B \to CF \mid b, \\ C \to cC \mid D, \\ D \to aD \mid BD \mid C, \\ E \to aA \mid BSA, \\ F \to bB \mid b \end{cases}$

Iteração	V_{Σ^+}	AUX
0	$\{B, F\}$	
1	$\{B, F, A, S\}$	$\{B, F\}$
2	$\{B, F, A, S, E\}$	$\{B, F, A, S\}$
3	$\{B, F, A, S, E\}$	$\{B, F, A, S, E\}$

- $G_T : \begin{cases} S \to BS \mid B, \\ A \to aA \mid aF, \\ B \to b, \\ E \to aA \mid BSA, \\ F \to bB \mid b \end{cases}$

Eliminação de símbolos inúteis

Algoritmo 4: Busca variáveis alcançáveis

Entrada: $GLC\ (V, \Sigma, P, S)$.
Saída: Conjunto V_S.

```
1  V_S ← {S};
2  AUX ← ∅;
3  repita
4      NOVAS ← V_S − AUX;
5      AUX ← V_S;
6      para cada (A ∈ NOVAS) faça
7          para cada (A → w_1A_1w_2A_2...w_kA_kw_{k+1}) ∈ P faça
               // w_1, w_2, ..., w_k, w_{k+1} ∈ Σ* e A_1, A_2, ..., A_k ∈ V
8              V_S ← V_S ∪ {A_1, A_2, ..., A_k};
9  até (AUX = V_S);
10 retorna (V_S);
```

Eliminação de símbolos inúteis

Exemplo 1.75

- $G_1 : \begin{cases} S \to AC \mid BS \mid B, \\ A \to aA \mid aF, \\ B \to CF \mid b, \\ C \to cC \mid D, \\ D \to aD \mid BD \mid C, \\ E \to aA \mid BSA, \\ F \to bB \mid b \end{cases}$

- $G_T : \begin{cases} S \to BS \mid B, \\ A \to aA \mid aF, \\ B \to b, \\ E \to aA \mid BSA, \\ F \to bB \mid b \end{cases}$

Iteração	V_S	AUX	$NOVAS$
0	$\{S\}$	$\emptyset$	
1	$\{S, B\}$	$\{S\}$	$\{S\}$
2	$\{S, B\}$	$\{S, B\}$	$\{B\}$

- $G_U : \begin{cases} S \to BS \mid B, \\ B \to b \end{cases} \equiv \{S \to bS \mid b\}$

Eliminação de símbolos inúteis

Lema 1.76

- *Dada uma gramática $G = (V, \Sigma, P, S)$, o Algoritmo 4 gera o conjunto de variáveis alcançáveis a partir de S.*

Demonstração.

1. Toda variável em V_S é derivável a partir de S:
   - Indução no número de iterações do algoritmo.
   - Hipótese Indutiva:
     Após n iterações todas as variáveis em V_S são alcançáveis a partir de S.
   - Passo Indutivo:
     Provar que qualquer variável adicionada no passo $n + 1$ também é alcançável a partir de S.

□

Eliminação de símbolos inúteis

Lema 1.76

- *Dada uma gramática $G = (V, \Sigma, P, S)$, o Algoritmo 4 gera o conjunto de variáveis alcançáveis a partir de S.*

Demonstração.

1. Toda variável em V_S é derivável a partir de S:
   - Passo Indutivo:
     Se B é essa variável, então existe a regra de derivação $A \to uBv$, onde $A \in V_S$ após n iterações.

     Por hipótese de indução, $S \overset{*}{\Longrightarrow} xAy$. Portanto, $S \overset{*}{\Longrightarrow} xAy \Longrightarrow xuBvy$ e B é alcançável a partir de S.

□

Eliminação de símbolos inúteis

Lema 1.76

- *Dada uma gramática $G = (V, \Sigma, P, S)$, o Algoritmo 4 gera o conjunto de variáveis alcançáveis a partir de S.*

Demonstração.

2. Toda variável alcançável a partir de S é inserida em V_S:
   - Indução no comprimento da derivação a partir S.
   - Base:
     Para $n = 0$, S é inserido em V_S no passo 1.
   - Hipótese Indutiva:
     Toda variável derivável a partir de S numa derivação de comprimento máximo n é inserida no conjunto V_S.

□

Eliminação de símbolos inúteis

Lema 1.76

- *Dada uma gramática $G = (V, \Sigma, P, S)$, o Algoritmo 4 gera o conjunto de variáveis alcançáveis a partir de S.*

Demonstração.

2. Toda variável alcançável a partir de S é inserida em V_S:
   - Passo Indutivo:
     Seja a derivação $S \overset{n}{\Longrightarrow} xAy \Longrightarrow xuBvy$, onde no passo $n+1$ a regra aplicada é $A \to uBv$.

     Por H.I., A foi inserido em V_S na iteração n. Logo, B é inserido na iteração seguinte.

□

Eliminação de símbolos inúteis

Teorema 1.77

- *Dada uma GLC $G = (V, \Sigma, P, S)$, existe um algoritmo para construir uma GLG $G_U = (V_U, \Sigma_U, P_U, S)$ que satisfaz:*
  1. $\mathcal{L}(G_U) = \mathcal{L}(G)$,
  2. *A gramática G_U não contém símbolos inúteis.*

Demonstração.

- $V_U = V_S$
- $P_U = \{A \to w \mid (A \to w) \in P,\ A \in V_S \text{ e } w \in (V_S \cup \Sigma)^*\}$
- $\Sigma_U = \{a \in \Sigma \mid (A \to uav) \in P,\ A \in V_U \text{ e } u, v \in (V_U \cup \Sigma)^*\}$

□

Eliminação de símbolos inúteis

Teorema 1.77

- *Dada uma GLC $G = (V, \Sigma, P, S)$, existe um algoritmo para construir uma GLG $G_U = (V_U, \Sigma_U, P_U, S)$ que satisfaz:*
  1. $\mathcal{L}(G_U) = \mathcal{L}(G)$,
  2. *A gramática G_U não contém símbolos inúteis.*

Demonstração.

- $\mathcal{L}(G_U) = \mathcal{L}(G_T)$:
  Mostrar que toda cadeia derivável em G_T também é derivável em G_U (Se $w \in \mathcal{L}(G_T)$, toda variável que ocorre na derivação de w é alcançável e toda regra de derivação pertence à P_U).

□

Eliminação de símbolos inúteis

- A eliminação de símbolos inúteis deve ser feita em duas etapas:
  1. Remoção de variáveis que não geram cadeias de terminais, e
  2. Remoção de variáveis que não são deriváveis a partir do símbolo inicial.
- A inversão dessas etapas pode não remover todos os símbolos inúteis.

Eliminação de símbolos inúteis

Exemplo 1.78

- Seja a GLC $G: \left\{ \begin{array}{l} S \to a \mid AB, \\ A \to b \end{array} \right\}$
- Elimina variáveis que:
  1. não geram cadeias de terminais, $G_1: \left\{ \begin{array}{l} S \to a, \\ A \to b \end{array} \right\}$
  2. não são deriváveis a partir de S, $G_2: \{S \to a\}$
- Elimina variáveis que:
  1. não são deriváveis a partir de S, $G_1: \left\{ \begin{array}{l} S \to a \mid AB, \\ A \to b \end{array} \right\}$
  2. não geram cadeias de terminais, $G_2: \left\{ \begin{array}{l} S \to a, \\ A \to b \end{array} \right\}$

▸ Transformações em GLC's

Remoção de recursão direta à esquerda

Definição 1.79

- Uma gramática $G = (V, \Sigma, P, S)$ tem recursão à esquerda se existe $A \in V$ tal que $A \overset{+}{\Longrightarrow} Aw$, com $w \in (V \cup \Sigma)^*$.

Definição 1.80

- Uma gramática $G = (V, \Sigma, P, S)$ tem recursão à direita se existe $A \in V$ tal que $A \overset{+}{\Longrightarrow} wA$, com $w \in (V \cup \Sigma)^*$.

Definição 1.81

- A recursão é dita direta se a derivação for em um passo:
  - G tem recursão direta à esquerda se $A \to Av \in P$,
  - G tem recursão direta à direita se $A \to vA \in P$.

Remoção de recursão direta à esquerda

Lema 1.82

- *Se $G = (V, \Sigma, P, S)$ é uma GLC e $A \in V$ uma variável com recursão à esquerda, então existe um algoritmo para construir uma gramática equivalente $G' = (V', \Sigma, P', S')$, na qual a variável A não apresenta recursão direta à esquerda.*

Demonstração.

- Assuma que:
  1. S não é recursivo,
  2. A única ε-regra é $S \to \varepsilon$,
  3. $(A \to A) \notin P$.

□

Remoção de recursão direta à esquerda

Lema 1.82

- *Se $G = (V, \Sigma, P, S)$ é uma GLC e $A \in V$ uma variável com recursão à esquerda, então existe um algoritmo para construir uma gramática equivalente $G' = (V', \Sigma, P', S')$, na qual a variável A não apresenta recursão direta à esquerda.*

Demonstração.

- Dividir as regras de derivação de A em dois grupos:
  1. Recursivas: $A \to Au_1 \mid Au_2 \mid \cdots \mid Au_j$.
  2. Não recursivas: $A \to v_1 \mid v_2 \mid \cdots \mid v_k$.

□

Remoção de recursão direta à esquerda

Lema 1.82

- *Se $G = (V, \Sigma, P, S)$ é uma GLC e $A \in V$ uma variável com recursão à esquerda, então existe um algoritmo para construir uma gramática equivalente $G' = (V', \Sigma, P', S')$, na qual a variável A não apresenta recursão direta à esquerda.*

Demonstração.

- Derivação à esquerda é encerrada com uma regra $A \to v_i$.
- Toda cadeia derivada a partir de A começa com algum v_i.
- Novas regras para derivação a partir de A:
  1. Colocar um dos v_i's na extremidade esquerda da cadeia a ser derivada.
  2. Usar recursão à direita para gerar os u_i's.

□

Remoção de recursão direta à esquerda

Lema 1.82

- *Se $G = (V, \Sigma, P, S)$ é uma GLC e $A \in V$ uma variável com recursão à esquerda, então existe um algoritmo para construir uma gramática equivalente $G' = (V', \Sigma, P', S')$, na qual a variável A não apresenta recursão direta à esquerda.*

Demonstração.

1. v_i's na extremidade esquerda:

$$A \to v_1 \mid \cdots \mid v_k \mid v_1 Z \mid \cdots \mid v_k Z.$$

2. u_i's a direita:

$$Z \to u_1 Z \mid \cdots \mid u_j Z \mid u_1 \mid \cdots \mid u_j.$$

□

Remoção de recursão direta à esquerda

Exemplo 1.83

- $G: \{A \to Aa \mid Aab \mid bb \mid b\}$
- $\mathcal{L}(G) = (b \cup bb)(a \cup ab)^*$
- $G': \begin{Bmatrix} A \to bb \mid b \mid bbZ \mid bZ, \\ Z \to aZ \mid abZ \mid a \mid ab \end{Bmatrix}$
- As regras de derivação de Z geram as cadeias $(a \cup ab)^+$

Remoção de recursão direta à esquerda

Exemplo 1.84

1. $G: \{A \to Aa \mid b\}$
   - $\mathcal{L}(G) = ba^*$
   - $G': \begin{Bmatrix} A \to b \mid bZ, \\ Z \to aZ \mid a \end{Bmatrix}$
2. $G: \{A \to Aa \mid Ab \mid b \mid c\}$
   - $\mathcal{L}(G) = (b \cup c)(a \cup b)^*$
   - $G': \begin{Bmatrix} A \to b \mid c \mid bZ \mid cZ, \\ Z \to aZ \mid bZ \mid a \mid b \end{Bmatrix}$
3. $G: \begin{Bmatrix} A \to AB \mid BA \mid a, \\ B \to b \mid c \end{Bmatrix}$
   - $\mathcal{L}(G) = (b \cup c)^* a (b \cup c)^*$
   - $G': \begin{Bmatrix} A \to a \mid BAZ \mid aZ \mid BA, \\ Z \to BZ \mid B, \\ B \to b \mid c \end{Bmatrix}$

▸ Transformações em GLC's

Fatoração de GLC

- Uma GLC é fatorada se é determinística.
  - Não possui regras de derivação que iniciam com o mesmo símbolo terminal ou que iniciam com variáveis que gerem subcadeias que iniciam com o mesmo símbolo terminal.

Exemplo 1.85

- $G_1: \begin{Bmatrix} S \to aA \mid aB, \\ A \to aA \mid a, \\ B \to aB \mid b \end{Bmatrix}.$
- $G_2: \begin{Bmatrix} S \to A \mid B, \\ A \to aA \mid a, \\ B \to aB \mid b \end{Bmatrix}.$

Fatoração de GLC

Não determinismo direto

Original: $\{S \to \alpha\beta \mid \alpha\delta\}$.

Fatorada: $\{S \to \alpha A \mid A \to \beta \mid \delta\}$.

Não determinismo indireto

1. Aplicar outras transformações para reduzir ao não determinismo direto.
2. Remover o não determinismo direto.

Fatoração de GLC

Exemplo 1.86

- $G_1 = (\{S, A, B\}, \{a, b\}, P, S)$ e
$$P = \begin{cases} S \to aA \mid aB, \\ A \to aA \mid a, \\ B \to aB \mid b \end{cases}.$$
- $$P' = \begin{cases} S \to aX, \\ X \to A \mid B, \\ A \to aY, \\ Y \to A \mid \varepsilon, \\ B \to aB \mid b \end{cases}.$$

- $G_2 = (\{S, A\}, \{a, b\}, P, S)$ e
$$P = \begin{cases} S \to Ab \mid ab \mid baA, \\ A \to aab \mid b \end{cases}.$$
- $$P' = \begin{cases} S \to aabb \mid bb \mid ab \mid baA, \\ A \to aab \mid b \end{cases}.$$
- $$P'' = \begin{cases} S \to aX \mid bY, \\ X \to abb \mid b, \\ Y \to b \mid aA, \\ A \to aab \mid b \end{cases}.$$

Fatoração de GLC

Aplicação – construção de compiladores

- Se a gramática for não determinística, o processo de compilação deve verificar cada uma das possibilidades de derivação.
- Uso de retornos (*backtracking*) reduz a eficiência do processo de compilação.

▸ Transformações em *GLC*'s

Livros texto


R. P. Grimaldi
Discrete and Combinatorial Mathematics – An Applied Introduction.
Addison Wesley, 1994.


D. J. Velleman
How To Prove It – A Structured Approach.
Cambridge University Press, 1996.


J. E. Hopcroft; J. Ullman.
Introdução Ã Teoria de Autômatos, Linguagens e Computação.
Ed. Campus.


T. A. Sudkamp.
Languages and Machines – An Introduction to the Theory of Computer Science.
Addison Wesley Longman, Inc. 1998.


J. Carroll; D. Long.
Theory of Finite Automata – With an Introduction to Formal Languages.
Prentice-Hall, 1989.


M. Sipser.
Introduction to the Theory of Computation.
PWS Publishing Company, 1997.


H. R. Lewis; C. H. Papadimitriou
Elementos de Teoria da Computação.
Bookman, 2000.