

Exemplo 5.14 Seja $L(G) = \{w = 0^i 1^i 2^i | i \geq 0\}$ e seja p o comprimento de bombeamento. Tomando $w = 0^p 1^p 2^p$, $|w| \geq p$, divide-se w em cinco partes, isto é, $w = uvxyz$. É necessário que $|vxy| \leq p$, logo vxy não pode conter 0 e 2 ao mesmo tempo.

Supondo que o 0 não ocorra em vxy (o caso com o símbolo 2 é análogo) e fazendo $i = 0$, tem-se que o número de 1's e 2's em $w_0 = uv^0xy^0z = uxz$ é menor que $2p$. Além disso, o número de 0's em w_0 tem que ser p , logo $w_0 \notin L(G)$ e L não é livre de contexto.

Exemplo 5.15 Seja $L(G) = \{w = 0^i 10^i 10^i | i \geq 1\}$ e seja p o comprimento de bombeamento. Tomando $w = 0^p 10^p 10^p$, $|w| \geq p$, divide-se w em cinco partes, isto é, $w = uvxyz$. É necessário que $|vxy| \leq p$ e que $|vy| > 0$. Existe dois casos a serem analisados: (1) quando vy não possui 0's; (2) quando vy possui pelo menos um 0.

No primeiro caso, se vy não possui 0's e $|vy| > 0$, então ou $v = 1$, ou $y = 1$. Assumindo que $v = 1$ (para $y = 1$ a análise é análoga), então o bombeamento produziria 1's contíguos, fazendo com que $w \notin L(G)$.

No segundo caso quando existe ao menos um 0 em vy e $|vxy| \leq p$. Assim, vxy é pequeno demais para acomodar todos os três agrupamentos de 0's presentes em $0^p 10^p 10^p$. Assim, $w = uv^0xy^0z = uxz$ possui pelo menos um bloco com p 0's e pelo menos um bloco com menos de p 0's. Logo, $uwz \notin L(G)$.

5.7 Gramáticas Regulares

Na Seção 5.4 foi definido que gramáticas regulares $\mathcal{G} = \langle \Sigma, \mathcal{L}, \mathcal{A}, \mathcal{R} \rangle$ são aquelas cujas regras de produção são da forma $A \rightarrow a$, $a \in T$ e $A \in N$ e $A \rightarrow aB$, $a \in T$ e $A, B \in N$, além de possuírem as propriedades de gramáticas tipo 2.

Desta forma, pode-se contruir uma gramática regular $G3 = \langle \Sigma_3, \mathcal{L}_3, \mathcal{A}_3, \mathcal{R}_3 \rangle$ obtida a partir da concatenação das gramáticas $G1 = \langle \Sigma_1, \mathcal{L}_1, \mathcal{A}_1, \mathcal{R}_1 \rangle$ e $G2 = \langle \Sigma_2, \mathcal{L}_2, \mathcal{A}_2, \mathcal{R}_2 \rangle$ tal que $L(G3) = \{ab | a \in L(G1), b \in L(G2)\}$. Nesta gramática tem-se que:

- $N_3 = N_1 \cup N_2$
- $T_3 = T_1 \cup T_2$
- $\mathcal{A}_3 = \mathcal{A}_1$
- \mathcal{R}_3 construído da seguinte forma:
 1. Se $A \rightarrow aB \in \mathcal{R}_1$, então $A \rightarrow aB \in \mathcal{R}_3$
 2. Se $A \rightarrow a \in \mathcal{R}_1$, então $A \rightarrow a\mathcal{A}_2 \in \mathcal{R}_3$
 3. Todas as produções de \mathcal{R}_2 estão em \mathcal{R}_3

Exemplo 5.16 Seja uma gramática G_1 , com $\mathcal{A}_1 = \{S_1\}$ cujas regras de produção são dadas por:

$$S_1 \rightarrow 0A$$

$$S_1 \rightarrow 1B$$

$$A \rightarrow 1$$

$$B \rightarrow 2$$

e G_2 , com $\mathcal{A}_2 = \{S_2\}$ cujas regras de produção são dadas por:

$$S_2 \rightarrow a$$

$$S_2 \rightarrow bC$$

$$S_2 \rightarrow aS_2$$

$$C \rightarrow a$$

As linguagem produzidas por estas gramáticas são $L(G_1) = \{01, 12\}$ e $L(G_2) = \{a, aa, aaa, \dots, ba, aba, aaba, \dots\}$. A concatenação $L(G_3) = L(G_2).L(G_2)$ resulta em $\{01a, 01aa, \dots, 01ba, 01aba, \dots, 12a, 12aa, \dots, 12ba, 12aba, \dots\}$. Para que isso aconteça, as regras de produção passam a ser:

$$S_1 \rightarrow 0A$$

$$S_1 \rightarrow 1B$$

$$A \rightarrow 1S_2$$

$$B \rightarrow 2S_2$$

$$S_2 \rightarrow a$$

$$S_2 \rightarrow bC$$

$$S_2 \rightarrow aS_2$$

$$C \rightarrow a$$

De modo análogo, pode-se contruir uma gramática regular $G3 = \langle \Sigma_3, \mathcal{L}_3, \mathcal{A}_3, \mathcal{R}_3 \rangle$ obtida a partir da união das gramáticas $G1 = \langle \Sigma_1, \mathcal{L}_1, \mathcal{A}_1, \mathcal{R}_1 \rangle$ e $G2 = \langle \Sigma_2, \mathcal{L}_2, \mathcal{A}_2, \mathcal{R}_2 \rangle$ tal que $L(G3) = L(G1) \cup L(G2) = \{w | w \in L(G1) \vee w \in L(G2)\}$. Nesta gramática tem-se que:

- $N_3 = N_1 \cup N_2 \cup \mathcal{A}_3$
- $T_3 = T_1 \cup T_2$
- $\mathcal{A}_3 = \{S_3\}$, onde S_3 é um novo símbolo não terminal
- \mathcal{R}_3 construído da seguinte forma:
 1. $\mathcal{R}_1 \in \mathcal{R}_3$ substituindo todos os S_1 do lado esquerdo da regra por S_3
 2. $\mathcal{R}_2 \in \mathcal{R}_3$ substituindo todos os S_2 do lado esquerdo da regra por S_3
 3. Remove-se as produções $S \rightarrow \Lambda$
 4. Somente se repetem as regras que contém S_1 e S_2 se estes símbolos aparecerem em algum momento no lado direito das produções

Exemplo 5.17 Utilizando as gramáticas G_1 e G_2 definidas no exemplo 5.16 tem-se

que $G_3 = G_1 \cup G_2$ é dada por:

$$\begin{aligned}
S &\rightarrow 0A \\
S &\rightarrow 1B \\
A &\rightarrow 1 \\
B &\rightarrow 2 \\
S &\rightarrow a \\
S &\rightarrow bC \\
S &\rightarrow aS_2 \\
C &\rightarrow a \\
S_2 &\rightarrow a \\
S_2 &\rightarrow bC \\
S_2 &\rightarrow aS_2
\end{aligned}$$

Por fim, se for considerada a estrela de Kleene, tem-se que $L^* = \{\Lambda\} \cup L \cup L.L \cup L.L.L \cup \dots$, isto é se $w \in L^*$ então w é obtido a partir da concatenação de zero ou mais cadeias de L , algo parecido com a definição de Σ^* . Assim, o próximo passo é definir uma gramática $G = \langle \Sigma, \mathcal{L}, \mathcal{A}, \mathcal{R} \rangle$ a partir de $G_1 = \langle \Sigma_1, \mathcal{L}_1, \mathcal{A}_1, \mathcal{R}_1 \rangle$ tal que $L(G) = L(G_1)^*$. Nesta gramática tem-se que:

- $N = N_1 \cup \mathcal{A}$
- $T = T_1$
- $\mathcal{A} = \{S\}$
- \mathcal{R} construído da seguinte forma, com $A, B \neq S_1$:
 1. Se $S_1 \rightarrow \alpha \in \mathcal{R}_1$ então $S \rightarrow \alpha \in \mathcal{R}$
 2. Se $A \rightarrow aB \in \mathcal{R}_1$ então $A \rightarrow aB \in \mathcal{R}$
 3. Se $A \rightarrow a \in \mathcal{R}_1$ então $A \rightarrow aS \in \mathcal{R}$ e $A \rightarrow a \in \mathcal{R}$
 4. $S \rightarrow \Lambda \in \mathcal{R}$

Exemplo 5.18 Seja $L(G_1) = \{a^n bc | n \geq 0\}$ construída pelo seguinte conjunto de produções:

$$S_1 \rightarrow aS_1$$

$$S_1 \rightarrow bA$$

$$A \rightarrow c$$

$L(G) = L(G_1)^*$ é dada por:

$$S \rightarrow aS$$

$$S \rightarrow bA$$

$$A \rightarrow cS$$

$$A \rightarrow c$$

$$S \rightarrow \Lambda$$

Definição 5.8 Um conjunto regular sobre um alfabeto finito Σ é construído da seguinte forma:

1. O conjunto \emptyset é um conjunto regular sobre o alfabeto Σ ;
2. O conjunto $\{\Lambda\}$ com a cadeia vazia é um conjunto regular sobre o alfabeto Σ ;
3. O conjunto $\{\sigma\}$ com $\sigma \in \Sigma$ é um conjunto regular sobre o alfabeto Σ ;
4. Se A e B são conjuntos regulares sobre o alfabeto Σ então $A \cup B$, $A.B$ e A^* são conjuntos regulares sobre o alfabeto Σ ;

O modo de representação de conjuntos regulares é conhecido como Expressão Regular, e é definido como se segue.

Definição 5.9 Uma expressão regular sobre um alfabeto finito Σ é definida como:

1. \emptyset é uma expressão regular denotando o conjunto regular \emptyset ;
2. Λ é uma expressão regular denotando o conjunto regular $\{\Lambda\}$;
3. σ é uma expressão regular denotando o conjunto regular $\{\sigma\}$;
4. Se a e b são expressões regulares denotando os conjuntos regulares A e B , então $(a + b)$ é uma expressão regular denotando $A \cup B$, (ab) é uma expressão regular denotando $A.B$, e $(a)^*$ é uma expressão regular denotando A^* ;⁵

⁵Na literatura, pode-se encontrar também a notação a^+ para denotar a expressão regular pp^* , porém esta notação não será empregada neste livro.

Em alguns casos, quando não há ambiguidade, é possível remover os parênteses. Para isto é importante ter em mente que a ordem crescente de precedência (abrangência do escopo) entre os operadores é $*$ \prec $.$ \prec $+$. Desta forma, a expressão $(0 + (1(0^*)))$ pode ser reescrita como sendo $0 + 10^*$.

Exemplo 5.19 A seguir são apresentados alguns significados de expressões regulares:

1. ab denota a linguagem regular $L(G) = \{ab\}$;
2. a^* denota a linguagem regular $L(G) = \{\Lambda, 0, 00, 000, \dots\}$;
3. $(a + b)$ denota a linguagem regular $L(G) = \{a, b\}$;
4. a^*b^* denota o conjunto de todas as cadeias de a 's seguidas de todas as cadeias de b 's e a cadeia nula;
5. $(a^*b^*)^*$ também denota o conjunto de todas as cadeias de a 's e b 's, incluindo a cadeia nula;
6. $(a + b)^*$ denota o conjunto de todas as cadeias de a 's e b 's, incluindo a cadeia nula;
7. $(a + b)^*aab$ denota o conjunto de todas as cadeias de a 's e b 's, terminando com o string aab ;
8. $(a + b)(\clubsuit + \diamondsuit + \heartsuit + \spadesuit)^*$ denota o conjunto de todas as cadeias de $\{\clubsuit, \diamondsuit, \heartsuit, \spadesuit\}^*$, começando com a ou b ;

O processo de determinação da gramática correspondente a uma expressão regular consiste em um problema de dividir para conquistar. Inicialmente divide-se a expressão regular em fragmentos para os quais seja facilmente determinada a gramática associada. Em seguida, vai se construindo a expressão original, operador por operador ($+$, $.$, $*$), e refletindo estas operações em operações entre gramáticas (\cup , $.$, $*$, respectivamente). Por exemplo, seja a expressão regular $(01 + 2)$. Esta expressão pode ser dividida em dois fragmentos, 01 e 2 , cada qual com sua gramática

associada:

$$\underbrace{\underbrace{(01)}_{G_1} + \underbrace{2}_{G_2}}_{G_1 \cup G_2}$$

onde as regras destas gramáticas são dadas por:

$$\begin{aligned} L(G_1) & : S_1 \rightarrow 0A \\ & A \rightarrow 1 \end{aligned}$$

$$L(G_2) : S_2 \rightarrow 2$$

$$\begin{aligned} L(G_1) \cup L(G_2) & : S \rightarrow 0A \\ & A \rightarrow 1 \\ & S \rightarrow 2 \end{aligned}$$

Exemplo 5.20 Determine as regras de produção associadas a expressão regular $(0 + 1 + 2)^*$.

Para a determinação desta gramática, serão criados os fragmentos:

$$\underbrace{\underbrace{\underbrace{0}_{G_1} + \underbrace{1}_{G_2}}_{G_4} + \underbrace{2}_{G_3}}_{G_5}^*_{G_6}$$

onde as regras destas gramáticas são dadas por:

$$\begin{aligned} L(G_1) & : S_1 \rightarrow 0 & L(G_5) & : S_5 \rightarrow 0 \\ & & & S_5 \rightarrow 1 \\ L(G_2) & : S_2 \rightarrow 1 & & S_5 \rightarrow 2 \\ \\ L(G_3) & : S_3 \rightarrow 2 & L(G_6) & : S \rightarrow 0S \\ & & & S \rightarrow 0 \\ L(G_4) & : S_4 \rightarrow 0 & & S \rightarrow 1S \\ & S_4 \rightarrow 1 & & S \rightarrow 1 \\ & & & S \rightarrow 2S \\ & & & S \rightarrow 2 \\ & & & S \rightarrow \Lambda \end{aligned}$$

Exemplo 5.21 Determine as regras de produção associadas a expressão regular $(00^* + 1)2$.

Para a determinação desta gramática, serão criados os fragmentos:

$$\begin{array}{c} \underbrace{\underbrace{0}_{G_1} \underbrace{0^*}_{G_2} + \underbrace{1}_{G_3}}_{G_5} 2 \\ \underbrace{\hspace{10em}}_{G_6} \\ \underbrace{\hspace{10em}}_{G_7} \end{array}$$

onde as regras destas gramáticas são dadas por:

$$\begin{array}{ll} L(G_1) : S_1 \rightarrow 0 & L(G_6) : S_6 \rightarrow 1 \\ & S_6 \rightarrow 0S_2 \\ L(G_2) : S_2 \rightarrow 0S_2 & S_2 \rightarrow 0S_2 \\ & S_2 \rightarrow \Lambda \\ & L(G_7) : S \rightarrow 1S_4 \\ L(G_3) : S_3 \rightarrow 1 & S \rightarrow 0S_2 \\ & S_2 \rightarrow 0S_2 \\ L(G_4) : S_4 \rightarrow 2 & S_4 \rightarrow 2 \\ \\ L(G_5) : S_5 \rightarrow 0S_2 \\ & S_2 \rightarrow 0S_2 \\ & S_2 \rightarrow \Lambda \end{array}$$

O processo inverso de determinação de uma expressão regular a partir de um conjunto de regras de uma gramática envolve algumas operações de uma álgebra chamada Álgebra de Kleene. Assim, sejam α, β, γ expressões regulares, tem-se as seguintes propriedades:

1. $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$
2. $\alpha.(\beta.\gamma) = (\alpha.\beta).\gamma$
3. $\alpha + \beta = \beta + \alpha$
4. $\alpha + \alpha = \alpha$
5. $\alpha.(\beta + \gamma) = (\alpha.\beta) + (\alpha.\gamma)$

$$6. (\beta + \gamma).\alpha = (\beta.\alpha) + (\gamma.\alpha)$$

$$7. \Lambda.\alpha = \alpha = \alpha.\Lambda$$

$$8. \alpha^* = \Lambda + \alpha.\alpha^*$$

$$9. \alpha^* = (\Lambda + \alpha)^*$$

Deste modo, para construir um expressão regular que denota $L(G)$ a partir de regras de produção, deve-se inicialmente associar uma linguagem a cada regra de produção. Por exemplo, as regras de produção a seguir seriam mapeadas pelas seguintes linguagens:

$$S \rightarrow aS \rightsquigarrow L(S) = \{a\}.L(S)$$

$$S \rightarrow bR \rightsquigarrow L(S) = \{b\}.L(R)$$

$$R \rightarrow aS \rightsquigarrow L(R) = \{a\}.L(S)$$

Em seguida, realiza-se a união das linguagens com mesma simbologia, obtendo-se:

$$L(S) = \{a\}.L(S) \cup \{b\}.L(R)$$

$$L(R) = \{a\}.L(S)$$

O passo seguinte é uma simplificação notacional que já remete a notação das expressões regulares. Os sinais de $\{ \}$ são omitidos, e cada linguagem $L(x)$ é substituída por uma variável X . Por fim, a operação de \cup é substituída pela operação regular homomorfica a mesma, isto é, $+$. Assim, tem-se:

$$S = aS + bR$$

$$R = aS$$

A etapa seguinte é realizar as substituições das expressões a fim se compor uma única expressão. Neste momento, começa-se a aplicar a álgebra de Kleene para compor esta única expressão. Neste caso, substituindo o valor de R e S tem-se:

$$S = aS + baS = (a + ba)S$$

Para prosseguir com o passo seguinte é importante observar uma propriedade das expressões denotativas de linguagens. Seja a expressão $X = \alpha X + \beta$, está

associada às regras de produção:

$$\begin{aligned} X &\rightarrow \beta \\ X &\rightarrow \alpha X \end{aligned}$$

que produz a linguagem $L(X) = \{\beta, \alpha\beta, \alpha\alpha\beta, \alpha\alpha\alpha\beta, \dots\} = \alpha^*\beta$. Esta observação permite concluir que $X = \alpha X + \beta$ por ser substituído por $X = \alpha^*\beta$. Esta propriedade é conhecida como lema de Arden.

Lema 5.2 [Lema de Arden] Sejam A e B linguagens de um alfabeto Σ , $\Lambda \notin A$ e uma linguagem X descrita por ela mesma. O subconjunto de Σ^* que satisfaz $X = AX + B$ é $X = A^*B$.

Desta forma, voltando à expressão $S = (a + ba)S$, e tomando $\alpha = (a + ba)$ e $\beta = \Lambda$, tem-se que $S = \alpha S + \beta = \alpha^*\beta = (a + ba)^*\Lambda = (a + ba)^*$. Como o símbolo S é o símbolo inicial da gramática G , tem-se que $L(G) = L(S)$. Portanto, $L(G) = (a + ba)^*$.

De forma análoga ao que aconteceu na seção 5.6, algumas vezes é necessário ser capaz de avaliar se uma linguagem não é regular. Nesta seção será utilizado novamente o *pumping lemma* (lema do bombeamento) para a execução desta atividade, desta vez adaptado para linguagens regulares.

Lema 5.3 [Pumping Lemma para Linguagens Regulares] Se $L(G)$ é uma linguagem regular, então existe um valor p (*pumping length*) onde, $w \in L(G)$, com $|w| \geq p$, então w pode ser dividido em três partes $w = xyz$ satisfazendo as condições:

- (a) para $i \geq 0$, $xy^iz \in L(G)$
- (b) $|y| > 0$
- (c) $|xy| \leq p$

As regras em uma gramática regular à direita são do tipo $A \rightarrow aB$, onde $A, B \in N$ e $a \in T$ (para gramáticas regulares à esquerda o raciocínio é análogo). Assim, para produzir uma cadeia w , com $|w| \geq p$ são necessárias p derivações. Seja $\#(\mathcal{R})$ o número de regras existentes em G e assumamos o comprimento de bombeamento

dado por $p = \#(\mathcal{R}) + 1$. Desta forma, para produzir w com $|w| \geq p$ serão necessárias ao menos p derivações, isto é, pelo menos $\#(\mathcal{R}) + 1$ derivações. Ora, se existem $\#(\mathcal{R})$ regras em G é porque ao menos uma regra R foi repetida (*pigeonhole principle*).

A derivação de w , com a regra R sendo repetida, é realizada como se segue:

$$\underbrace{w_1 \Rightarrow \cdots \Rightarrow w_{j-1}}_x \xRightarrow{R} \underbrace{w_j \Rightarrow \cdots \Rightarrow w_{k-1}}_y \xRightarrow{R} \underbrace{w_k \Rightarrow \cdots \Rightarrow w_n}_z$$

Assim, a cadeia que pode ser repetida fica cercada por duas outras cadeias, produzindo a condição (a), xy^iz , $i \geq 0$. A condição (b) é facilmente satisfeita uma vez que $j \neq k$ e portanto $|y| > 0$. Além disso, como w_k é a cadeia imediatamente antes da regra R ser aplicada novamente, tem-se que $|w_k| = |xy| \leq \#(\mathcal{R}) + 1$, ou seja, $|xy| \leq p$, atendendo a condição (c).

Exemplo 5.22 Seja $L(G) = \{w = 0^n 1^n | n > 0\}$ e tomando o comprimento de bombeamento p , tem-se que $w = 0^p 1^p$ e $|w| \geq p$. Dividindo w em três partes xyz , tem-se que considerar três hipóteses distintas sobre y : (1) y só possui 0's; (2) y só possui 1's; (3) y possui 0's e 1's.

No primeiro caso onde y só possui 0's, então x também só possui 0's. Desta forma xy^2z tem que possuir mais 0's do que 1's, e portanto, $xy^2z \notin L(G)$. O caso onde y só possui 1's é análoga a situação ora descrita.

No caso em que y possui 0's e 1's, então xy^2z vai possuir uma sequência $0^i 1^k 0^i 1^k$, com $j > 0$ e $k > 0$. Logo $xy^2z \notin L(G)$.

Exemplo 5.23 Seja $L(G) = \{w = 0^i 1^j | i > j\}$ e tomando o comprimento de bombeamento p , pode-se escolher $w = 0^{p+1} 1^p$ e $|w| \geq p$. Em seguida, divide-se w em três partes xyz . Como uma das condições do pumping lemma é que $|xy| \leq p$, tem-se que y é composto somente por 0's. Fazendo o bombeamento de y tem-se que a cadeia xy^2z também possuirá mais 0's do que 1's, o que em princípio não contradiz a linguagem $L(G)$.

Contudo $xy^iz \in L(G)$, até mesmo para $i = 0$. Assim, fazendo $w_0 = xy^0z = xz$ se reduz a quantidade de 0's em w_0 . Lembrando que w possui apenas um 0 a mais que 1, então xz não pode ter mais 0's que 1's, e portanto $w_0 \notin L(G)$.

Referências Bibliográficas

- [1] HOPCROFT, J. E., *Theory os Machines and Compuatations*, chapter An n log n algorithm for minimizing states in a finite automaton, Academic Press, pp. 189–196, 1971.
- [2] MONTEIRO, A. A., PAULO, J. D. S., *Aritmética Racional*. Lisboa, Livraria Avelar Machado, 1945.
- [3] IFRAH, G., *Os Números: a história de uma grande invenção*. São Paulo, Globo S.A., 2005.
- [4] DEDEKIND, R., *Was sind und was sollen die Zahlen*, v. 3, *Gesammelte Mathematische Werke*. New York, Chelsea Publishing Company, 1969. pp. 335-391.
- [5] GÖDEL, K., *Collected Works*, v. 2, *Gesammelte Mathematische Werke*. Oxford, Oxford University Press, 1990.
- [6] ISRAEL, D., “Reflections on Gödel’s and Gandy’s Reflection on Turing’s Thesis”, *Minds and Machines*, v. 12, n. 2, pp. 181–201, 2002.
- [7] SOBRINHO, J. Z., “Aspectos da Tese de Church-Turing”, *Revista Matemática Universitária - USP*, v. 1, n. 6, pp. 1–23, 1987.
- [8] MCDERMOTT, D., “Artificial Intelligence Meets Natural Stupidity”, *SI-GART Newsletter*, v. 57, pp. 4–9, April 1976.
- [9] SETTI, M. D. O. G., *O Processo de Discretiza çã o do Raciocínio Matemático na Tradu çã o para o Raciocínio Computacional*, Report, Universidade Federal do Paraná, 2009.

- [10] GUERREIRO, G., “A Vanguarda Matemática e os Limites da Razão”, *Scientific American Brasil*, v. 5, n. 12, pp. 39–56, 2007. Coleção Gênios da Ciência.
- [11] SMULLYAN, R., *Gödel’s Incompleteness Theorems*. Oxford University Press, 1992.
- [12] GÖDEL, K., *The Undecidable*, chapter On Formally Undecidable Propositions of Principia Mathematica and Related Systems, New York, Raven Press, pp. 5–38, 1965.
- [13] WHITEHEAD, A. N., RUSSELL, B., *Principia Mathematica*. Londres, Cambridge University Press, 1913.
- [14] NAGEL, E., NEWMAN, J. R., *Gödel’s Proof*. USA, Routledge, 1989.
- [15] GÖDEL, K., “Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme - On Formally Undecidable Propositions of Principia Mathematica and Related Systems”, *Kurt Gödel: Collected Works*, v. 1, n. 1, pp. 144–195, 1986. Tradução para o inglês por Martin Hirzel. www.research.ibm.com/people/h/hirzel/papers/canon00-goedel.pdf [capturado em 13 de agosto de 2011].
- [16] MELO, A. C. V. D., SILVA, F. S. C. D., *Modelos Clássicos de Computação*, Coleção Schaum. São Paulo, Thomson, 2006.
- [17] KUBRUSLY, R. D. S., *Uma viagem informal ao Teorema de Gödel ou (O preço da matemática é o eterno matemático)*, Report, Universidade Federal do Rio de Janeiro, 2007. IM/UFRJ.
- [18] SMULLYAN, R., *Recursion Theory for Metamathematics*. Oxford University Press, 1993.
- [19] SMULLYAN, R., *What’s the Name of This Book*. Penguin Books, 1978.
- [20] SMULLYAN, R., *Forever Undecided: A Puzzle Guide to Gödel*. Oxford University Press, 1987.
- [21] GOLDSTEIN, R., *Incompleteness: The Proof and Paradox of Kurt Gödel*. W. W. Norton Company, Inc., 2005.

- [22] HOFSTADTER, D. R., *Gödel, Escher e Bach: an Eternal Golden Braid*. Nova Iorque, Basic Books, 1979.
- [23] Rodríguez-Consuegra, F. A. (ed.), *Kurt Gödel - Unpublished Philosophical Essays*. Berlin, Birkhäuser Verlag, 1995.
- [24] Feferman, S., *et al.* (eds.), *Kurt Gödel - Collected Works*, v. I, II, III. New York, Oxford University Press, 1986.
- [25] WANG, H., *Reflections on Kurt Gödel*. Cambridge, Massachusetts, The MIT Press, 1988.
- [26] SUPPES, P., *Axiomatic Set Theory*. New York, Dover, 1972.
- [27] LIPSCHUTZ, S., *Teoria Elementar dos Conjuntos*, Cole çã o Schaum. Sã o Paulo, McGraw-Hill do Brasil, 1990.
- [28] SUPPES, P., *Axiomatic Theory Set*. USA, Van Nostrand Company Inc., 1960.
- [29] MELLO, F. L. D., CARVALHO, R. L. D., “Knowledge Geometry”, *Journal of Information and Knowledge Management*, v. 14, pp. 1550028, 2015.
- [30] STOLL, R. R., *Set Theory and Logic*. USA, Dover Publications Inc., 1961.
- [31] MIRAGLIA, F., *Teoria dos Conjuntos: um mínimo*. Brasil, Edusp - Editora da Universidade de São Paulo, 1992.
- [32] HALMOS, P., *Teoria Ingênua dos Conjuntos*. Brasil, Edusp - Editora da Universidade de São Paulo, 1970.
- [33] GRATZER, G., *Universal Algebra*. USA, Van Nostrand Company Inc., 1968.
- [34] COHN, P. M., *Universal Algebra*. USA, Harper and Row, 1965.
- [35] GALLIER, J. H., *Logic for Computer Science*. USA, John Wiley and Sons Inc., 1987.
- [36] PREPARATTA, F. P., YEH, R. T., *Introduction to Discreate Structures for Computer Science and Engineering*. USA, Addison-Wesley Publishing Company, 1973.

- [37] SHOENFIELD, J. R., *Degrees of Unsolvability*. North-Holland Publishing Company, 1971.
- [38] BRAINERD, W. S., LANDWEBER, L. H., *Theory of Computation*. USA, John Wiley and Sons, 1974.
- [39] EILENBERG, S., ELGOT, C., *Recursiveness*. New York: Academic Press, 1970.
- [40] CARVALHO, R. L. D., OLIVEIRA, C. M. G. M. D., *Modelos de Computação e Sistemas Formais*, 11^a Escola de Computação. Rio de Janeiro, Universidade Federal do Rio de Janeiro, 1998.
- [41] BRAINERD, W. S., LANDWEBER, L. H., *Theory of Computation*. John Wiley & Sons, 1974.
- [42] KLEENE, S. C., *Introduction to Metamathematics*. D. Van Nostrand Company, Inc., 1952.
- [43] Rogers Jr., H., *Theory of Recursive Functions and Effective Computability*. USA, McGraw-Hill Book Company, 1967.
- [44] DAVIS, M., *Computability and Unsolvability*. New York, Dover, 1983.
- [45] BOOLOS, G. S., JEFFREY, R. C., *Computability and Logic*. Cambridge University Press, 1974.
- [46] MARTIN DAVIS, R. S., WEYUKER, E. J., *Computability Complexity and Languages*. New York, Academic Press, 1994.
- [47] MALLOZZI, J. S., LILLO, N. J. D., *Computability with Pascal*. New Jersey, Prentice-Hall, Inc., 1984.
- [48] TURING, A. M., *The Undecidable*, chapter On Computable Numbers, with an Application to the Entscheidungsproblem, New York, Raven Press, pp. 115–151, 1965.
- [49] ELGOT, C. C., ROBINSON, A., “Random-access stored-program machines, an approach to programming languages”, *Journal of the ACM*, v. 11, pp. 365–399, 1964.

- [50] PREPARATTA, F. P., YEH, R. T., *Introduction to Discrete Structures for Computer Science and Engineering*. Addison-Wesley Publishing Company, 1973.
- [51] HENNIE, F., *Introduction to Computability*. Addison-Wesley Publishing Company, 1977.
- [52] NELSON, R. J., *Introduction to Automata*. USA, Jonh Wiley & Sons, Inc., 1968.
- [53] CARVALHO, R. L. D., *Máquinas, Programas e Algoritmos*, 2^a Escola de Computação. Campinas, Universidade Estadual de Campinas, 1981.
- [54] MENDELSON, E., *Introduction to Mathematical Logic*, Cole Mathematics Series. 3 ed. The Wadsworth and Brooks, 1987.
- [55] MANIN, Y. I., *A Course in Mathematical Logic*, Graduate Texts in Mathematics 53. 1 ed. Springer-Verlag, 1977.
- [56] HOMER, S., SELMAN, A. L., *Computability and Complexity Theory*, Texts in Computer Science. 2 ed. Springer, 2011.
- [57] CUTLAND, N. J., *Computability: An introduction to recursive function theory*. Cambridge University Press, 1980.
- [58] SIPSER, M., *Introduction to The Theory of Computation*, Course Technology Series. 2 ed. Thomson, 2006.
- [59] WALTER CARNIELLI, R. L. E., *Computability: computable functions, logic and the foundations of mathematics*. Belmont, Wadsworth and Brooks, 1989.
- [60] TARSKI, A., *Logic, semantics, metamathematics*, chapter Fundamental concepts of the methodology of the deductive sciences, London, Oxford at the Clarendon Press, pp. 60–109, 1969.
- [61] ADAM YOUNG, M. Y., *Malicious Cryptography: Exposing Cryptovirology*. John Wiley and Sons Inc., 2004.

- [62] BONFANTE, G., KACZMAREK, M., MARION, J.-Y., *A Classification of Viruses through Recursion Theorems*, volume 4497 of Lecture Notes in Computer Science. 2 ed. CiE 2007, 2007.
- [63] MACHTEY, M., YOUNG, P., *An Introduction to the General Theory of Algorithms*. New York, North Holland, 1978.
- [64] ROBINSON, J. A., “A machine oriented logic based on the resolution principle”, *J. Assoc. Comput.*, v. 12, pp. 23–41, 1965.
- [65] ROBINSON, J. A., “Automatic deduction with hyper-resolution”, *Internat. J. Comput. Math*, v. 1, pp. 227–234, 1965.
- [66] GILMORE, P. C., “A proof method for quantification theory: Its justification and realization”, *IBM Journal of Research and Development*, v. 4, n. 1, pp. 28–35, 1960.
- [67] DAVIS, M., PUTNAM, H., “A computing procedure for quantification theory”, *Journal of the ACM*, v. 7, n. 3, pp. 201–215, 1960.
- [68] CHANG, C.-L., LEE, R. C.-T., *Symbolic Logic and Mechanical Theorem Proving*. Academic Press Inc, 1973.
- [69] KLEENE, S. C., *Mathematical Logic*. Wiley, 1967.
- [70] HILBERT, D., ACKERMANN, W., *Prinmciples of Mathematical Logic*. Chelsea, 1950.
- [71] MCCAWLEY, J. D., *Everything That Linguists Have Always Wanted To Know About Logic*. 2 ed. The University of Chicago Press, 1993.
- [72] SUPPES, P., *Introduction to Logic*. D. van Nostrand, 1966.
- [73] RUSSELL, B., *A Filosofia do Atomismo Lógico*, *Lógica e Conhecimento*. 1 ed. Abril Cultural, 1974. (Os Pensadores, 42).
- [74] RUSSELL, B., *Significado e Verdade*. 1 ed. Zahar, 1978.
- [75] POPPER, K. R., *A Lógica da Pesquisa Científica*. 2 ed. Cultrix, 1974.

- [76] LAKATOS, I., , MUSGRAVE, A., *A Crítica e o Desenvolvimento do Conhecimento*. 1 ed. EDUSP, Cultrix, 1979. Tradução: M. O. Caiado.
- [77] WANG, H., *From Mathematics to Philosophy*. 1 ed. Cultrix, 1974.
- [78] GREEN, C. C., *The Application of Theorem Proving to Question-Answering Systems*. Ph.D. dissertation, Stanford, June 1969. AI Project MEMO AI-96.
- [79] LOVELAND, D. W., *Automated Theorem Proving: a Logical Basis*. 1 ed. North Holland, 1978.
- [80] HUGHES, G. E., LONDEY, D. G., *The Elements of Formal Logic*. USA, Methuen and Co Ltd, 1965.
- [81] BOOK, R. V., OTTO, F., *String-Rewriting Systems*. USA, Springer-Verlag, 1993.
- [82] HOPCROFT, J. E., ULLMAN, J. D., *Introduction to Automata Theory, Language and Computation*. USA, Addison-Wesley Publishing Company, 1979.
- [83] HOPCROFT, J. E., ULLMAN, J. D., *Formal Languages and their Relation to Automata*. USA, Addison-Wesley Publishing Company, 1969.
- [84] CURRY, H. B., *Foundations of Mathematical Logic*. New York, Academic Press, 1977.
- [85] SMULLYAN, R., *Theory of Formal Systems*. USA, Princeton, 1961.
- [86] BERSTEL, J., BOASSON, L., CARTON, O., *et al.*, *Handbook of Automata: from Mathematics to Applications*, chapter Minimization of automata, European Mathematical Society, pp. 189–196, 2010.
- [87] BEAL, M. P., CROCHEMORE, M., “Minimizing incomplete automata”, *Workshop on Finite State Methods and Natural Language Processing*, , september 2008. Ispra.
- [88] VALMARI, A., LEHTINEN, P., “Efficient minimization of DFAs with partial transition”, *Proc. 25th Symp. Theoretical Aspects of Comp. Sci.*, v. 08001, pp. 645–656, 2008. S. Albers and P. Weil, editors.

- [89] PAPADONIKOLAKIS, M., BOUGANIS, C.-S., CONSTANTINIDES, G.,
“Performance comparison of GPU and FPGA architectures for the SVM training problem”, *IEEE International Conference on FieldProgrammable Technology*, pp. 388–391, 2009.
- [90] MU, S., WANG, C., LIU, M., *et al.*, “Evaluating the potential of graphics processors for high performance embedded computing”, *Proc. IEEE Design, Automation and Test in Europe Conference and Exhibition*, pp. 709–714, 2011.
- [91] KAI HWANG, F. A. B., *Computer Architecture and Parallel Processing*. McGraw-Hill, 1984.
- [92] AGARWAL, P., KRISHNAN, S., MUSTAFA, N., *et al.*, *Algorithms - ESA 2003, Lecture Notes in Computer Science*, chapter Streaming Geometric Optimization Using Graphics Hardware, Springer Berlin-Heidelberg, pp. 115–151, 2003.
- [93] TANENBAUM, A. S., *Organização Estruturada de Computadores*. 3 ed. Prentice Hall do Brasil, 1997.
- [94] BACKUS, J., “Can Programming be Liberated from von Neumann Style? A functional style and its algebra of program”, *ACM Turing Award Lecture, Communications of the ACM*, v. 21, n. 8, pp. 613–641, 1978.
- [95] OWENS, J. D., LUEBKE, D., GOVINDARAJU, N., *et al.*, “A Survey of General-Purpose Computing on Graphics Hardware”, *Eurographics 2005, State of the Art Reports*, pp. 21–51, 2005.
- [96] GUSTAFSON, J. L., “Reevaluating Amdahl’s law”, *Communications of the ACM*, v. 5, n. 31, pp. 532, 1988.
- [97] HANDLER, W., *Parallel Processing Systems, an advanced course*, chapter Innovative computer architecture - how to increase parallelism but not complexity, Cambridge University Press, pp. 1–41, 1982.
- [98] LOBUR, J., NULL, L., *The Essentials of Computer Organization And Architecture*. Jones and Bartlett Pub, 2006.

- [99] LEWIS, H. R., PAPADIMITRIOU, C. H., *Elements of the Theory of Computation*. 2 ed. New York, Prentice-Hall, 1998.
- [100] DUNNE, P., *Computability Theory: Concepts and Applications*. Ellis Horwood, 1991.
- [101] AARONSON, S., “NP-complete Problems and Physical Reality”, *ACM SIGACT News*, , march 2005. Complexity Theory Column 46.