

4.10 pode ser representada pelo conjunto $\{\neg P(x, f(x)) \vee R(x, f(x), g(x)), Q(x, g(x)) \vee R(x, f(x), g(x))\}$. O conjunto S de cláusulas será um ponto chave no procedimento para prova automática de teoremas.

4.6 Teorema de Herbrand

Até o presente momento, foi apresentado um ferramental de suporte para ser empregado na prova de teoremas. Nesta seção serão considerados os procedimentos desta prova. A tarefa de se encontrar um procedimento para averiguar a validade (consistência) de uma fórmula foi uma tarefa estudada no passado. Primeiramente foi tentado por Leibniz, e em seguida por Peano quase duzentos anos depois, próximo a virada para o século XX. Na década de 1920 foram dados os primeiros passos efetivos neste sentido por Herbrand, mas foi só em 1936, através de Church e Turing que isso se mostrou ser uma tarefa impossível. De forma independente, tanto Church como Turing mostraram que não existe um procedimento genérico para checar a validade de fórmulas em lógica de primeira ordem. Entretanto, para o caso específico no qual as fórmulas são de fato válidas, existem procedimentos que podem chegar a esta conclusão. Para o caso em que as fórmulas não são válidas, estes procedimentos nunca se encerram. Sob a ótica de Church e Turing, está é, na melhor das hipóteses, a situação limite que se pode atingir com estes procedimentos de prova.

Uma abordagem importante para a prova automática de teoremas foi dada por Herbrand² em 1931. Por definição, uma fórmula é válida se ela assumir valores 1 para todas as interpretações. Herbrand propôs um algoritmo para encontrar uma interpretação que refuta uma determinada fórmula. Todavia, se uma fórmula realmente é válida, tal interpretação não deverá existir, e seu algoritmo deverá abortar após um número finito de tentativas. Seu método era impraticável de se aplicar até a invenção do computador digital. Só após o artigo de Robinson [64, 65] em 1965, junto com o desenvolvimento do princípio da resolução, foi possível o desenvolvi-

²Jacques Herbrand (1908-1931) foi um matemático francês, cujo estudo principal foi a teoria de demonstração e as funções recursivas gerais intitulado "On the consistency of arithmetic". Em uma escalada dos Alpes franceses com dois amigos, caiu nas montanhas de granito do Massif des Écrins e morreu. "On the consistency of arithmetic" foi publicado postumamente.

mento dos provadores.

Os procedimentos de Hebrand não seguem a tradição de prova direta, mas sim busca a prova por refutação, demonstrando que a negação da fórmula é inconsistente. A idéia intuitiva por de trás deste procedimento é de que se a negação de um teorema for falsa então ele será verdadeiro (princípio do meio excluído).

Por definição, um conjunto S de cláusulas é insatisfatório se e somente se ele é falso (0) sob todas as interpretações sobre todos os domínios. Uma vez que é inconveniente e impossível considerar todas as interpretações sobre todos os domínios, seria interessante se fosse possível fixar um domínio especial H no qual S é insatisfatório se e somente se S é falso sob todas as interpretações sobre este domínio. Felizmente, existe tal conjunto, e ele é chamado de *universo de Herbrand* de S , sendo definido conforme se segue.

Definição 4.13 [Universo de Hebrand] Seja H_0 um conjunto de constantes que ocorrem em S . Se não existe constante em S , então H_0 consiste de uma única constante, $H_0 = \{a\}$. Para $i = 0, 1, 2, \dots$ seja H_{i+1} a união de H_i e o conjunto de todos os termos da forma $f^n(t_1, \dots, t_n)$ para todas as funções f^n que ocorrem em S , onde t_j , $j = 1, \dots, n$ pertencem ao conjunto H_i . Os H_i são chamados de conjuntos nível i de constantes de S , e H_∞ é chamado de *Universo de Herbrand* de S .

Exemplo 4.11 Seja $S = \{P(a), \neg P(x) \vee P(f(x))\}$

$$\begin{aligned} H_0 &= \{a\} \\ H_1 &= \{a, f(a)\} \\ H_2 &= \{a, f(a), f(f(a))\} \\ &\vdots \\ H_\infty &= \{a, f(a), f(f(a)), f(f(f(a))), \dots\} \end{aligned}$$

Exemplo 4.12 Seja $S = \{P(x) \vee Q(x), R(z), T(y) \vee \neg W(y)\}$. Uma vez que não existe constante em S , tem-se que $H_0 = \{a\}$. Além disto, também não existem funções em S , logo $H_0 = H_1 = \dots = H_\infty = \{a\}$.

Exemplo 4.13 Seja $S = \{P(f(x), a, g(x, y), b)\}$

$$\begin{aligned} H_0 &= \{a, b\} \\ H_1 &= \{a, b, f(a), f(b), g(a, a), g(a, b), g(b, a), g(b, b)\} \\ H_2 &= \{a, b, f(a), f(b), g(a, a), g(a, b), g(b, a), g(b, b), f(f(a)), f(f(b)), f(g(a, a)), \\ &\quad f(g(a, b)), f(g(b, a)), f(g(b, b)), \dots\} \\ &\vdots \end{aligned}$$

Definição 4.14 [Base de Herbrand] Seja S um conjunto de cláusulas. O conjunto de todas as fórmulas atômicas de S é chamado de Base de Herbrand, ou conjunto atômico de S .

Exemplo 4.14 Seja $S = \{P(x, a), \neg Q(x) \vee P(f(x), b)\}$ contendo duas constantes e um símbolo funcional. Assim,

$$\begin{aligned} H_0 &= \{a, b\} \\ H_1 &= \{a, b, f(a), f(b)\} \\ H_2 &= \{a, b, f(a), f(b), f(f(a)), f(f(b))\} \\ &\vdots \\ H_\infty &= \{a, b, f(a), f(b), f(f(a)), f(f(b)), \dots\} \end{aligned}$$

Como P e Q são símbolos predicativos, tem-se que a base de Hebrand é dada por $A = \{P(a, b), Q(a), Q(b), P(a, f(a)), P(a, f(b)), Q(f(a)), Q(f(b)), \dots\}$

Chama-se árvore semântica a árvore binária tal que os descendentes de cada nó são respectivamente elementos da base de Herbrand e sua negação (fórmulas atômicas complementares). A árvore semântica T associada ao conjunto de cláusulas do Exemplo 4.14 é apresentada na Figura 4.1. Nesta representação, optou-se por explicitar os valores verdades assumidos pelas fórmulas durante a transição de um nó para outro.

Seja a fórmula $(\forall x)(\forall y)((P(x) \rightarrow Q(x)) \wedge P(f(y)) \wedge \neg Q(f(y)))$. Assim, tem-se que a fórma de skolen é dada por

$$\begin{aligned} G &\equiv (\forall x)(\forall y)((P(x) \rightarrow Q(x)) \wedge P(f(y)) \wedge \neg Q(f(y))) \\ &\equiv (\forall x)(\forall y)((\neg P(x) \vee Q(x)) \wedge P(f(y)) \wedge \neg Q(f(y))) \end{aligned}$$

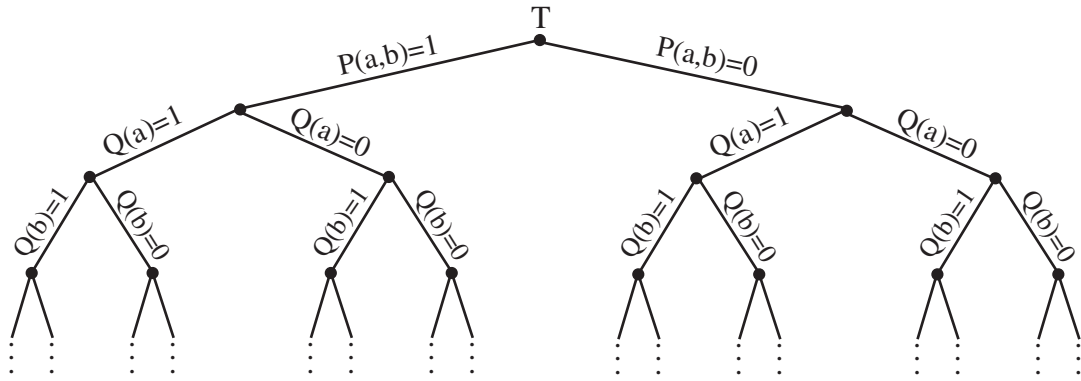


Figura 4.1: Árvore semântica para $S = \{P(x, a), \neg Q(x) \vee P(f(x), b)\}$.

Logo, o conjunto de cláusulas é dado por $S = \{\neg P(x) \vee Q(x), P(f(y)), \neg Q(f(y))\}$.

Deste modo tem-se:

$$\begin{aligned}
 H_0 &= \{a\} \\
 H_1 &= \{a, f(a)\} \\
 H_2 &= \{a, f(a), f(f(a))\} \\
 &\vdots \\
 H_\infty &= \{a, f(a), f(f(a)), \dots\}
 \end{aligned}$$

Como se tem símbolos predicativos P e Q , tem-se que a base de Herbrand é dada por $A = \{P(a), Q(a), P(f(a)), Q(f(a)), \dots\}$. A árvore semântica cheia associada a este conjunto de cláusulas é apresentada na Figura 4.2(a).

Os caminhos existentes em uma árvore semântica oferecem interpretações para o conjunto de cláusulas S que representa, podendo satisfazê-lo ou não. Sob esta ótica, chama-se a atenção para o fato de que é comum que as árvores semânticas cheias possuam uma altura infinita, dado que a própria base de Herbrand pode ser um conjunto infinito. Consequentemente, o conjunto de interpretações também é infinito, o que dificulta de sobremaneira uma solução computacional.

Se o conjunto de cláusula S não for passível de ser satisfeito, então todas as interpretações irão falhar em tentar assumir um valor verdade 1 para S . Observe que quando uma interpretação I , isto é, um caminho na árvore semântica falsifica S , pode-se ignorar as demais ramificações abaixo deste caminho. Desta forma, pode-se efetuar as definições a seguir.

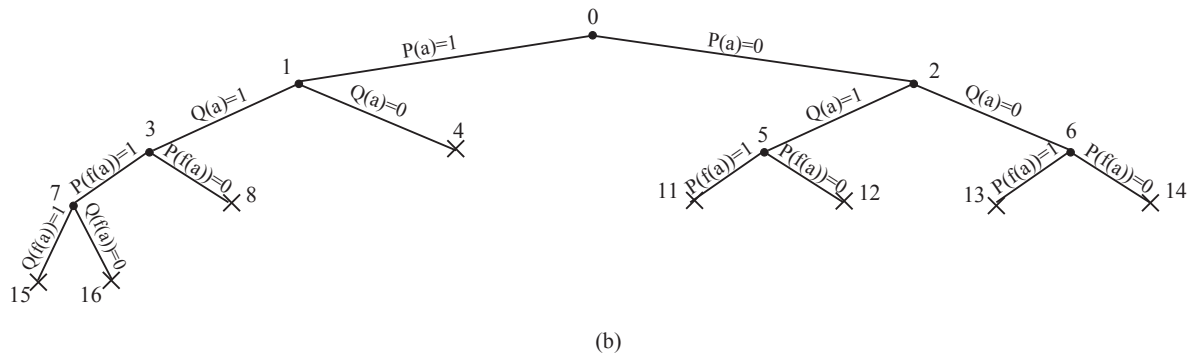
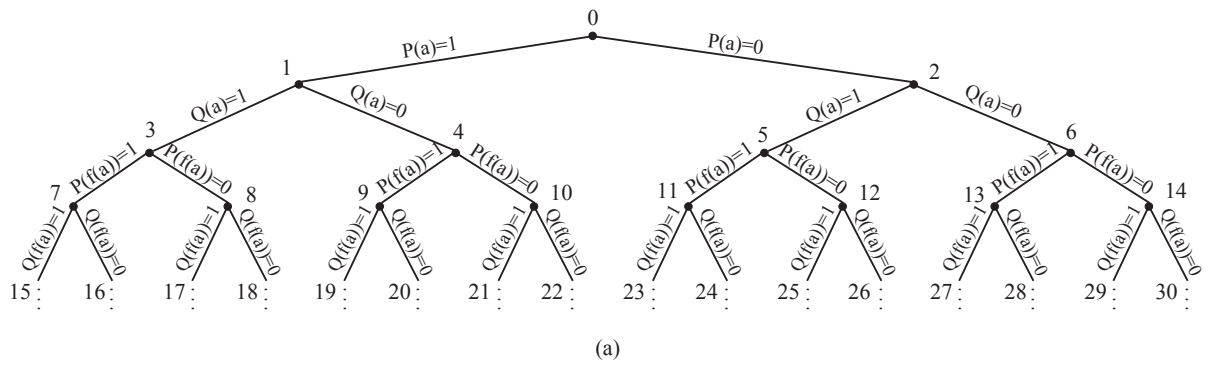


Figura 4.2: Árvore semântica para $S = \{\neg P(x) \vee Q(x), P(f(y)), \neg Q(f(y))\}$: (a) cheia, (b) fechada.

Definição 4.15 [Nó de Falha] Um nó N é um nó de falha se a interpretação $I(N)$ associada ao caminho desde a raiz da árvore semântica até o nó N em questão falsifica ao menos uma das cláusulas de S . Os nós desdobrados após o nó de falha são ignorados, isto é, são podados da árvore diminuindo a altura daquele ramo.

Definição 4.16 [Árvore Semântica Fechada] Uma árvore semântica é dita ser fechada se e somente se todos os ramos da árvore se encerram em nós de falha.

Definição 4.17 [Nó de Inferência] Um nó N de uma árvore semântica fechada é chamado de nó de inferência se todos os nós descendentes imediatos de N são nós de falha.

Neste sentido, retornando ao exemplo da Figura 4.2 cujo conjunto de cláusulas é $S = \{\neg P(x) \vee Q(x), P(f(y)), \neg Q(f(y))\}$ pode-se concluir que a árvore da Figura 4.2(b) corresponde à sua árvore semântica fechada. A descrição do processo de poda, viabilizado pela identificação dos nós de falha é detalhado a seguir.

De modo a facilitar que o leitor acompanhe o processo de poda da árvore, as cláusulas de S serão chamadas de $C1$, $C2$ e $C3$, onde $C1 = \neg P(x) \vee Q(x)$, $C2 = P(f(y))$ e $C3 = \neg Q(f(y))$. Além disto, observe que os nós da árvore semântica (ver Figura 4.2) encontram-se numerados segundo um percurso em nível. As implementações de provadores automáticos de teoremas não utilizam necessariamente este percurso. Nós o fazemos apenas por uma questão didática.

- Caminho 0 – 1: $P(a) = 1$ não torna falsa nenhuma das cláusulas. O valor verdade deste caminho pode vir a tornar a cláusula $C1$ falsa, mas isto dependerá de $Q(x)$. Nada pode ser afirmado sobre as demais cláusulas.
- Caminho 0 – 2: $P(a) = 0$ não torna falsa nenhuma das cláusulas.
- Caminho 0 – 1 – 3: $P(a) = 1$, $Q(a) = 1$ não torna falsa nenhuma das cláusulas.
- Caminho 0 – 1 – 4: $P(a) = 1$, $Q(a) = 0$ falsifica a cláusula $C1$, logo trata-se de um nó de falha. Os demais desdobramentos deste nó são removidos.
- Caminho 0 – 2 – 5: $P(a) = 0$, $Q(a) = 1$ não torna falsa nenhuma das cláusulas.

- Caminho 0–2–6: $P(a) = 0, Q(a) = 0$ não torna falsa nenhuma das cláusulas.
- Caminho 0–1–3–7: $P(a) = 1, Q(a) = 1, P(f(a)) = 1$ não torna falsa nenhuma das cláusulas. Chama-se a atenção para o início da utilização do elemento de H_∞ chamado de $f(a)$. Isto significa que, a partir deste ponto, devem ser consideradas todas as substituições envolvendo a e $f(a)$.
- Caminho 0–1–3–8: $P(a) = 1, Q(a) = 1, P(f(a)) = 0$ falsifica a cláusula $C2$, logo trata-se de um nó de falha.
- Caminho 0–2–5–11: $P(a) = 0, Q(a) = 1, P(f(a)) = 1$ não provocam nenhuma falsidade quando procedemos a substituição de $x = a$ e $y = a$. Entretanto, a substituição de $f(y) = a$ falsifica a cláusula $C2$, de onde se conclui que trata-se de um nó de falha.
- Caminho 0–2–5–12: $P(a) = 0, Q(a) = 1, P(f(a)) = 0$ falsifica a cláusula $C2$, logo trata-se de um nó de falha.
- Caminho 0–2–6–13: $P(a) = 0, Q(a) = 0, P(f(a)) = 1$ não provocam nenhuma falsidade quando procedemos a substituição de $x = a$ e $y = a$. Entretanto, a substituição de $f(y) = a$ falsifica a cláusula $C2$, de onde se conclui que trata-se de um nó de falha.
- Caminho 0–2–6–14: $P(a) = 0, Q(a) = 0, P(f(a)) = 0$ falsifica a cláusula $C2$, logo trata-se de um nó de falha.
- Caminho 0–1–3–7–15: $P(a) = 1, Q(a) = 1, P(f(a)) = 1, Q(f(a)) = 1$ falsifica a cláusula $C3$, logo trata-se de um nó de falha.
- Caminho 0–1–3–7–16: $P(a) = 1, Q(a) = 1, P(f(a)) = 1, Q(f(a)) = 0$ não provocam nenhuma falsidade quando procedemos a substituição de $x = a$ e $y = a$. Entretanto, a substituição de $f(y) = a$ falsifica a cláusula $C3$, de onde se conclui que trata-se de um nó de falha.

Teorema 4.3 [Teorema de Herbrand] Um conjunto de cláusulas S é insatisfatível se e somente se para sua árvore semântica completa existe uma árvore semântica fechada finita.

Deste modo, um conjunto de cláusulas S é insatisfatível se existe um conjunto finito de instâncias de S que é insatisfatível. Sob esta ótica, o conjunto $S = \{\neg P(x) \vee Q(x), P(f(y)), \neg Q(f(y))\}$, cujas árvores semânticas foram ilustradas na Figura 4.2 é insatisfatível. O teorema, do modo como é colocado, sugere a implementação proposta no Algoritmo 4.2.

Data: Conjunto de cláusulas S

Result: S é satisfatível, ou insatisfatível

$B = \emptyset$;

while B é satisfatível **do**

$b = \text{nova} - \text{instância}(S)$;

$B = B \cup \{b\}$;

end

Algoritmo 4.2: Algoritmo imediato para a implementação do Teorema de Herbrand

Observe que este algoritmo não garante sempre obter uma resposta dado que o loop somente se encerra no momento em que B passa a apresentar um conjunto de cláusulas insatisfatíveis. Neste sentido torna-se importante definir uma estratégia de produção de cláusulas capazes de abreviar a execução do programa. Entre estas estratégias pode-se destacar como mais significativas o método de Gilmore [66], o método de Davis-Putman [67] e finalmente o método da resolução de Robinson [64, 65].

4.7 O Princípio da Resolução

O Teorema de Herbrand permite decidir se um conjunto de cláusulas dadas é insatisfatível. Porém, como a árvore semântica cresce exponencialmente, dependendo das cláusulas em questão, a quantidade de elementos envolvidos pode superar a capacidade de armazenamento dos maiores computadores existentes, o que a torna impraticável.

Para evitar esta excessiva geração de elementos, Robinson estabeleceu em 1965 outro algoritmo que pode ser aplicado diretamente sobre qualquer conjunto S de cláusulas para verificar sua insatisfabilidade. Sua idéia essencial é verificar se S

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.