

Capítulo 1

Introdução

A Teoria da Computação tem fundamental importância na formação de Engenheiros de Computação. Ela não só proporciona o embasamento teórico necessário para um correto e amplo entendimento da Computação, como também proporciona o desenvolvimento do raciocínio lógico e formal, do pensamento indutivo e recursivo e da capacidade de abstração. Além disto, introduz os conceitos fundamentais que são desenvolvidos em outras áreas, tais como: linguagens formais, semântica formal, compiladores, orientação a objetos, algoritmos, complexidade computacional entre outros.

Os profissionais de áreas correlatas à Computação se posicionam em áreas de atuação de acordo com suas especialidades, tal como ilustrado na Figura 1.1. O círculo mais interno enfatiza as atividades relacionadas com o hardware e comumente está associado aos Engenheiros Eletrônicos.

No círculo de software básico encontram-se os Engenheiros de Computação. Nesta região agrupam-se as atividades relacionadas com sistemas operacionais tais como o acesso ao hardware e tarefas com alta abstração para o usuário, bem como aquelas relacionadas com tradutores, isto é, programas que aceitam outros programas escritos em uma linguagem e os reproduzem em uma outra linguagem. Os Engenheiros de Computação também atuam no círculo seguinte, o de software de suporte, juntamente com os Engenheiros de Software. Neste círculo predominam os sistemas gerenciadores de bancos de dados, as linguagens de quarta geração, e

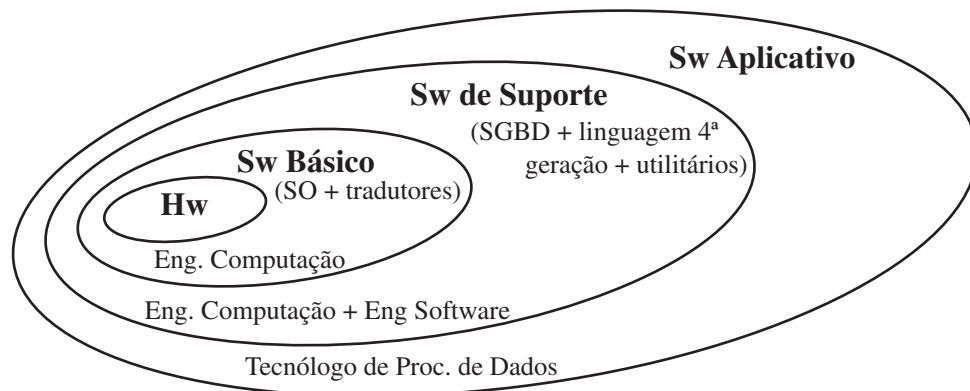


Figura 1.1: Círculos concêntricos da Computação.

os utilitários que implementam tarefas básicas para a utilização do sistema operacional, muitas vezes confundidos com o próprio sistema (geralmente distribuídos juntamente com o sistema operacional). Por fim, o círculo mais externo corresponde ao software aplicativo, o local de origem dos Tecnólogos de Processamento de Dados.

A Figura 1.1 apresenta didaticamente uma estratificação entre as áreas de atuação, entretanto, na prática estas fronteiras não são bem definidas. Dentre estas fronteiras, a única que recebe um nome é aquela entre o hardware e o software básico, chamada firmware. Os profissionais podem transpor livremente estas fronteiras, mas em geral, quanto mais eles se afastam de seu círculo de atuação, maior deverá ser o esforço para se inserir no novo meio.

Independente de suas áreas de atuação, os profissionais de todos os círculos concêntricos da Computação buscam uma solução para um problema utilizando sistemas interpretados ou compilados, que implementam um algoritmo. Apesar de termos como sistemas compilados, interpretados e algoritmo ainda não terem sido oportunamente definidos, tem-se que a busca pela solução para o problema passa, em geral, através da capacidade de representação numérica do ser humano.

O modo através do qual é feita esta representação varia de cultura para cultura, de acordo com as necessidades da sociedade. Culturas indígenas, como por exemplo a Tupi, contam apenas até 10 pois sua sociedade não demanda mais que isso da aritmética. Sob esta ótica, necessidade, cultura e aritmética se combinam

produzindo sistemas de contagem curiosos, tal como o apresentado na Tabela 1.1. Entretanto não são somente as sociedades marginais a nossa era que empregam sistemas de contagem pouco convencionais. Sociedades modernas, tal como a francesa, também podem utilizar sistemas de contagem pouco tradicionais (ver novamente Tabela 1.1).

Tabela 1.1: Sistemas de contagem curiosos.

Tupi			Francês		
Cardinal	Representação	Entendimento	Cardinal	Representação	Entendimento
1	oïepe		10	dix	
2	mokôï		20	vingt	
3	mosapÿr		30	trente	
4	irundyk		40	quarante	
5	xe pó	minha mão	50	cinquante	
6	xe pó oïepe	minha mão e um	60	soixante	
7	xe pó mokôï	minha mão e dois	70	soixante dix	sessenta dez
8	xe pó mosapÿr	minha mão e três	80	quatre vingts	quatro vintes
9	xe pó irundyk	minha mão e quatro	90	quatre vingts dix	quatro vintes dez
10	mokôï pó	duas mãos	100	cent	

A noção de número natural tem uma origem empírica e é uma das mais antigas criações do espírito humano, através da operação de contar. Certas gravuras feitas em cavernas, compostas de traços e pontos, parecem indicar sinais de numeração. Os progressos feitos na arte de contar, ou seja, no que se denomina Aritmética, é uma medida do progresso na vida econômica da humanidade. Segundo Monteiro[2], os indígenas Bakumu (talvez extintos) não sabiam contar além de 30 ou 40, pois seus contratos comerciais ou de matrimônio não ultrapassavam 30 ou 40 unidades. Além disto, certos costumes de registrar o resultado de contagem, que são provavelmente pré-históricos, ainda hoje são utilizados, mesmo por pessoas educadas. Por exemplo é usual se contar pontos em jogos usando-se figuras como a seguir para representar o resultado da contagem 1 a 5.



A evolução na representação dos números naturais desenvolveu, pouco a pouco, certas crenças de fundo místico, como na escola Pitagórica: “o número é a alma das coisas” ou “o número três representa a divindade”, etc. A partir do uso empírico da Aritmética os homens desenvolveram leis de cálculo sobre as representações inventadas. Há pelo menos 5.000 anos a humanidade sabe calcular com os números naturais. No Egito antigo e na Babilônia existiam calculadores profissionais, chamados **escribas** pelos egípcios e **logísticos** pelos gregos. Ifrah [3] apresenta detalhadamente a história dos números.

Depois dos egípcios e babilônios, foram os gregos que mais contribuíram para o desenvolvimento da Aritmética. Euclides iniciou uma ordenação sistemática dos conhecimentos sobre a Aritmética nos Elementos, onde já se pode observar passagens com demonstrações formais de certas regras de cálculo. A preponderância da Geometria, o apelo constante ao uso de figuras para representação geométrica e o desprezo pela “prática” paralizaram o desenvolvimento da Aritmética e geraram uma estagnação no desenvolvimento das técnicas de cálculo. O desenvolvimento de simbolismos é praticamente o único avanço que surge deste período até o Renascimento.

A história das notações aritméticas passou por períodos distintos. O primeiro é conhecido como período retórico em que os problemas da Aritmética eram resolvidos por uma seqüência de raciocínios expressos inteiramente por meio de discurso em linguagem natural (português, francês, etc). Neste sentido, o problema enunciado da seguinte forma “*Eu possuo vinte animais, sendo uma dúzia de ovelhas. Quantos camelos eu possuo?*”, seria resolvido através do discurso “*Supõe-se que ele possui ovelhas e camelos somente. Logo, a quantidade de camelos é resultante da contagem de todos animais, excluindo as ovelhas, isto é, oito camelos*”.

No período sincopado, durante a Idade Média, a Aritmética passou a adotar uma notação em que usavam-se abreviaturas para algumas operações e quantidades, como por exemplo, “*camelos = 20 - 1dz*”. Por fim, nasce a Aritmética simbólica. Pode-se dizer, que com Viète, em seu livro “Logística Speciosa”, propõe o uso de certas letras para os valores desconhecidos e outras para as constantes ou valores conhecidos. Este tipo de notação é empregada até os dias de hoje (“ $x = 20 - 12$ ”).

A criação de um sistema de notações adequadas ao cálculo, nasceu da necessidade de abreviar e simplificar a resolução de diversos problemas que surgiam na vida humana, e determinou um grande desenvolvimento da Aritmética. Apareceram então as regras fixas que permitem calcular com rapidez e segurança, poupando o espírito e a imaginação (Leibniz), e um dos resultados é a Mecanização do Cálculo.

1.1 Algoritmos

Descartes acreditava no emprego sistemático do cálculo algébrico como um método poderoso e universal para resolver *todos os problemas*. Esta crença, junta-se a de outros e surgem as primeiras idéias sobre máquinas universais, capazes de resolver todos os problemas. Esta era uma crença de mentes poderosas que deixaram obras respeitáveis em Matemática e ciências em geral.

A noção de computabilidade foi pela primeira vez formalizada por Gödel em 1931, através do desenvolvimento da teoria das funções recursivas empregando funções primitivas recursivas, definidas anteriormente por Dedekind em 1888 [4]. A recursividade, segundo Gödel [5] pode ser assim definida: as funções recursivas primitivas são precisamente aquelas funções aritméticas que podem ser derivadas do núcleo da recursividade por meio de um número finito de operações mecânicas específicas [6]. Esta forma de observar a questão tornou possível referenciar objetos infinitos a partir de regras finitas de construção.

Alonzo Church, em 1936, utilizou as funções recursivas que Gödel introduziu nos seus estudos sobre os teoremas da incompletude de modo a criar o conceito de algoritmo. O *lambda-calculus* foi materializado através de diversas linguagens funcionais de computador, considerando que seu reticulado de iterações não era mais difícil para um computador do que qualquer outra operação mecânica [7].

A preocupação constante em minimizar o esforço humano gerou o desenvolvimento de máquinas que passaram a substituir o homem na realização de tarefas, que são consideradas como atividades *inteligentes*, por exemplo: jogar xadrez, demonstrar teoremas, etc. Hoje poderíamos reservar o qualificativo *inteligente* apenas para atividades que parecem depender fortemente de fatores perceptivos. As tentativas

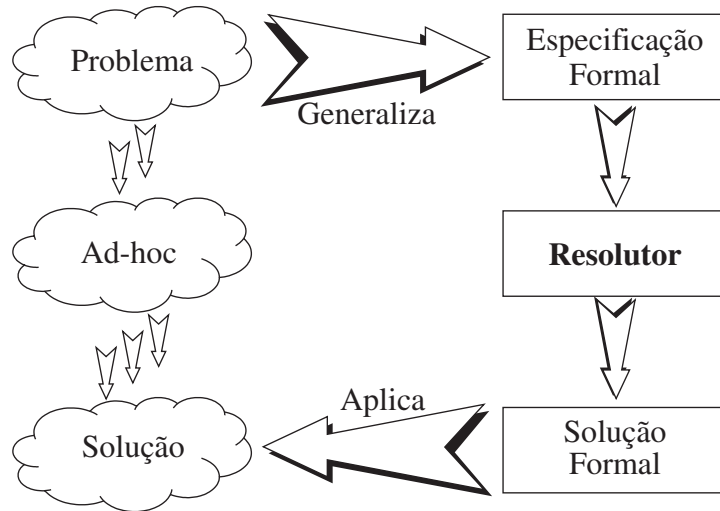


Figura 1.2: Métodos efetivos para resolução de problemas.

de mecanização de atividades desta natureza, como por exemplo a formação, identificação e manipulação de conceitos abstratos, tem apresentado resultados pouco satisfatórios. Na verdade não existem argumentos definitivos em tais questões. O que existe é uma atitude generalizada em não aceitar como *inteligente* nenhuma atividade passível de tratamento *sistemático*, ou executada pelo uso de *procedimentos efetivos*. A questão é, portanto, saber que atividades gozam desta prerrogativa, ou seja, que atividades são *computáveis*.

A teoria da computação é responsável pela formalização e explicação dos conceitos mencionados. Ela precedeu o surgimento dos computadores e seu desenvolvimento é absolutamente independente do estado corrente da tecnologia. Esta teoria baseia-se na definição e construção de máquinas abstratas e no estudo do poder destas máquinas na solução de problemas.

O homem tem, através dos tempos, tentado construir métodos efetivos para resolução de problemas. Os problemas podem ser resolvidos um a um, ou agrupados em classes de forma que, dada a solução de um dos elementos, os demais estarão solucionados. A segunda hipótese requer a concepção de métodos gerais a que chamamos *resolutores*.

A Figura 1.2 ilustra as abordagens direta, pelo uso de *métodos ad hoc*, ou seja

pela ausência de métodos e a indireta obtida pela utilização de *mecanismos* desenvolvidos com o objetivo de resolver classes cada vez mais abrangentes de problemas, aos quais damos o nome de *resolutores*.

Alguns destes resolutores foram delineados na antiguidade histórica, sob a forma de *algoritmos* e seu uso faz parte dos currículos escolares em todo o mundo. Aprendemos a resolver problemas aritméticos como adição, subtração, multiplicação, divisão e até mesmo a extração de raízes, sem nunca nos perguntarmos o *porquê* de tais *métodos*, com a confiança que somente as crianças possuem. Nos tornamos adultos e aprendemos outros *métodos* para problemas mais complicados e na maioria dos casos com a mesma crença no acerto dos mesmos. Muitas vezes os resolutores são aparelhos mecânicos como os ábacos, régua de cálculo ou máquinas de calcular ou aparelhos eletrônicos como calculadoras eletrônicas. O uso de tais aparelhos exige um aprendizado de uma linguagem ou pelo menos a identificação de botões ou manivelas e uma seqüência de operação, ou seja, de um algoritmo de uso. De um modo informal, podemos dizer que, o que chamamos de algoritmo, é nada mais que uma seqüência finita de instruções escritas ou faladas de alguma linguagem, normalmente uma língua natural, por exemplo o português.

Nos anos 20 e 30 houve um grande interesse na noção de algoritmo, no sentido de tornar esta noção intuitiva em um conceito preciso, para propiciar o estudo de suas propriedades de maneira rigorosa. Vários matemáticos - na Inglaterra (Alan Turing), nos Estados Unidos (Alonzo Church, Emile Post), na Áustria (Kurt Gödel) na União Soviética (Andrey Markov) - desenvolveram definições da noção de algoritmo através o desenvolvimento de certas máquinas abstratas e de linguagens artificiais. Todas as propostas feitas para definir *algoritmo*, até os dias atuais, foram mostradas equivalentes. O autor de uma dessas linguagens, o americano Alonzo Church evidenciou este fato, e isto o levou a formular o que passou a ser conhecida como a **Tese de Church**: "tais formalismos são caracterizações tão gerais da noção do efetivamente computável quanto consistentes com o entendimento intuitivo usual".

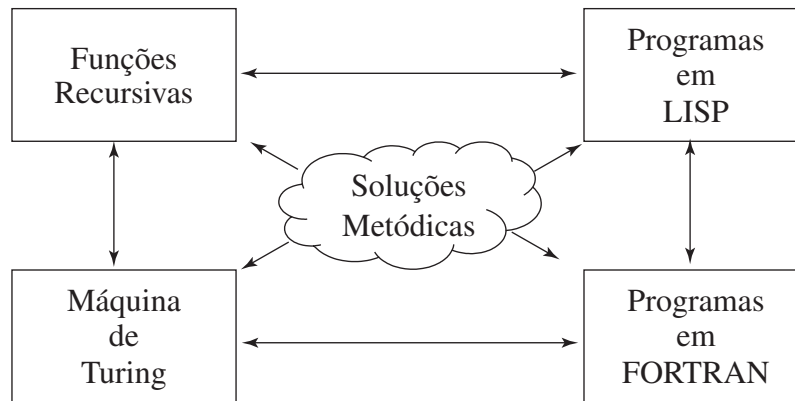


Figura 1.3: Esquema ilustrativo da Tese de Church-Turing.

1.2 Tese de Church-Turing

A Tese de Church, ou hipótese de Church-Turing, ilustrada na Figura 1.3, não é um teorema mas sim um resultado epistêmico cuja aceitação é quase universal. O conceito intuitivo de computável é identificado com uma certa classe de funções aritméticas chamadas funções recursivas, cuja caracterização é um dos objetivos deste livro.

O leitor pode pensar que se a hipótese não pode ter sua veracidade comprovada de forma direta, então talvez ela possa ser refutada. Assim, para negar a tese basta encontrar um procedimento que não pudesse demonstradamente ser computado por uma Máquina de Turing. Até o presente momento isto não ocorreu, e ainda, em virtude do grande número de dados experimentais favoráveis, esta tese tem sido aceita pelos estudiosos do assunto. Além disto, as diversas tentativas independentes de formalizar o conceito de algoritmo resultaram em formalismos que podem ser demonstrados como equivalentes ao de Turing.

Neste sentido, a Tese de Church é uma hipótese sobre a natureza mecânica do ato de calcular, relacionando-se diretamente com o computador, e com os tipos de algoritmos eles podem executar. Assim, toda função considerada sistematizável pode ser computada por uma Máquina de Turing. Programas podem ser traduzidos em uma Máquina de Turing, bem como, qualquer Máquina de Turing pode ser traduzida para uma linguagem de programação. Conseqüentemente, qualquer linguagem de

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*, Coleção Schaum. São 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.