LINGUAGENS FORMAIS: Teoria, Modelagem e Implementação

Material adicional, versão do dia 22 de março de 2020 às 08:35

1. Página 31, primeiro parágrafo da Seção 1.1:
   onde se lê:
   "Um **conjunto** é uma coleção de símbolos, também denominados átomos ou elementos, em que ..."
   leia-se:
   "Um **conjunto** é uma coleção de elementos em que ..."

2. Página 31, Exemplo 1.1:
   onde se lê:
   "A inclusão do símbolo ..."
   leia-se:
   "A inclusão do elemento ..."

3. Página 31:
   eliminar o parágrafo "Um **símbolo** ... arbitrário deles."

4. Página 31, Exemplo 1.2:
   eliminar

5. Página 32, imediatamente depois do Exemplo 1.8:
   acrescentar:
   A teoria de conjuntos apresentada nesta seção é um resumo da chamada Teoria Ingênua de Conjuntos (do inglês *Naïve Set Theory*) elaborada por Georg Cantor no final do século XIX. Tal teoria, apesar de simples, permite enunciar alguns paradoxos, entre os quais o mais famoso é o Paradoxo de Russell, proposto por Bertrand Russell em 1901, e que envolve apenas os conceitos de formação de conjunto e de pertencimento:

   > Seja S o conjunto formado por todos os conjuntos que não são elementos de si mesmos, e considere a pergunta: "S é elemento de si mesmo?"

   Para tentar responder à essa pergunta, pode-se considerar duas situações distintas. Na primeira, supõe-se que S seja um elemento de si mesmo. Então, de acordo com a definição, S não deveria fazer parte de S, uma vez que S contém apenas conjuntos que não são elementos de si mesmos. Por outro lado, pode-se supor o caso contrário, ou seja, que S não seja um elemento de si mesmo. Então, pela definição, S se qualifica como um elemento de si mesmo. Portanto, qualquer que seja o caso que se considere, temos uma contradição. Logo, a hipótese é falsa e não existe um conjunto S com tal característica.

   A fim de evitar a formulação de paradoxos como esse, foram desenvolvidas teorias de conjuntos alternativas, como é o caso da Teoria de Tipos do próprio Russell e também a Teoria Axiomática de Zermelo, que posteriormente serviu de base para a Teoria Axiomática de Zermelo-Fraenkel com o Axioma da Escolha (ZFC). Essa última é considerada um dos principais fundamentos da matemática moderna.

6. Página 33, antes do Exemplo 1.13:
   acrescentar:
   "A operação de união é também comutativa, ou seja:

   $$A \cup B = B \cup A$$

   para quaisquer conjuntos A e B. O conjunto vazio $\emptyset$ é o elemento neutro da operação de união."

7. Página 33, antes do Exemplo 1.14:
   acrescentar:
   "Da mesma forma que a união, a operação de intersecção é também comutativa:

   $$A \cap B = B \cap A$$

   para quaisquer conjuntos A e B. A intersecção de qualquer conjunto com o conjunto vazio produz como resultado o próprio conjunto vazio."

8. Página 35, antes do Teorema 1.2:
   acrescentar:
   Exemplo:
   Suponha $X = \{a,b\}, Y = \{b,c\}$ e considere $Z = \{a,b,c\}$. Então, $X \cup Y = \overline{(\overline{X}_Z \cap \overline{Y}_Z)}_Z = \overline{(\{c\} \cap \{a\})}_Z = \overline{\emptyset}_Z = \{a,b,c\}$. Por outro lado, $X \cap Y = \overline{(\overline{X}_Z \cup \overline{Y}_Z)}_Z = \overline{(\{c\} \cup \{a\})}_Z = \overline{\{a,c\}}_Z = \{b\}$.

9. Página 35, Teorema 1.2:
   onde se lê:
   "Considere-se $A \subseteq C$ e $B \subseteq C$, de forma que ..."
   leia-se:
   "Considere-se a condição (1) e, além disso, $A \subseteq C$ e $B \subseteq C$, de forma que ..."
   acrescentar, antes de "Portanto...", o seguinte parágrafo:
   "No caso da relação (ii), é possível ainda supor que $A = \emptyset$ e $B \neq \emptyset$. Se isto acontecer, então temos que $A \cap \overline{B} = \emptyset$, configurando assim uma possibilidade alternativa para satisfazer a condição (1). No entanto, se isto for verdadeiro, então a condição (2) não poderá ser satifeita, pois $\overline{A} \cap B \neq \emptyset$. O mesmo raciocínio pode ser aplicado para a condição (2), se $A \neq \emptyset$ e $B = \emptyset$.

10. Página 35, antes da Seção 1.2:
    acrescentar:

    - $\mathbb{Q}$, representando os números racionais.

11. Página 37, imediatamente antes do Exemplo 1.23:
    acrescentar:
    Teorema:
    Seja R uma relação binária reflexiva, simétrica e transitiva sobre um conjunto A. Então existe uma partição $P_0, P_1, ..., P_n$ de A tal que:

    - Se aRb, então $a,b \in P_i$, para algum $0 \leq i \leq n$;
    - Se $(a,b) \notin R$, então $a \in P_i$ e $b \in P_j$, com $i \neq j$.

    Prova:
    Para cada $a \in A$, considere o conjunto $classe(a) = \{b | aRb\}$. Tais conjuntos recebem o nome de classes de equivalência.

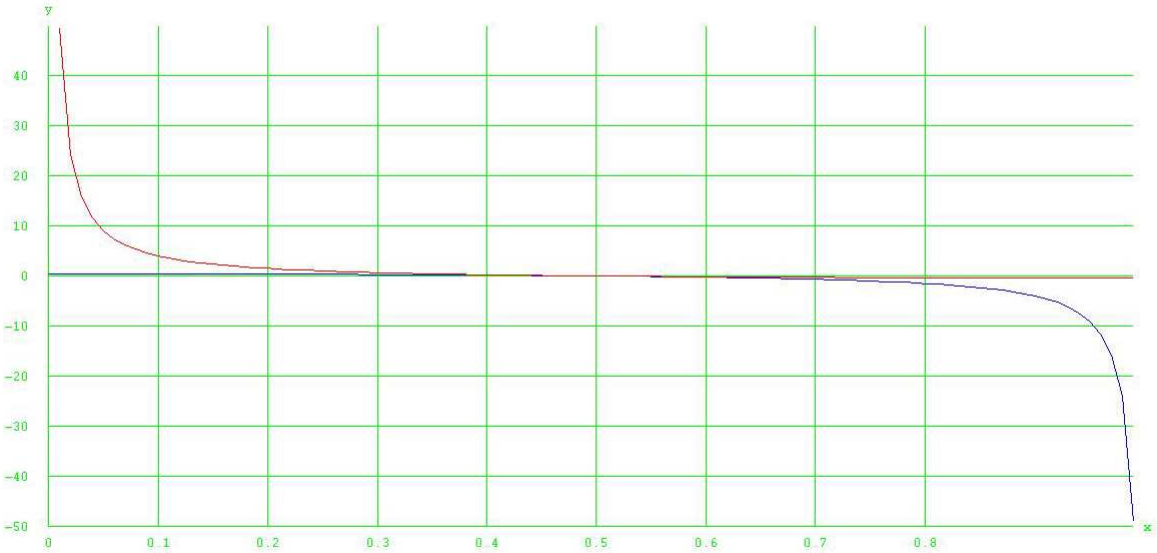
    - Primeira parte:
      Considere $c \in classe(a)$. Portanto, aRc. Por outro lado, como a R é reflexiva, segue que bRa. Como ela também é simétrica, bRa e aRc, segue que bRc, ou seja, que $c \in class(b)$. Portanto, todo elemento de $classe(a)$ também é elemento de $classe(b)$. Considere agora $c \in classe(b)$. Como a relação é simétrica, então bRc. Pela transitividade de R, temos que aRc, pois bRb e bRc. Logo, $c \in class(a)$ e todo elemento de $classe(b)$ também é elemento de $classe(a)$. Segue que $class(a) = class(b)$, e portanto que a e b pertencem à mesma classe de equivalência pois, pela reflexividade, $a \in class(a)$ e $b \in classe(b)$.
    - Segunda parte:
      Suponha que exista $c \in class(a) \cup class(b)$. Logo, aRc e bRc. Pela simetria, temos que cRb e, pela transitividade, podemos assumir que aRb. Mas aRb contradiz a hipótese de que $(a,b) \notin R$. Logo, a hipótese é falsa e não pode existir tal c. Ou seja, $class(a) \cup class(b) = \emptyset$ e a e b pertencem à classes de equivalência distintas.

Finalmente, resta provar que as classes de equivalência acima definidas constituem uma partição de A. Para isso, basta provar que:

- Todo elemento de A pertence à uma única classe de equivalência (ou seja, as classes são disjuntas duas a duas): Como aRa, segue que $a \in class(a)$. Logo, todo elemento a pertence à alguma classe de equivalência. Para provar que essa classe é única, suponha que $a \in c_1$ e $a \in c_2$, com $c_1 \neq c_2$. Conforme o resultado anterior, isso implicaria na falsidade de aRa, uma vez se tratam de elementos de classes distintas. Mas isso contradiz aRa, logo a hipótese é falsa e a não pode pertencer à duas classes diferentes.
- A união de todas as classes de equivalência resulta em A:
  Como todo elemento de A pertence a uma única classe de equivalência, a união de tais classes resulta em A.

12. Página 53, Exemplo 1.54:
    onde se lê:
    "e o conjunto dos números racionais também são enumeráveis"
    leia-se:
    "e o conjunto dos números racionais $\mathbb{Q}$ também são enumeráveis"

13. Página 53, depois da definição de $f(x)$ e antes do último parágrafo ("A prova de que...")
    acrescentar:
    O gráfico da Figura 1.XX ilustra o comportamento da função $f(x)$ no intervalo 0 a 1. Como se pode perceber, ela efetua um espalhamento do seu domínio de forma a mapear os elementos do mesmo em elementos de $\mathbb{R}$.



14. Página 54, Exemplo 1.55, logo depois de "diferente de todos eles."
    acrescentar:
    Sejam:
    $\mathbb{R}_0 = 0,\underbrace{d_{0_0}}\ d_{0_1}d_{0_2}d_{0_3}...d_{0_n}...$
    $\mathbb{R}_1 = 0,d_{1_0}\underbrace{d_{1_1}}\ d_{1_2}d_{1_3}...d_{1_n}...$
    $\mathbb{R}_2 = 0,d_{2_0}d_{2_1}\underbrace{d_{2_2}}\ d_{2_3}...d_{2_n}...$
    $\mathbb{R}_3 = 0,d_{3_0}d_{3_1}d_{3_2}\underbrace{d_{3_3}}\ ...d_{3_n}...$
    ...
    Então escolhe-se $0,x_0x_1x_2x_3...x_n...$ com $x_0 \neq d_{0_0}$, $x_1 \neq d_{1_1}$, $x_2 \neq d_{2_2}$, $x_3 \neq d_{3_3}$ etc.

15. Página 57, depois do Teorema 1.7 e antes da Seção 1.8
acrescentar:
Exemplo ...
Conforme mostrado anteriormente, o conjunto dos números reais $\mathbb{R}$ possui cardinalidade $\aleph_1$ e o conjunto dos números racionais $\mathbb{Q}$ possui cardinalidade $\aleph_0$. Logo, o resultado acima prova que o conjunto dos números irracionais $(\mathbb{R} - \mathbb{Q})$ possui cardinalidade $\aleph_1$.

16. Página 77:
acrescentar, depois do primeiro parágrafo:
"A noção de conjunto, introduzida no capítulo anterior, é utilizada para definir duas novas noções fundamentais: alfabetos (conjuntos finitos e não-vazios cujos elementos são símbolos) e linguagens (conjuntos finitos ou infinitos cujos elementos são cadeias construídas com os símbolos de um alfabeto).

Símbolos podem ser agrupados na forma de um conjunto, caso em que o mesmo recebe o nome de alfabeto. Conjuntos, por outro lado, podem ser formados por elementos de outra natureza, e não apenas por símbolos. É o caso, por exemplo, de conjuntos formados por cadeias (seqüências finitas de símbolos) e conjuntos cujos elementos também são conjuntos."

17. Página 77, Seção 2.1:
acrescentar, ao término do primeiro parágrafo:
" Um alfabeto é, portanto, um conjunto onde os elementos do mesmo são símbolos usados na construção de cadeias".

18. Página 79, Seção 2.2:
onde se lê:
"Uma **linguagem formal** é um conjunto, finito ou infinito, de cadeias de comprimento finito, formadas pela concatenação de elementos de um alfabeto finito e não-vazio."
leia-se:
"Uma **linguagem formal** é um conjunto, finito ou infinito, de cadeias de comprimento finito, formadas pela concatenação de símbolos de um alfabeto finito e não-vazio. Uma linguagem formal é, portanto, um conjunto onde os elementos do mesmo são cadeias construídas sobre um alfabeto. Note que uma linguagem pode ser vazia (ou seja, pode não conter nenhuma cadeia) ou ainda pode conter a cadeia vazia como um dos seus elementos."

19. Página 79:
acrescentar, no final de último parágrafo
"Estas novas operações, definidas a seguir, derivam diretamente da operação de concatenação de cadeias introduzida na seção anterior".

20. Página 80, dois primeiros parágrafos
onde se lê:
Antes de apresentá-las, convém notar a distinção que há entre os seguintes conceitos: cadeia vazia ε, conjunto vazio $\emptyset$ e o conjunto que contém apenas a cadeia vazia $\{\varepsilon\}$.

O primeiro deles, ε, denota a **cadeia** vazia, ou seja, uma cadeia de comprimento zero, ao passo que os dois seguintes são casos particulares de **linguagens** (que por sua vez são conjuntos): $\emptyset$ denota uma linguagem vazia, ou seja, uma linguagem que não contém nenhuma cadeia, e $\{\varepsilon\}$ denota uma linguagem que contém uma única cadeia, a cadeia vazia. Observe-se que $|\emptyset| = 0$ e $|\{\varepsilon\}| = 1$.

leia-se:
Antes de apresentá-las, convém notar a distinção que há entre os seguintes conceitos:

- cadeia vazia ε;
- conjunto vazio $\emptyset$;
- conjunto que contém apenas a cadeia vazia $\{\varepsilon\}$;
- conjunto que contém apenas o conjunto vazio $\{\emptyset\}$.

O primeiro deles, ε, denota a cadeia vazia, ou seja, uma cadeia de comprimento zero, ao passo que os demais são casos particulares de conjuntos: $\emptyset$ denota uma linguagem vazia, ou seja, uma linguagem que não contém nenhuma cadeia, $\{\varepsilon\}$ denota uma linguagem que contém uma única cadeia (a cadeia vazia), e $\{\emptyset\}$ denota um conjunto que contém um único elemento, o conjunto vazio. Observe-se que $|\varepsilon| = |\emptyset| = 0$ e $|\{\varepsilon\}| = |\{\emptyset\}| = 1$.

21. Página 93, Exemplo 2.24
onde se lê:
Considere $G_2 = (V_2, \Sigma_2, P_2, S)$, com:

$$\begin{aligned} V_2 &= \{a, b, c, S, B, C\} \\ \Sigma_2 &= \{a, b, c\} \\ P_2 &= \{S \to aSBC, S \to abC, CB \to BC, bB \to bb, bC \to bc, cC \to cc\} \end{aligned}$$

A linguagem gerada por G_2 é $\{a^n b^n c^n | n \geq 1\}$. De fato, a seqüência de derivações iniciada com a regra $S \to abC$ conduz à geração da sentença abc ($S \Rightarrow abC \Rightarrow abc$). Seqüências iniciadas com a aplicação repetida da regra $S \to aSBC$ conduzem às seguintes formas sentenciais subseqüentes:

$$S \Rightarrow^i a^i S(BC)^i \Rightarrow a^i abC(BC)^i \Rightarrow^i a^{i+1} bB^i C^{i+1} \Rightarrow^i a^{i+1} b^{i+1} C^{i+1}$$

A aplicação da regra $bC \to bc$, seguida da aplicação sucessiva da regra $cC \to cc$, resulta em:

$$\Rightarrow a^{i+1} bb^i C^i \Rightarrow^i a^{i+1} b^{i+1} c^{i+1}$$

gerando, portanto, as sentenças *aabbcc*, *aaabbbccc* etc. A sentença *aabbcc*, por exemplo, é derivada da seguinte forma nessa gramática:

$$S \Rightarrow aSBC \Rightarrow aabCBC \Rightarrow aabBCC \Rightarrow aabbCC \Rightarrow aabbbcC \Rightarrow aabbcc$$

pela aplicação, respectivamente, das produções:

$$S \to aSBC, S \to abC, CB \to BC, bB \to bb, bC \to bc \text{ e } cC \to cc$$

leia-se:
Considere $G_2 = (V_2, \Sigma_2, P_2, S)$, com:

$$\begin{aligned} V_2 &= \{a, b, c, S, B, C\} \\ \Sigma_2 &= \{a, b, c\} \\ P_2 &= \{S \to aSBC, S \to abC, CB \to BC, bB \to bb, bC \to bc, cC \to cc\} \end{aligned}$$

A linguagem gerada por G_2 é $\{a^n b^n c^n | n \geq 1\}$. De fato, a seqüência de derivações iniciada com a regra $S \to abC$ conduz à geração da sentença abc ($S \Rightarrow abC \Rightarrow abc$). Por outro lado, seqüências iniciadas com a aplicação repetida i vezes da regra $S \to aSBC$ conduzem à geração da seguinte forma sentencial subseqüente:

$$S \Rightarrow^i a^i S(BC)^i$$

A posterior aplicação da regra $S \to abC$ faz com que:

$$a^i S(BC)^i \Rightarrow a^i abC(BC)^i = a^{i+1} b(CB)^i C$$

Através da aplicação sucessiva da regra $CB \to BC$, obtém-se agora:

$$a^{i+1} b(CB)^i C \Rightarrow^* a^{i+1} bB^i C^i C = a^{i+1} bB^i C^{i+1}$$

Finalmente, a aplicação i vezes da regra $bB \to bb$ faz com que todos os símbolos B sejam substituídos por símbolos b:

$$a^{i+1} bB^i C^{i+1} \Rightarrow^i a^{i+1} bb^i C^{i+1} = a^{i+1} b^{i+1} C^{i+1}$$

A aplicação uma única vez da regra $bC \to bc$ substitui o primeiro símbolo da cadeia de símbolos C pelo símbolo c:

$$a^{i+1}b^{i+1}C^{i+1} \Rightarrow a^{i+1}b^{i+1}cC^{i}$$

Para terminar, a aplicação i vezes da regra $cC \to cc$ substitui todos os demais símbolos C por símbolos c:

$$a^{i+1}b^{i+1}cC^{i} \Rightarrow^{i} a^{i+1}b^{i+1}cc^{i} = a^{i+1}b^{i+1}c^{i+1}$$

A forma sentencial:

$$a^{i+1}b^{i+1}c^{i+1}$$

gera, portanto, as sentenças *aabbcc*, *aaabbbccc* etc. A sentença *aabbcc*, por exemplo, é derivada da seguinte forma nessa gramática:

$$S \Rightarrow aSBC \Rightarrow aabCBC \Rightarrow aabBCC \Rightarrow aabbCC \Rightarrow aabbbcC \Rightarrow aabbcc$$

pela aplicação, respectivamente, das produções:

$$S \to aSBC, S \to abC, CB \to BC, bB \to bb, bC \to bc \text{ e } cC \to cc$$

22. Página 142, antes de **Teorema 3.1**:
acrescentar:
Uma **gramática linear** (não necessariamente à esquerda ou à direita) é uma gramática que possui no máximo um único símbolo não-terminal do lado direito das suas regras. Assim, gramáticas lineares à esquerda ou à direita são casos particulares de gramáticas lineares: toda gramática linear à esquerda ou à direita é também uma gramática linear porém o inverso nem sempre é verdadeiro. A linguagem gerada por uma gramática linear não é necessariamente uma linguagem regular. Por exemplo, a gramática linear abaixo gera uma linguagem linear que é livre de contexto e não-regular:

$$\begin{aligned} S &\to aSa \\ S &\to b \end{aligned}$$

Logo, toda linguagem regular é também linear, porém nem toda linguagem linear é regular.

23. Página 142, depois do enunciado do **Teorema 3.1** até antes do parágrafo "Alguns autores consideram..." na página 144:
substituir todo o conteúdo pelo que segue:
A demonstração da equivalência entre gramáticas lineares à direita e à esquerda é feita em duas etapas. Na primeira, mostra-se como obter uma outra gramática linear à direita G, a partir da gramática linear à direita fornecida G_1, tal que $L(G) = L(G_1)^R$ (Algoritmo 3.1). Em seguida, mostra-se como obter uma gramática linear à esquerda G_2, a partir da gramática linear à direita G obtida na etapa anterior, tal que $L(G_2) = L(G)^R$ (Algoritmo 3.2). Como $L(G) = L(G_1)^R$, segue que $L(G_2) = (L(G_1)^R)^R = L(G_1)$.

Algoritmo 3.1 (Linguagem reversa, GLD/GLD) "Obtenção de uma gramática linear à direita que gera L^R a partir de uma gramática linear à direita que gera L."

- Entrada: uma gramática linear à direita $G' = (V, \Sigma, P', S)$;
- Saída: uma gramática linear à direita $G'' = (V \cup \{S''\}, \Sigma, P'', S'')$, tal que $L(G'') = L(G')^R$;
- Método:
  (a) $P'' \leftarrow \emptyset$;
  (b) Se $X \to aY \in P'$, então $Y \to aX \in P''$;
  (c) Se $X \to Y \in P'$, então $Y \to X \in P''$;
  (d) Se $X \to a \in P'$, então $S'' \to aX \in P''$;
  (e) Se $X \to \varepsilon \in P'$, então $S'' \to X \in P''$;
  (f) $S \to \varepsilon \in P''$.

Com G'' construída dessa forma, é possível demonstrar que $L(G'') = L^R(G')$. Note que G'' é, por construção, também linear à direita.

As gramáticas lineares à direita geram formas sentenciais em que o símbolo não-terminal é sempre o último símbolo das mesmas ($\gamma X, \gamma \in \Sigma^*, X \in N$). Portanto, as sentenças da linguagem vão sendo construídas através da inserção de novos símbolos terminais sempre à direita das formas sentenciais, imediatamente antes do símbolo não-terminal (daí o nome "linear à direita"). A estratégia do algoritmo acima é gerar a linguagem reversa porém inserindo os novos símbolos ainda no final das formas sentenciais.

Exemplo 3.1 Considere a gramática linear à direita G_1 definida a seguir:

$$\begin{aligned} S &\rightarrow aS \\ S &\rightarrow bS \\ S &\rightarrow P \\ P &\rightarrow cQ \\ Q &\rightarrow cR \\ R &\rightarrow dR \\ R &\rightarrow d \end{aligned}$$

$L(G_1)$ corresponde ao conjunto das cadeias w sobre $\{a,b,c,d\}$ tais que:

(a) w começa com zero ou mais símbolos a ou b;
(b) w continua com exatamente dois símbolos c;
(c) w termina com um ou mais símbolos d.

Uma gramática linear à direita G', tal que $L(G') = L^R(G_1)$, pode ser obtida pela aplicação do ***Algoritmo 3.1***:

$$\begin{aligned} S &\rightarrow aS \\ S &\rightarrow bS \\ P &\rightarrow S \\ Q &\rightarrow cP \\ R &\rightarrow cQ \\ R &\rightarrow dR \\ S'' &\rightarrow dR \\ S &\rightarrow \varepsilon \end{aligned}$$

Logo, $L^R(G_1)$ é tal que:

(a) w começa com um ou mais símbolos d.
(b) w continua com exatamente dois símbolos c;
(c) w termina com zero ou mais símbolos a ou b;

Note que G_1 e G' são ambas lineares à direita e, além disso, $L(G') = L^R(G_1)$.

Exemplo 3.2 Considere a gramática linear à direita G_1 definida a seguir:

$$\begin{aligned} S &\rightarrow aS \\ S &\rightarrow X \\ X &\rightarrow bX \\ X &\rightarrow Y \\ Y &\rightarrow cY \\ Y &\rightarrow \varepsilon \end{aligned}$$

$L(G_1)$ corresponde ao conjunto das cadeias w sobre $\{a,b,c\}$ tais que:

(a) w começa com zero ou mais símbolos a;
(b) w continua com zero ou mais símbolos b;
(c) w termina com zero ou mais símbolos c.

Uma gramática linear à direita G', tal que $L(G') = L^R(G_1)$, pode ser obtida pela aplicação do ***Algoritmo 3.1***:

$$\begin{aligned} S &\rightarrow aS \\ X &\rightarrow S \\ X &\rightarrow bX \\ Y &\rightarrow X \\ Y &\rightarrow cY \\ S'' &\rightarrow Y \\ S &\rightarrow \varepsilon \end{aligned}$$

Logo, $L^R(G_1)$ é tal que:

(a) w começa com zero ou mais símbolos c.
(b) w continua com zero ou mais símbolos b;
(c) w termina com zero ou mais símbolos a;

Note que G_1 e G' são ambas lineares à direita e, além disso, $L(G') = L^R(G_1)$.

Algoritmo 3.2 (Linguagem reversa, GLD/GLE) "Obtenção de uma gramática linear à esquerda que gera L^R a partir de uma gramática linear à direita que gera L."

- Entrada: uma gramática linear à direita $G' = (V, \Sigma, P', S)$;
- Saída: uma gramática linear à esquerda $G'' = (V, \Sigma, P'', S)$, tal que $L(G'') = L(G')^R$;
- Método:
  (a) $P'' \leftarrow \emptyset$;
  (b) Se $\alpha \rightarrow \beta \in P'$, então $\alpha \rightarrow \beta^R \in P''$.

Com G'' construída dessa forma, é possível demonstrar que $L(G'') = L^R(G')$. Note que G'' é, por construção, linear à esquerda.

Este algoritmo mostra como obter uma gramática linear à esquerda que gera a mesma linguagem de uma dada gramática linear à direita. De fato, a linguagem gerada é a mesma, com a única diferença de que agora as sentenças são construídas no sentido oposto, ou seja, através da inserção de símbolos terminais sempre à esquerda das formas sentenciais, imediatamente depois do símbolo não-terminal, que agora é o primeiro das formas sentenciais geradas ($X\gamma, \gamma \in \Sigma^*, X \in N$). Só muda, portanto, o sentido em que as sentenças são geradas.

Exemplo 3.3 Considere a gramática linear à direita G' obtida no ***Exemplo 3.1***. A aplicação do ***Algoritmo 3.2*** à mesma resulta na gramática G_2:

$$\begin{aligned} S &\rightarrow Sa \\ S &\rightarrow Sb \\ P &\rightarrow S \\ Q &\rightarrow Pc \\ R &\rightarrow Qc \\ R &\rightarrow Rd \\ S'' &\rightarrow Rd \\ S &\rightarrow \varepsilon \end{aligned}$$

Como é fácil observar, G_2 é linear à esquerda e $L(G_2) = L(G')^R = (L(G_1)^R)^R = L(G_1)$, onde G_1 é a gramática linear à direita do ***Exemplo 3.1***. De fato, considerem-se as derivações da sentença *abaccdd*, respectivamente em G_1 e G_2:

- $S \overset{G_1}{\Rightarrow} aS \overset{G_1}{\Rightarrow} abS \overset{G_1}{\Rightarrow} abaS \overset{G_1}{\Rightarrow} abaP \overset{G_1}{\Rightarrow} abacQ \overset{G_1}{\Rightarrow} abaccR \overset{G_1}{\Rightarrow} abaccdR \overset{G_1}{\Rightarrow} abaccdd$
- $S'' \overset{G_2}{\Rightarrow} Rd \overset{G_2}{\Rightarrow} Rdd \overset{G_2}{\Rightarrow} Qcdd \overset{G_2}{\Rightarrow} Pccdd \overset{G_2}{\Rightarrow} Sccdd \overset{G_2}{\Rightarrow} Saccdd \overset{G_2}{\Rightarrow} Sbaccd \overset{G_2}{\Rightarrow} Sabaccdd \overset{G_2}{\Rightarrow} abaccdd$

Exemplo 3.4 Considere a gramática linear à direita G' obtida no ***Exemplo 3.2***. A aplicação do ***Algoritmo 3.2*** à mesma resulta na gramática G_2:

$$\begin{aligned} S &\rightarrow Sa \\ X &\rightarrow S \\ X &\rightarrow Xb \\ Y &\rightarrow X \\ Y &\rightarrow Yc \\ S'' &\rightarrow Y \\ S &\rightarrow \varepsilon \end{aligned}$$

Como é fácil observar, G_2 é linear à esquerda e $L(G_2) = L(G')^R = (L(G_1)^R)^R = L(G_1)$, onde G_1 é a gramática linear à direita do ***Exemplo 3.2***.

24. Página 169, Teorema 3.4, imediatamente antes do Algoritmo 3.5
acrescentar:
O mecanismo de mapeamento é baseado na elimnação sistemática das transições em vazio do autômato original, substituindo-as por transições não-vazias de modo que o autômato resultante aceite as mesmas cadeias que o autômato original.

Se o autômato original possui uma transição em vazio de um estado q_i para um estado q_j, e se do estado q_j existem transições para, por exemplo, os estados q_k e q_l, respectivamente com os símbolos a e b, então o autômato modificado deverá adicionar transições de q_i para q_k com o símbolo a e de q_i para q_l com o símbolo b. Dessa maneira, o conjunto de cadeias que são processadas a partir do estado q_i permanece o mesmo em ambos os casos. Se q_j for um estado final, então q_i deverá ser tornado final, a fim de permitir a aceitação das cadeias que conduzem o autômato resultante a uma configuração final nesse estado. As Figuras 3.XX e 3.YY ilustram essa idéia.

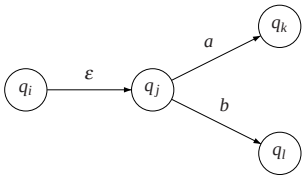


Figura 3.XX Situação com transição em vazio original

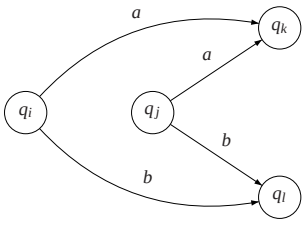


Figura 3.YY Situação sem transição em vazio equivalente à da Figura 3.XX

25. Página 170, imediatamente antes do último parágrafo
acrescentar:

Conforme mencionado anteriormente, o Algoritmo 3.5 pode não produzir o resultado desejado caso o autômato M possua um ou mais ciclos de transições em vazio. Um detalhamento dessa questão, no entanto, revela a condição exata em que o algoritmo falha (note-se que a simples existência de ciclos formado por transições em vazio não é condição suficiente para caracterizar a situação em que o algoritmo deixa de produzir o resultado desejado):

(a) M possui pelo menos um ciclo formado formado por transições em vazio;

(b) Existe em M pelo menos um estado q_i, não pertencente ao ciclo, um estado q_j, pertencente ao ciclo, e um caminho formado por transições em vazio que tem q_i como origem e q_j como destino.

Num caso como esse, se a escolha da transição a ser eliminada recair sobre uma transição pertencente ao caminho que conduz de q_i até q_j, antes de se eliminar pelo menos uma das transições pertencente ao ciclo, ela trará como conseqüência o ressurigmento recorrente da transição original e a conseqüente impossibilidade de se alcançar o objetivo inicial. As Figuras 3.XX até 3.XX ilustram a situação.

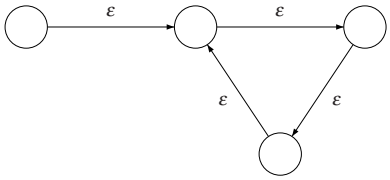


Figura 3.XX Tentativa de eliminação da transição em vazio externa ao ciclo, passo 1

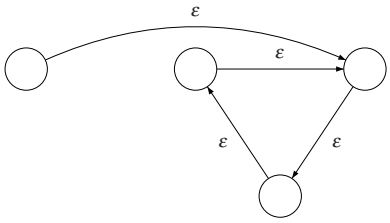


Figura 3.XX Tentativa de eliminação da transição em vazio externa ao ciclo, passo 2

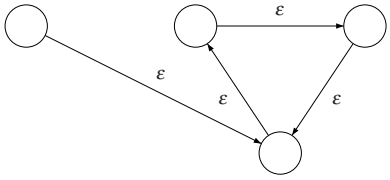


Figura 3.XX Tentativa de eliminação da transição em vazio externa ao ciclo, passo 3

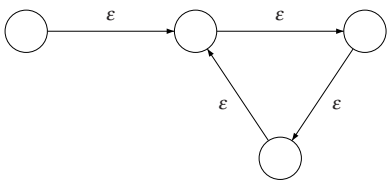


Figura 3.XX Tentativa de eliminação da transição em vazio externa ao ciclo, passo 4

Para evitar que isso aconteça, uma possível solução seria eliminar inicialmente alguma transição pertencente ao ciclo, para apenas depois considerar as demais transições do ciclo e as do caminho entre q_i e q_j, não importando a ordem em que isso for feito. As Figuras 3.XX até 3.XX ilustram a situação.

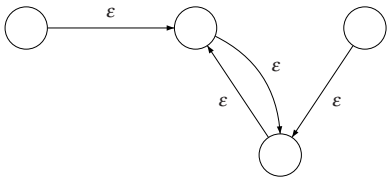


Figura 3.XX Eliminação da transição em vazio pertencente ao ciclo, passo 1

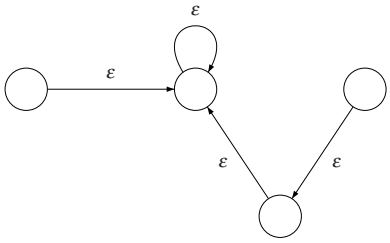


Figura 3.XX Eliminação da transição em vazio pertencente ao ciclo, passo 2

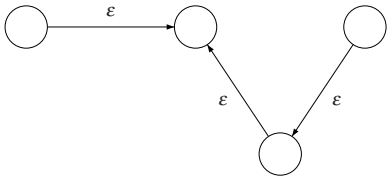


Figura 3.XX Eliminação da transição em vazio pertencente ao ciclo, passo 3

Essa solução, no entanto, exige que se determine antecipadamente se *M* possui ciclos formados por transições em vazio, para apenas depois determinar a ordem em que a eliminação das suas transições em vazio poderá acontecer. Uma outra solução, que independe desse tipo de análise, e portanto se configura geral, será apresentada mais adiante.

Exemplo 3.XX:

Como exemplo, considere o autômato da Figura 3.XX. Nesse caso, qualquer tentativa de eliminar as transições em vazio que vão de q_0 para q_1, de q_1 para q_2, ou mesmo de q_3 para q_2, sem antes eliminar os respectivos ciclos formados por transições em vazio que são atingidos, respectivamente, a partir dos estados q_0, q_1 e q_3, resultará em insucesso, com a iteração infinita dos passos do algoritmo. Essas soluções não funcionam pois, nelas, os estados de origem dos caminhos formados por transições em vazio (q_0, q_1 e q_3) não fazem parte do ciclo em questão.

Uma solução, nesse caso, seria eliminar inicialmente a transição em vazio que vai de q_2 para q_1 e depois as demais, em qualquer ordem. Ou ainda, a que vai de q_2 para q_3, seguida das demais. Esses casos funcionam pois o estado de origem do caminho formado por transições em vazio (q_2) é também parte do ciclo que é atingido pelo caminho.

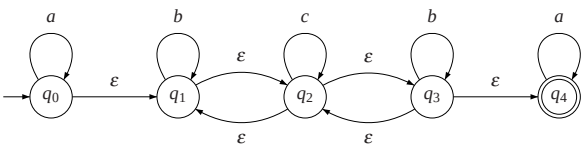


Figura 3.XX Eliminação da transição em vazio pertencente ao ciclo, passo 3

26. Página 187
onde se lê:
"A eliminação de estados inacessíveis ou inúteis:

- Não faz surgir não-determinismos;
- Não introduz transições em vazio."

leia-se:
"A eliminação de estados inacessíveis ou inúteis:

- Não faz surgir não-determinismos;
- Não introduz transições em vazio;
- Pode ser feita em qualquer ordem."

acrescentar, imediatamente antes do parágrafo iniciado com "Logo, é fácil provar...":
"De fato, consideremos dois cenários:

- Aplicação do algoritmo de eliminação de estados inacessíveis seguido da aplicação do algoritmo de eliminação de estados inúteis:
  - A aplicação do primeiro algoritmo produz um autômato isento de estados inacessíveis, mas que ainda pode conter estados inúteis;
  - A aplicação do segundo algoritmo produz um autômato isento de estados inúteis. A única possibilidade para a caracterização de novos estados inacessíveis no autômato, anteriormente inexistentes, ocorre quando estes são sucessores de um estado acessível e inútil eliminado pelo algoritmo. Mas, neste caso, eles são também inúteis e são também eliminados pelo segundo algoritmo; logo, todos os estados que permanecem após a aplicação do segundo algoritmo são acessíveis e úteis;
- Aplicação do algoritmo de eliminação de estados inúteis seguido da aplicação do algoritmo de eliminação de estados inacessíveis:
  - A aplicação do primeiro algoritmo produz um autômato isento de estados inúteis, mas que ainda pode conter estados inacessíveis;
  - A aplicação do segundo algoritmo produz um autômato isento de estados inacessíveis. A única possibilidade para a caracterização de novos estados inúteis no autômato ocorre quando estes são antecessores de um estado inacessível e útil eliminado pelo algoritmo. Mas, neste caso, eles são também inacessíveis e são também eliminados pelo segundo algoritmo; logo, todos os estados que permanecem após a aplicação do segundo algoritmo são acessíveis e úteis;"

onde se lê:
"3. Eliminação de estados inacessíveis e inúteis, caso existam."
leia-se:
"3. Eliminação de estados inacessíveis e inúteis (em qualquer ordem), caso existam."
onde se lê:
"Qualquer outra ordem poderá..."
leia-se:
"Qualquer outra ordem na aplicação destes três passos poderá..."

27. Página 198, logo depois do primeiro parágrafo e antes do Exemplo 3.33
acrescentar:
As figuras seguintes ilustram a operação do Algoritmo 3.13. Cada figura representa o mesmo sistema de equações, em momentos diferentes do seu processamento. Em todos os casos, elas representam um sistema com m equações e m variáveis, com uma equação em cada linha. Assim, a primeira linha representa a primeira equação (da variável X_1) , a segunda linha a segunda equação (da variável X_2) e assim por diante. Dentro de cada célula, o número i que varia de 1 a m indica que a equação em questão pode conter referência para a variável correspondente (X_i). Desta

forma, na condição inicial do sistema, cada equação pode conter referências à todas as demais variáveis do sistema. Na medida em que o passo 2 do algoritmo é executado, no entanto, vão sendo eliminadas referências às variáveis das equações seguintes. No final do passo 2, a última equação (da variável X_m) refere-se apenas à própria variável que está sendo definida. Terminado o movimento descendente, tem início o movimento ascendente representado pelo passo 4 do algoritmo. Na primeira passagem, ele resolve a última equação, por aplicação do Teorema 3.18, e substitui o valor da variável X_m em todas as anteriores. Desta forma, todas as equações vão sendo resolvidas de forma a não conter referências à nenhuma variável. Na última passagem do passo 4, todas as variáveis do sistema estão representadas por equações regulares que não contêm variáveis.

Situação inicial:

1	2	3	4	...	$m-3$	$m-2$	$m-1$	m
1	2	3	4	...	$m-3$	$m-2$	$m-1$	m
1	2	3	4	...	$m-3$	$m-2$	$m-1$	m
1	2	3	4	...	$m-3$	$m-2$	$m-1$	m
				...				
1	2	3	4	...	$m-3$	$m-2$	$m-1$	m
1	2	3	4	...	$m-3$	$m-2$	$m-1$	m
1	2	3	4	...	$m-3$	$m-2$	$m-1$	m
1	2	3	4	...	$m-3$	$m-2$	$m-1$	m

Passo 2, linha 1:

1	2	3	4	...	$m-3$	$m-2$	$m-1$	m
	2	3	4	...	$m-3$	$m-2$	$m-1$	m
	2	3	4	...	$m-3$	$m-2$	$m-1$	m
	2	3	4	...	$m-3$	$m-2$	$m-1$	m
				...				
	2	3	4	...	$m-3$	$m-2$	$m-1$	m
	2	3	4	...	$m-3$	$m-2$	$m-1$	m
	2	3	4	...	$m-3$	$m-2$	$m-1$	m
	2	3	4	...	$m-3$	$m-2$	$m-1$	m

Passo 2, linha 2:

1	2	3	4	...	$m-3$	$m-2$	$m-1$	m
	2	3	4	...	$m-3$	$m-2$	$m-1$	m
		3	4	...	$m-3$	$m-2$	$m-1$	m
		3	4	...	$m-3$	$m-2$	$m-1$	m
				...				
		3	4	...	$m-3$	$m-2$	$m-1$	m
		3	4	...	$m-3$	$m-2$	$m-1$	m
		3	4	...	$m-3$	$m-2$	$m-1$	m
		3	4	...	$m-3$	$m-2$	$m-1$	m

Passo 2, linha 3:

1	2	3	4	...	$m-3$	$m-2$	$m-1$	m
	2	3	4	...	$m-3$	$m-2$	$m-1$	m
		3	4	...	$m-3$	$m-2$	$m-1$	m
			4	...	$m-3$	$m-2$	$m-1$	m
				...				
			4	...	$m-3$	$m-2$	$m-1$	m
			4	...	$m-3$	$m-2$	$m-1$	m
			4	...	$m-3$	$m-2$	$m-1$	m
			4	...	$m-3$	$m-2$	$m-1$	m

Passo 2, linha $m-1$:

1	2	3	4	...	$m-3$	$m-2$	$m-1$	m
	2	3	4	...	$m-3$	$m-2$	$m-1$	m
		3	4	...	$m-3$	$m-2$	$m-1$	m
			4	...	$m-3$	$m-2$	$m-1$	m
				...				
					$m-3$	$m-2$	$m-1$	m
						$m-2$	$m-1$	m
							$m-1$	m
								m

Passo 4, linha m:

1	2	3	4	...	$m-3$	$m-2$	$m-1$	
	2	3	4	...	$m-3$	$m-2$	$m-1$	
		3	4	...	$m-3$	$m-2$	$m-1$	
			4	...	$m-3$	$m-2$	$m-1$	
				...				
					$m-3$	$m-2$	$m-1$	
						$m-2$	$m-1$	
							$m-1$	

Passo 4, linha $m-1$:

1	2	3	4	...	$m-3$	$m-2$		
	2	3	4	...	$m-3$	$m-2$		
		3	4	...	$m-3$	$m-2$		
			4	...	$m-3$	$m-2$		
				...				
					$m-3$	$m-2$		
						$m-2$		

Passo 4, linha $m-2$:

1	2	3	4	...	$m-3$			
	2	3	4	...	$m-3$			
		3	4	...	$m-3$			
			4	...	$m-3$			
				...				
					$m-3$			

Passo 4, linha 1:

				...				

28. Página 248, Teorema 3.24, imediatamente depois do segundo parágrafo
acrescentar:
Uma outra forma de entender esse resultado é a seguinte: admita-se, por hipótese, que não exista nenhuma cadeia $w \in L$ tal que $0 \leq |w| < n$, e considere-se a cadeia w' como sendo aquela que possui o menor comprimento entre todas as cadeias de L cujo comprimento é maior ou igual a n (se L é não-vazia e não há em L, por hipótese, cadeias de comprimento menor que n, então deve haver pelo menos uma cadeia que satisfaça essa condição). Como $|w'| \geq n$, então $w' = xyz$, com $1 \leq |y| \leq n$, $|xz| < |w'|$ e $xz \in L(M)$. Seguem duas possibilidades:

   (a) Se $|xz| \geq n$, isso contradiz a hipótese de que w' seria a cadeia de L com o menor comprimento entre todas as que possuem comprimento maior ou igual a n;
   (b) Se $|xz| < n$, isso contradiz a hipótese de que não existiria nenhuma cadeia de L com comprimento maior ou igual a 0 e menor que n.

   Portanto, em qualquer caso a hipótese é falsa e deve existir pelo menos uma cadeia $w \in L$ tal que $0 \leq |w| < n$.

29. Página 249, Teorema 3.25, imediatamente depois do antepenúltimo parágrafo
acrescentar:
Uma outra forma de entender esse resultado é a seguinte: admita-se, por hipótese, que não exista nenhuma cadeia $w \in L$ tal que $n \leq |w| < 2n$, e considere-se a cadeia w' como sendo aquela que possui o menor comprimento entre todas as cadeias de L cujo comprimento é maior ou igual a $2n$ (se L é infinita, então deve haver pelo menos uma cadeia que satisfaça essa condição). Como $|w'| \geq n$, então $w' = xyz$, com $1 \leq |y| \leq n$, $|xz| < |w'|$ e $xz \in L(M)$. Além disso, como $|w'| \geq 2n$ e $1 \leq |y| \leq n$, então $|xz| \geq n$. Seguem duas possibilidades:

   (a) Se $|xz| \geq 2n$, isso contradiz a hipótese de que w' seria a cadeia de L com o menor comprimento entre todas as que possuem comprimento maior ou igual a $2n$;
   (b) Se $|xz| < 2n$, isso contradiz a hipótese de que não existiria nenhuma cadeia de L com comprimento maior ou igual a n e menor que $2n$.

   Portanto, em qualquer caso a hipótese é falsa e deve existir pelo menos uma cadeia $w \in L$ tal que $n \leq |w| < 2n$.

30. Página 352, imediatamente depois do Exemplo 4.37
acrescentar:
Exemplo 4.XX Considere a gramática abaixo, que gera a linguagem do Exemplo 4.37 e que não foi convertida para a Forma Normal de Greibach.

$$\begin{aligned} \{E &\rightarrow T|T+E, \\ T &\rightarrow F|F*T, \\ F &\rightarrow (E)|a\} \end{aligned}$$

A aplicação do Algoritmo 4.8 resulta no autômato de pilha não-determinístico cuja função de transição δ é:

$$\begin{aligned} \{(q,\varepsilon,E) &\rightarrow \{(q,T),(q,T+E)\}, \\ (q,\varepsilon,T) &\rightarrow \{(q,F),(q,F*T)\}, \\ (q,\varepsilon,F) &\rightarrow \{(q,(E)),(q,a)\}, \\ (q,a,a) &\rightarrow \{(q,\varepsilon)\}, \\ (q,(,() &\rightarrow \{(q,\varepsilon)\}, \\ (q,),)) &\rightarrow \{(q,\varepsilon)\}, \\ (q,+,+) &\rightarrow \{(q,\varepsilon)\}, \\ (q,*,*) &\rightarrow \{(q,\varepsilon)\}\} \end{aligned}$$

Entre as várias possibilidades de movimentação que esse autômato possui para a cadeia de entrada $a+a*a$, e que simulam derivações mais à esquerda na gramática, a seqüência abaixo conduz o autômato à aceitação da mesma:

$(q,a+a*a,E) \Rightarrow (q,a+a*a,T+E) \Rightarrow (q,a+a*a,F+E) \Rightarrow (q,a+a*a,a+E) \Rightarrow (q,+a*a,+E) \Rightarrow (q,a*a,E) \Rightarrow (q,a*a,T) \Rightarrow (q,a*a,F*T) \Rightarrow (q,a*a,a*T) \Rightarrow (q,*a,*T) \Rightarrow (q,a,T) \Rightarrow (q,a,F) \Rightarrow (q,a,a) \Rightarrow (q,\varepsilon,\varepsilon)$

31. Página 353, Algoritmo 4.13
onde se lê:
para qualquer seqüência de $q_j \in Q, 2 \leq j \leq (k+1)$*;*
leia-se:
para toda e qualquer seqüência de estados $q_2, q_3, ..., q_k, q_{k+1}$ *que possa ser obtida a partir de Q (repetições são permitidas);*

32. Página 355, Exemplo 4.38, entre a primeira e a segunda linha
acrescentar:
A obtenção de G, tal que $L(G) = V(M)$, tem como ponto de partida as regras:

$$\begin{aligned} S &\rightarrow [q_0 Z_0 q_0] \\ S &\rightarrow [q_0 Z_0 q_1] \end{aligned}$$

Analisando-se as transições de M individualmente, as seguintes regras adicionais são obtidas:

- Para $\delta(q_0, a, Z_0) = (q_0, XZ_0)$, considerar $[q_0 Z_0 \_\_] \rightarrow a[q_0 X \_\_][\_\_ Z_0 \_\_]$ com as listas (q_0, q_0), (q_0, q_1), (q_1, q_0) e (q_1, q_1), gerando:

$$\begin{aligned} [q_0 Z_0 q_0] &\rightarrow a[q_0 X q_0][q_0 Z_0 q_0] \\ [q_0 Z_0 q_1] &\rightarrow a[q_0 X q_0][q_0 Z_0 q_1] \\ [q_0 Z_0 q_0] &\rightarrow a[q_0 X q_1][q_1 Z_0 q_0] \\ [q_0 Z_0 q_1] &\rightarrow a[q_0 X q_1][q_1 Z_0 q_1] \end{aligned}$$

- Para $\delta(q_0, a, X) = (q_0, XX)$, considerar $[q_0 X \_\_] \rightarrow a[q_0 X \_\_][\_\_ X \_\_]$ com as listas (q_0, q_0), (q_0, q_1), (q_1, q_0) e (q_1, q_1), gerando:

$$\begin{aligned} [q_0 X q_0] &\rightarrow a[q_0 X q_0][q_0 X q_0] \\ [q_0 X q_1] &\rightarrow a[q_0 X q_0][q_0 X q_1] \\ [q_0 X q_0] &\rightarrow a[q_0 X q_1][q_1 X q_0] \\ [q_0 X q_1] &\rightarrow a[q_0 X q_1][q_1 X q_1] \end{aligned}$$

- Para $\delta(q_0, \varepsilon, X) = (q_1, X)$, considerar $[q_0 X \_\_] \rightarrow [q_1 X \_\_]$ com as listas (q_0) e (q_1), gerando:

$$\begin{aligned} [q_0 X q_0] &\rightarrow [q_1 X q_0] \\ [q_0 X q_1] &\rightarrow [q_1 X q_1] \end{aligned}$$

- Para $\delta(q_1, b, X) = (q_1, \varepsilon)$:

$$[q_1 X q_1] \rightarrow b$$

- Para $\delta(q_1, \varepsilon, X) = (q_1, XX)$, considerar $[q_1 X \_\_] \rightarrow [q_1 X \_\_][\_\_ X \_\_]$ com as listas (q_0, q_0), (q_0, q_1), (q_1, q_0) e (q_1, q_1), gerando:

$$\begin{aligned} [q_1 X q_0] &\rightarrow [q_1 X q_0][q_0 X q_0] \\ [q_1 X q_1] &\rightarrow [q_1 X q_0][q_0 X q_1] \\ [q_1 X q_0] &\rightarrow [q_1 X q_1][q_1 X q_0] \\ [q_1 X q_1] &\rightarrow [q_1 X q_1][q_1 X q_1] \end{aligned}$$

- Para $\delta(q_1, b, Z_0) = (q_1, \varepsilon)$:

$$[q_1 Z_0 q_1] \rightarrow b$$

A renomeação dos símbolos não-terminais e o agrupamento das regras produz como resultado o conjunto:

$$\begin{aligned} S &\rightarrow A|B \\ A &\rightarrow aCA|aDE \\ B &\rightarrow aCB|aDF \\ C &\rightarrow aCC|aDG|G \\ D &\rightarrow aCD|aDH|H \\ F &\rightarrow b \\ G &\rightarrow GC|HG \\ H &\rightarrow GD|b|HH \end{aligned}$$

Finalmente, a eliminação de símbolos inacessíveis e inúteis resulta em:

$$\begin{aligned} S &\rightarrow B \\ B &\rightarrow aDF \\ D &\rightarrow aDH|H \\ F &\rightarrow b \\ H &\rightarrow b|HH \end{aligned}$$

ou seja, $L(G) = V(M) = \{a^i b^j | i \geq 1 \text{ e } j > i\}$. O procedimento ora apresentado é dirigido pelas transições do autômato, e por isso ele gera todas as regras e todos os símbolos não-terminais previstos pelo algoritmo, independentemente de eles serem de fato utilizados ou necessários na gramática final. Por isso, um procedimento alternativo pode ser adotado, procedimento esse que é dirigido não pelas transições do autômato, mas sim pelo próprio conjunto de regras que está sendo gradativamente construído, trazendo com isso a vantagem de evitar a geração de regras e símbolos que não sejam relevantes para a gramática resultante. Esse procedimento alternativo é apresentado a seguir.

33. Página 357, primeira linha
onde se lê:
ou seja, $L(G) = V(M) = \{a^i b^j | i \geq 1 \text{ e } j > i\}$.
leia-se:
Note-se que, exceto pela mudança de nome de alguns símbolos não-terminais, a gramática gerada no segundo procedimento é a mesma que foi gerada no primeiro procedimento.

34. Página 359
onde se lê:
"Como $vwx = \alpha_1$, então então $2 \leq |vwx| \leq 2^k$;"
leia-se:
"Como $vwx = \alpha_1$, então então $2 \leq |vwx| \leq 2^k$, ou seja, $|vwx| \leq (n-1) * 2$;"
onde se lê:
"Como $1 \leq |w|$ e $2 \leq |vwx|$, então $|vx| \geq 1$;"
leia-se:
"Como $1 \leq |w|$ e $2 \leq |vwx|$, então $|vx| \geq 1$, ou seja, v e x não podem ser ambas vazias;"

35. Página 359
acrescentar, no final do parágrafo "Através desta figura...":
"Em outras palavras, árvores com altura $i+1$ geram sentenças de comprimento máximo 2^i. Logo, se uma sentença tem comprimento mínimo (maior ou igual a) 2^i, então a árvore de derivação correspondente possuirá altura mínima (maior ou igual a) $i+1$. De fato, é necessária uma árvore com altura pelo menos $i+1$ para gerar uma sentença de comprimento 2^i e, além disso, são necessárias árvores com alturas maiores para sentenças ainda mais longas."