

Capítulo 5

Sistemas Formais

A Informática é a ciência do artificial por excelência e, mais ainda, é a ciência do formal: não existe na história da humanidade nenhum profissional que tenha se comportado de modo mais formal. Os computadores seguem fielmente regras, e não admitem exceções; é preciso especificar, codificar, digitar, depurar e, mesmo assim, o programa pode não funcionar simplesmente porque foi trocado um ‘0’ (zero) por ‘O’ , ou ‘a’ por ‘A’. Depois de semanas buscando erros lógicos, alguém de fora, por trás dos ombros do programador diz: **“Claro que não funciona: você usou ‘1’ e não ‘l’.”**, e então descobre-se que o engano era de natureza apenas formal. O grande objetivo dos sistemas amigáveis é a eliminação dos erros formais.

Explicar ou definir formal é uma tarefa difícil e, se for buscado o seu significado em um dicionário, encontra-se que:

- (0) Formal - do latim formale - relativo a forma.
- (1) Forma - do latim forma - Configuração exterior dos corpos; disposição das partes de um corpo, aparência; feitio de um objeto; modelo; norma.
- (2) Forma - do latim forma - molde dentro do qual ou sobre o qual se forma qualquer coisa que toma o feitio desse molde.

Depois desta consulta, descobre-se que “formal” é definido através de “forma” e o problema é saber qual a escolha a ser feita: se (1) ou (2), notando-se que há di-

ferença fonética, ainda que não gráfica entre elas: em (1) a letra “o” é pronunciada aberta e em (2) fechada.

Então pode-se usar (1) e (2) para melhorar o entendimento de formal. Pois o próprio conceito definido em (1) deve se encaixar com a “forma” (com “o” fechado) de um grupo social.

A matemática é considerada como a mais formal das ciências sendo a linguagem formal utilizada pelas outras, pois todos os resultados são baseados em regras e apresentados por fórmulas. No entanto os formalistas são apenas um grupo dentre os matemáticos, tendo existido sérias controvérsias com respeito à validade do enfoque formal. Na realidade, a maioria dos matemáticos desenvolve seus resultados dentro de um espírito informal e intuitivo, mais geométrico que algébrico; e quando algébrico, se analisado de forma mais rigorosa, pouco formal. De qualquer modo, a mais formal das correntes em matemática tem feito e faz concessões ao informal. No que se segue será apresentado que o conceito de formal a ser adotado é extremamente exigente; pode-se dizer que o formal de que se necessita é também mecânico, completamente dissociado de qualquer intuição ou fatores de natureza cognitiva.

5.1 Sistemas Formais

Nesta seção será definido o que se entende por um sistema formal. Para isto é preciso esclarecer o que significa alfabeto e palavra, pois o conceito de formal - e, particularmente sistemas formais - depende da definição e conhecimentos básicos sobre representação gráfica de símbolos e a fixação de critérios particulares de sua aceitação ou definição (recomenda-se ao leitor uma releitura da Seção 2.7). Dá-se o nome de letra a todo sinal gráfico satisfazendo os seguintes critérios:

1. As letras devem possuir uma estrutura espacial que facilite sua reprodução e reconhecimento. Por exemplo: $\square, \triangle, |, *$. Como contra-exemplo o leitor pode imaginar figuras complicadas como rubricas pessoais, tais como $b\ddot{q}$.
2. As letras devem possuir uma estrutura que impossibilite decomposições horizontais. Assim, “ $||$ ” não seria uma escolha apropriada, pois é composta

horizontalmente de dois sinais iguais (“|” e “|”); no entanto, “-” e “==” seriam escolhas adequadas - apesar de poderem ser decompostas verticalmente.

3. Para a construção de alguns sistemas formais, existe a necessidade de um suprimento infinito de letras, assim deve-se exigir que elas possam ser produzidas de modo uniforme.

Os elementos de $\Sigma_1 = \{*, |\}$ e $\Sigma_2 = \{\square, \triangle, \odot\}$ satisfazem os critérios (a), (b) e (c) recém descritos. *Alfabetos* são conjuntos recursivos de letras.

*Expressões, palavras*¹ ou *cadeias* são seqüências de letras justapostas horizontalmente, tendo seus limites claramente identificados por interespaço separador com mesma função que o espaço em branco na escrita convencional. Assim, $\square\triangle$ e $\square\odot\square\triangle$ são expressões no alfabeto Σ_2 .

Alguns autores definem uma cadeia u em um alfabeto Σ como uma função $u : \{1 \cdots n\} \rightarrow \Sigma$, onde n é o comprimento de u . Assim, uma cadeia u pode ser escrita como $a_1a_2 \cdots a_n$, onde a_i é a i -ésima letra. A cadeia de comprimento 0 é chamada **cadeia nula** e denotada por Λ .

A justaposição de duas cadeias é chamada concatenação. Sejam $u : \{1 \cdots m\} \rightarrow \Sigma_1$ e $v : \{1 \cdots n\} \rightarrow \Sigma_2$, então $u \frown v$ (ou simplesmente uv) é definida como:

$$u \frown v : \{1 \cdots m+n\} \rightarrow \Sigma_1 \cup \Sigma_2$$

$$u \frown v = \begin{cases} u(i) & \text{se } 1 \leq i \leq m \\ v(i-m) & \text{se } m+1 \leq i \leq m+n \end{cases}$$

O produto de dois alfabetos é dado pela combinação de pares feita entre seus elementos. Por exemplo, se $\Sigma_1 = \{*, |\}$ e $\Sigma_2 = \{\square, \triangle, \odot\}$, tem-se que $\Sigma_1 \times \Sigma_2 = \{*\square, *\triangle, *\odot, |\square, |\triangle, |\odot\}$, e ainda que $\Sigma_1 \times \Sigma_2 \neq \Sigma_2 \times \Sigma_1$.

¹Alguns autores mais rigorosos consideram que palavras são expressões que respeitam determinados critérios explícitos para sua formação. Desta forma pode-se explicitar que o critério de formação para as palavras em Σ_1 seja: *as únicas palavras são as expressões onde não apareçam mais que duas ocorrências sucessivas de ‘*’ e não menos que duas de ‘|’ em seqüência*. Utilizando-se este critério, conclui-se (informalmente) que, $**|*$ é uma palavra, e que $**|*$ não é, pois possui menos que duas ocorrências sucessivas de “|”.

A exponenciação de um alfabeto é dada por:

$$\begin{aligned}\Sigma^0 &= \{\Lambda\} \\ \Sigma^1 &= \Sigma \\ \Sigma^n &= \Sigma^{n-1} \times \Sigma\end{aligned}$$

observe que Σ^n é o conjunto de todas as cadeias de símbolos de Σ com comprimento n .

Por fim, se Σ for um alfabeto, denota-se por Σ^* o conjunto de todas as cadeias de Σ , incluindo a cadeia nula, isto é, $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots \cup \Sigma^n \cup \dots$. Segue-se que uma linguagem em Σ é qualquer subconjunto de Σ^* . Desta forma, se Σ é um alfabeto, então qualquer conjunto de expressões em Σ é uma *linguagem* em Σ . Assim, por exemplo, $\mathcal{L} = \{\square, \square\Delta, \square\odot, \Delta\}$ é uma linguagem em Σ .

Definição 5.1 Um sistema formal \mathcal{F} é uma quádrupla $\langle \Sigma, \mathcal{L}, \mathcal{A}, \mathcal{R} \rangle$, onde:

Σ – é um alfabeto.

\mathcal{L} – é um conjunto recursivo em Σ , chamado de linguagem do sistema formal.

\mathcal{A} – é um subconjunto recursivo de \mathcal{L} , chamado de *axiomas*

\mathcal{R} – é um conjunto recursivo de relações em \mathcal{L}

Exemplo 5.1 Seja um sistema formal, onde o alfabeto, as palavras, os axiomas e as relações estejam definidas a seguir:

$$\begin{aligned}\Sigma &= \{ |, * \}, \mathcal{L} = \{ \Sigma^* \}, \mathcal{A} = \{ |, * \}, \mathcal{R} = \{ r_1, r_2 \}, \text{ onde :} \\ r_1 &= \{ \langle x |, x* \rangle \mid x \in \Sigma^* \} \\ r_2 &= \{ \{ \langle x | *, x* \rangle \mid x \in \Sigma^* \} \cup \\ &\quad \{ \langle x | **, x* \rangle \mid x \in \Sigma^* \} \cup \\ &\quad \{ \langle x*, x \rangle \mid x \in \Sigma^* \} \}\end{aligned}$$

Definição 5.2 Seja $\mathcal{F} = \langle \Sigma, \mathcal{L}, \mathcal{A}, \mathcal{R} \rangle$ um sistema formal, $\Gamma \subseteq \mathcal{L}$. Uma dedução de α a partir de Γ em \mathcal{F} é uma seqüência $\alpha_1, \alpha_2, \dots, \alpha_n$ de palavras de \mathcal{L} , tal que:

1. α_n é α ; e

2. Para todo $j, 1 \leq j < n, \alpha_j \in \Gamma \cup \mathcal{A}$, ou existem $\alpha_{j_1}, \dots, \alpha_{j_k}, j_i \in \{1, \dots, j-1\}, 1 \leq i \leq k$ tais que $\langle \alpha_{j_1}, \dots, \alpha_{j_k}, \alpha_j \rangle \in r$ com $r \in \mathcal{R}$.

Se existir uma dedução de α a partir de Γ diz-se que α é dedutível a partir de Γ em \mathcal{F} . Isto é denotado por $\Gamma \vdash_{\mathcal{F}} \alpha$.

Exemplo 5.2 No sistema formal do exemplo 5.1 uma dedução de $*|$ é:

$$\begin{array}{l} | \quad (\in \mathcal{A}) \\ * \quad (\langle |, * \rangle \in r_1) \\ || \quad (\langle *, || \rangle \in r_2) \\ |* \quad (\langle ||, |* \rangle \in r_1) \\ *| \quad (\langle |*, *| \rangle \in r_2) \end{array}$$

portanto a seqüência $*, ||, |*, *|$ é uma dedução de $*|$ onde $\Gamma = \emptyset$, assim $\emptyset \vdash *|$.

Existem várias considerações que devem ser feitas com respeito às componentes de um sistema formal.

Σ — Os alfabetos dos sistemas formais podem conter vários tipos de letras, assim em geral devem ser especificados tais tipos de modo não ambíguo. Os tipos de letras correspondem a modos de construir as palavras da linguagem \mathcal{L} . Assim na definição de Σ pode-se ter:

$$\Sigma = \Sigma_1 \cup \Sigma_2 \cup \dots \cup \Sigma_k$$

Por exemplo, nos dialetos formalizáveis oriundos de simplificações das linguagens naturais tem-se letras para representar as diversas categorias gramaticais. É frequente se incluir no alfabeto letras que são auxiliares (variáveis sintáticas) para representar subclasses da linguagem \mathcal{L} .

\mathcal{L} — A definição da linguagem pode já necessitar de um outro sistema formal, ou notações informais para sua especificação detalhada. Note que a exigência de ser a linguagem um conjunto recursivo não é suficiente quando o sistema formal visa aplicações. As operações que são utilizadas para os critérios de construção dos componentes da linguagem devem ser de baixa complexidade.

- \mathcal{A} – Existem várias maneiras de se especificar os axiomas de um sistema formal. Se o conjunto for finito basta uma simples enumeração, no caso de um conjunto infinito pode-se utilizar um outro sistema formal para sua especificação. É frequente na literatura a utilização de *axiomas esquemas* que são bastante difíceis de serem efetivos nas aplicações computacionais.
- \mathcal{R} – A definição das regras de inferência é a parte mais complicada de um sistema formal. As regras de inferência devem ser relações recursivas definidas na linguagem \mathcal{L} podendo, portanto, ser bastante complexas para definir e para serem efetivamente utilizadas. Nos exemplos que serão apresentados se discutirá a complexidade de tais regras.

Estas observações devem ser levadas em conta quando se deseja ter um sistema formal que possa ser efetivamente utilizado. O desenvolvimento de critérios para a escolha de um sistema formal entre alternativas de formalização é uma necessidade pouco estudada. Deve-se acrescentar que os sistemas formais foram inventados com o objetivo de se *mecanizar* a solução de problemas, principalmente na matemática tradicional, e que o *agente* executante das deduções era inicialmente imaginado como um ser humano treinado como matemático. Hoje deve-se dirigir os esforços na construção de sistemas formais cujos *agentes* são de outra natureza, muitas vezes uma máquina ou sistema de programação.

Na especificação de regras de inferência escreve-se:

$$\frac{\alpha_1, \alpha_2, \dots, \alpha_l}{\beta}, \text{ ou } \alpha_1, \alpha_2, \dots, \alpha_l \rightarrow \beta$$

significando que $\langle \alpha_1, \alpha_2, \dots, \alpha_l, \beta \rangle \in \mathcal{R}$.

No caso em que $\Gamma \subseteq \mathcal{L} = \emptyset$, diz-se que α é um teorema em \mathcal{F} . O conjunto de teoremas de um sistema formal \mathcal{F} é denotado por $\mathcal{T}_{\mathcal{F}}$, ou simplesmente por \mathcal{T} , quando é claro a que sistema formal pertencem os teoremas. O conjunto de teoremas de um sistema formal inclui os axiomas, ou seja, todo axioma é um teorema. Em geral, o conjunto de teoremas é maior que o conjunto de axiomas, mas é claro que as regras de um sistema formal podem não ser aplicáveis, e neste caso o conjunto de teoremas é composto apenas dos axiomas. O conjunto de teoremas pode ser igual à

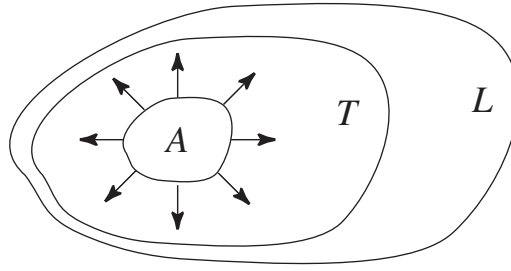


Figura 5.1: Derivação de Teoremas.

linguagem, e neste caso dizemos que o sistema formal é inconsistente². A figura 5.1 mostra a relação entre axiomas, teoremas e linguagem.

Existem três tipos básicos de sistemas formais: geradores, reconhecedores e transdutores. Os geradores são sistemas formais cujo objetivo é produzir cadeias, somente a partir dos axiomas. Os reconhecedores admitem outras cadeias como entradas e verificam se tais cadeias são adequadas, sem produzir cadeias de saída. Por fim, os transdutores transformam cadeias de entrada em cadeias de saída.

A seguir serão apresentadas algumas variações dos tipos básicos, que serão importantes para o entendimento de questões a serem discutidas nos próximos capítulos.

5.2 Sistemas de Produções de Post

Definição 5.3 Um sistema de produção de Post (*SPP*) \mathcal{S} é um sistema formal $\langle \Sigma, \mathcal{L}, \mathcal{A}, \mathcal{R} \rangle$, onde:

Σ É um alfabeto consistindo de dois subconjuntos disjuntos N e T , chamados de alfabeto não terminal e terminal, respectivamente, onde $N = \{\square | i > 0\}$

\mathcal{L} É o conjunto Σ^*

\mathcal{A} É um sub-conjunto de Σ^* , e é dito ser o conjunto de palavras de partida.

²Existem várias concepções da noção de inconsistência. Esta, em particular, é chamada de inconsistência absoluta [80]

\mathcal{R} É um conjunto de relações binárias em \mathcal{L} , que são chamadas de regras de produção.

Cada regra é da forma:

$$x_0 \boxed{i_1} x_1 \boxed{i_2} \cdots x_{n-1} \boxed{i_n} x_n \rightarrow y_0 \boxed{j_1} y_1 \boxed{j_2} \cdots y_{k-1} \boxed{j_k} y_k$$

,onde $i_1, i_2, \dots, i_n \in \mathbb{N}$, e $j_1, j_2, \dots, j_k \in \{i_1, i_2, \dots, i_n\}$

e $x_0, x_1, \dots, x_n, y_0, \dots, y_k \in (\Sigma - N)^*$

Definição 5.4 Seja $\mathcal{S} = \langle \Sigma, \mathcal{L}, \mathcal{A}, \mathcal{R} \rangle$ um *SPP* e $\alpha \in \mathcal{L}$ uma palavra.

(a) Diz-se que uma regra:

$$x_0 \boxed{i_1} x_1 \boxed{i_2} \cdots x_{n-1} \boxed{i_n} x_n \rightarrow y_0 \boxed{j_1} y_1 \boxed{j_2} \cdots y_{k-1} \boxed{j_k} y_k$$

é aplicável a α se e somente se:

$$\alpha = x_0 z_{i_1} x_1 z_{i_2} \cdots x_{n-1} z_{i_n} x_n$$

,onde $z_{i_1}, z_{i_2}, \dots, z_{i_n} \in \Sigma^*$ e se $i_t = i_s$ então $z_{i_t} = z_{i_s}$

(b) Se uma regra:

$$x_0 \boxed{i_1} x_1 \boxed{i_2} \cdots x_{n-1} \boxed{i_n} x_n \rightarrow y_0 \boxed{j_1} y_1 \boxed{j_2} \cdots y_{k-1} \boxed{j_k} y_k$$

é aplicável a uma palavra $\alpha = x_0 z_{i_1} x_1 z_{i_2} \cdots x_{n-1} z_{i_n} x_n$ então o resultado da aplicação é a palavra

$$y_0 z_{j_1} y_1 z_{j_2} y_2 \cdots y_{k-1} z_{j_k} y_k$$

(c) Diz-se $\alpha \Rightarrow_{\mathcal{S}} \beta$ se β foi obtida a partir da palavra α pela aplicação de uma regra $r \in \mathcal{R}$.

(d) Diz-se que β é derivável em \mathcal{S} a partir de α , o que denota-se por $\alpha \Rightarrow_{\mathcal{S}}^* \beta$, se e somente se $\beta = \alpha$ ou existem $\beta_1, \beta_2, \dots, \beta_l$ tais que $\beta_l = \beta$ e para todo i , $1 \leq i < l$ $\beta_i \Rightarrow_{\mathcal{S}} \beta_{i+1}$

(e) A linguagem gerada por \mathcal{S} é definida por:

$$\mathcal{L}(\mathcal{S}) = \{x \in (\Sigma - N)^* \mid \alpha \Rightarrow_{\mathcal{S}}^* x, \alpha \in \mathcal{A}\}$$

Intuitivamente as caixas $\boxed{i_m}$ capturam as palavras z_{i_m} quando a regra é aplicada e reproduzem as palavras capturadas por estas caixas na cadeia do lado direito da regra.

Exemplo 5.3 A regra $aa\boxed{1}b\boxed{2}a \rightarrow a\boxed{2}ba\boxed{1}$ aplicada à palavra $aaabbaa$ coloca ab na caixa numerada com 1 e a na caixa numerada por 2 e gera a palavra $aabaab$. Note que outra solução seria colocar a na caixa 1 e ba na caixa 2 e o resultado neste caso seria $ababaa$.

Exemplo 5.4 Seja S o SPP no qual

$$\begin{aligned}\Sigma &= \{\square, \diamond, \underline{suc}, S, , \} \\ N &= \{S, \underline{suc}, , \} \\ \mathcal{A} &= \{\underline{suc}x | x \in (\Sigma - N)^*\} \\ \mathcal{R} &= \left\{ \begin{array}{l} \underline{suc}\boxed{1} \rightarrow S\boxed{1}, \\ S, \boxed{1} \rightarrow \square\boxed{1} \\ S\boxed{1}\square, \boxed{2} \rightarrow \boxed{1} \diamond \boxed{2} \\ S\boxed{1} \diamond, \boxed{2} \rightarrow S\boxed{1}, \square\boxed{2} \end{array} \right.\end{aligned}$$

A seguir serão exibidas algumas derivações em S :

Axioma	Geração	Axioma	Geração
\underline{suc}	$\underline{suc} \Rightarrow S,$ $S, \Rightarrow \square$	$\underline{suc}\diamond$	$\underline{suc}\diamond \Rightarrow S\diamond,$ $S\diamond, \Rightarrow S, \square$ $S, \square \Rightarrow \square\square$
$\underline{suc}\square$	$\underline{suc}\square \Rightarrow S\square,$ $S\square, \Rightarrow \diamond$	$\underline{suc}\square\square$	$\underline{suc}\square\square \Rightarrow S\square\square,$ $S\square\square, \Rightarrow \square\diamond$

Pode-se interpretar este SPP como gerando a partir de um axioma \underline{suc} , onde n é a representação diádica de um natural no alfabeto $\{\square, \diamond\}$, a representação $n + 1$.

As regras de inferência de um *SPP* podem ser bastante complexas. As operações básicas são de concatenação, casamento de padrões e substituição. Tais operações não fazem parte do sistema formal e são de difícil execução, mesmo quando o agente é um ser humano bem treinado. O leitor familiarizado com os processos de análise sintática para a construção de compiladores pode apreciar tais dificuldades. A complexidade da linguagem \mathcal{L} é um fator preponderante tanto para o uso do casamento de padrões como para a construção das regras de inferência. O leitor interessado pode consultar Book e Otto [81] que trata dos chamados sistemas de reescrita.

5.3 Linguagens Formais e Gramáticas

O aprendizado de uma língua qualquer envolve o estudo da estrutura das sentenças. A grosso modo esta estrutura é chamada de gramática para a língua, e sua apresentação usual é como um conjunto de critérios chamados de regras gramaticais que definem a correteza das sentenças. As regras gramáticas são sistemas formais geradores de linguagens, portanto deve-se esperar que uma gramática construa a língua.

Em geral, uma linguagem natural tende a ser muito extensa e complexa. Duas das atividades dos linguistas são definir com precisão as sentenças válidas de linguagens e encontrar formas estruturadas de representá-las. Entretanto, há uma dificuldade para definir completamente estas linguagens. Considere a seguinte sentença do português: *O menino louco pintava o lindo quadro*.

Pode-se dizer que a sentença é constituída de sujeito “O menino louco- e predicado - “pintava o lindo quadro”. Estes elementos podem ser mais analisados. O predicado é constituído de verbo “pintava” e objeto “lindo quadro”. A análise desta sentença é apresentada na Figura 5.2 e descreve uma representação conhecida como árvore de derivação sintática. Quando uma sentença produz uma árvore de derivação válida, diz-se que esta sentença é válida. Contudo, para uma árvore de derivação ser válida é preciso que ela seja especificada, uma tarefa pouco trivial quando se trata de uma linguagem natural. Uma tentativa incompleta de formalização das regras

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 Discrete 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.