

Figura 6.3: Autômato finito determinístico reconhecedor de strings que não contenham 3 b 's consecutivos.

fácil afirmar que a máquina pára no estado final q_0 quando houver um número par de b 's.

Exemplo 6.2 Projetar um autômato finito determinístico M que reconheça a linguagem $L(M) = \{w \in \{a, b\}^* \mid w \text{ não contém 3 } b\text{'s consecutivos}\}$

$$k = \{q_0, q_1, q_2, q_3\}$$

$$\Sigma = \{a, b\}$$

$$s = q_0$$

$$F = \{q_0, q_1, q_2\}$$

A Figura 6.3 apresenta o autômato correspondente.

Definição 6.3 Uma linguagem aceita por um DFA é uma linguagem regular.

6.2 Autômato Finito Não Determinístico - NFA

Na seção anterior foi observado que os DFA's apresenta como característica mais marcante a transição inequívoca de um estado para outro, isto é, $\delta(a, q) = q'$. Os autômatos finitos não determinísticos (NFA), abordados nesta seção, apresentam uma transição que não pode ser definida de modo assertivo. De fato, a função de transição desta espécie de autômato pode apresentar mais de uma opção de estado de destino, ou seja, $\delta(a, q) = q'$ ou q'' ou $q''' \dots$.

Sob esta ótica, pode-se afirmar que o conjunto de todos os autômatos finitos não determinísticos é um super conjunto dos autômatos finitos determinísticos.

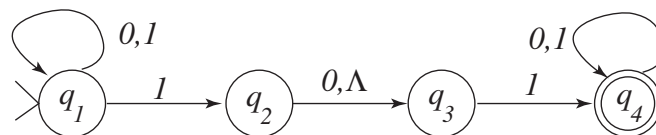


Figura 6.4: Exemplo de autômato finito não determinístico.

Além disto, também é possível afirmar que para todo NFA, existe um DFA semelhante.

O exemplo da Figura 6.4 apresenta a representação gráfica de um NFA. Este autômato guarda muita semelhança com os DFA's já estudados, porém ele apresenta um símbolo novo, Λ , que representa um *símbolo vazio*. Isto quer dizer que a transição do estado q_2 para o estado q_3 pode acontecer por dois motivos: (1) uma transição tradicional que é realizada sempre que for lido um símbolo 0; (2) uma transição imediata que pode ser feita a qualquer momento, bastando que a máquina esteja no estado q_2 , independente se leu ou não um símbolo. Observe que o não determinismo do autômato reside justamente nesta situação. Quando o autômato está no estado q_2 e surge o símbolo 0, ele é obrigado a ir para o estado q_3 . Entretanto, se ele está neste mesmo estado q_2 e surge qualquer outro símbolo diferente de 0, então ele pode, ou não, ir para o estado q_3 .

Os NFA's não foram concebidos para serem utilizados por modelos realistas computacionais, eles são generalizações notacionais dos autômatos finitos. Os NFA's são, em geral, muito menores que os DFA's, além de simplificarem significativamente a descrição dos autômatos. Em um NFA, uma cadeia é aceita se existir alguma maneira de levar o dispositivo de um estado inicial para um estado final, percorrendo as setas rotuladas. Utilizando o exemplo da Figura 6.4, observe o esquema da Figura 6.5 contendo as possíveis configurações que o autômato pode assumir, dado uma string de entrada 010110 (sugere-se que sejam testadas também as strings 010, 101, 0110). Uma análise deste autômato permite afirmar que a linguagem aceita por esta máquina é dada pelo conjunto de strings que contenham as substrings 101 ou 11.

Observe ainda que existem dois percursos que permitem atingir o único estado final desta máquina, no caso o q_4 . Isto significa que existem dois comportamentos

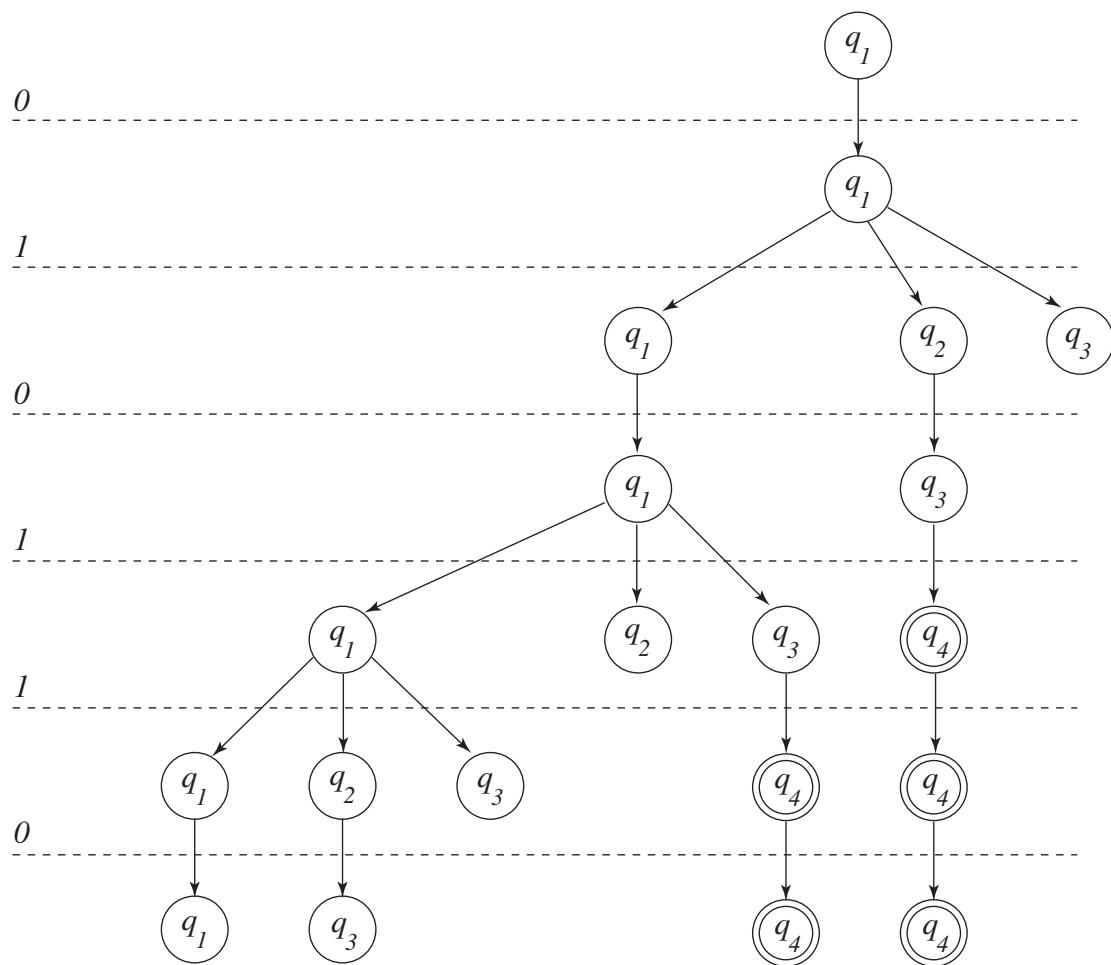


Figura 6.5: Possíveis configurações para o autômato da Figura 6.4.

distintos para a máquina que permitem a aceitação da string 010110. Além disto, também se pode constatar que uma vez o autômato configurado no estado q_1 e lendo um símbolo 1, a máquina pode: (a) ficar em q_1 , devido ao laço existente no próprio estado; (b) seguir para q_2 através da transição existente entre q_1 e q_2 ; (c) ir para q_3 , pois uma vez em q_2 , o autômato pode avançar para q_3 sem consumir uma entrada devido a existência de uma transição para o símbolo Λ .

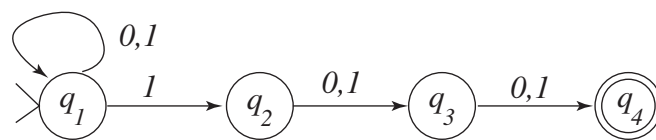
Definição 6.4 Um autômato finito não determinístico deve possuir ao menos uma das seguintes características:

- i) Mais de uma seta rotulada com um mesmo símbolo σ saindo de q ;
- ii) Para um estado e uma determinada entrada pode não haver movimento possível, isto é, $\exists \delta(q, \sigma)$ mas $\nexists \delta(q, \sigma')$;
- iii) Podem existir setas rotuladas Λ indicando que a máquina pode ir para outro estado sem precisar consumir uma entrada.

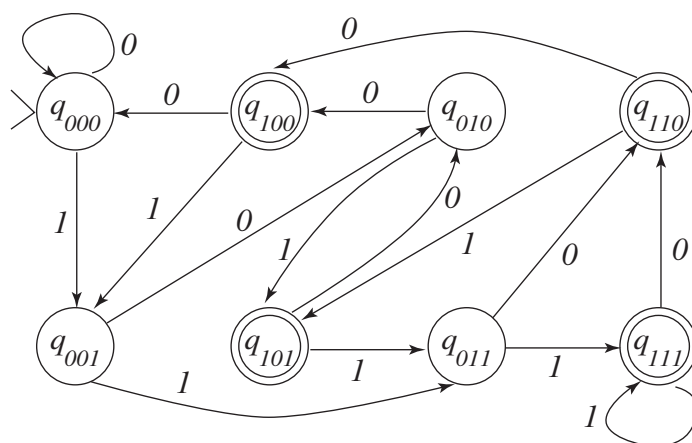
Considere a linguagem formada por strings que contenham o símbolo 1 na terceira posição de trás para frente da string. Existem dois possíveis autômatos para a representação desta linguagem. O autômato não determinístico é bastante simples, sendo ilustrado na Figura 6.6a. A idéia central deste autômato é que ele permanece em q_1 até que adivinhe que faltam apenas três símbolos para o término da string.

O leitor deve se sentir desconfortável com o uso desta adivinhação, haja vista que este procedimento não é passível de ser computável. Aqui, fica evidente a noção de que um NFA é uma generalização notacional, e não um modelo realista computacional. Entretanto, este autômato é mais agradável para ser compreendido do que seu dual determinístico, apresentado na Figura 6.6b. Para eliminar este comportamento não computável, o DFA está a todo momento se preparando para caracterizar um símbolo 1 na anti-penúltima posição, sem necessariamente saber se ainda existem símbolos na fita, ou não.

Sob esta ótica, uma tarefa que se torna interessante, e também desejável, é a transformação de algo cuja notação seja mais simples, para algo que possa ser



(a)



(b)

Figura 6.6: Autômato reconhecedor de strings com o símbolo 1 na 3ª posição de trás para frente da string: (a) NFA; (b) DFA.

exeqüível em sistemas computadorizados. A seção a seguir trata justamente desta conversão.

6.3 Conversão de NFA para DFA

Um NFA, ao adivinhar o caminho a ser seguido durante uma execução, exige de um hardware qualquer desempenhar atividades de difícil sistematização. Apesar de serem de mais fácil compreensão e com notação menos densa, os NFAs não são apropriados para serem transformados em programas de computadores. Por este motivo, existe uma necessidade de converter um autômato não determinístico em um determinístico.

O segredo desta conversão é que cada estado de um NFA corresponde a um conjunto de estados do DFA correspondente. Assim, existem cinco propriedades que devem ser seguidas:

1. Se o NFA possui os estados $k = \{q_0, q_1, q_2, \dots\}$ então o conjunto de estados do DFA é dado pelo conjunto potência de k , isto é:
 $k' = \{\emptyset, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}, \dots\}$. Desta forma, a complexidade de pior caso para a construção destes estados é dada por $O(2^n)$, onde n é a quantidade de estados do NFA.
2. O estado inicial $s = q_0$ do NFA corresponde a um estado inicial $s' = \{q_0\}$ no DFA.
3. Se q_j é um estado final do NFA então todos os conjuntos de estados pertencentes à k' que contém q_j serão estados finais no DFA.
4. A função de transição passa a ser uma função de transição estendida δ^* onde $\delta^*(q, \sigma) = \left\{ \text{todos os estados atingíveis partindo de } q \text{ e lendo } \sigma \right\}$
5. Estados não atingíveis do DFA podem ser ignorados.

Considere o exemplo da Figura 6.7. Neste exemplo sugere-se um autômato não determinístico que é capaz de aceitar strings que contenham em seu término a

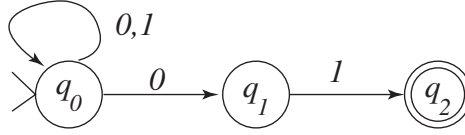


Figura 6.7: Autômato não determinístico reconhecedor de strings contendo 01 no final.

seqüência 01. Uma vez que o NFA possui apenas 3 estados (q_0, q_1, q_2) , temos que o DFA irá possuir $2^3 = 8$ estados, conforme apresentado na representação da função de transição estendida a seguir.

q	0	1
\emptyset	\emptyset	\emptyset
q_0	q_0, q_1	q_0
q_1	\emptyset	q_2
q_2	\emptyset	\emptyset
q_0, q_1	q_0, q_1	q_0, q_2
q_0, q_2	q_0, q_1	q_0
q_1, q_2	\emptyset	q_2
q_0, q_1, q_2	q_0, q_1	q_0, q_2

Segundo que foi apresentado na quarta propriedade da conversão de NFA para DFA, a função de transição estendida δ^* , onde $\delta^*(q, \sigma)$ é dada por todos estados que são factíveis de serem atingidos no NFA, estando no estado q e lendo um símbolo σ , que neste caso pode ser 0 ou 1. Consideremos o caso trivial, o estado \emptyset . Não importa se o autômato da Figura 6.7 lê 0 ou 1, este estado não é previsto, logo concluímos que o melhor a ser feito é manter a máquina configurada no próprio estado \emptyset . Em seguida, considere o caso seguinte, onde a máquina está no estado q_0 . Nesta situação, ao ler um símbolo 0 a máquina tem duas opções: continuar em q_0 ou ir para q_1 . Desta forma dizemos a máquina irá para a junção destes dois estados, isto é, $\{q_0, q_1\}$. Por outro lado, ainda considerando que a máquina esteja em q_0 , e que ela leia o símbolo 1, então observamos que a única opção é se manter em $\{q_0\}$.

Um caso mais interessante é quando o autômato encontra-se no estado $\{q_0, q_1\}$. A montagem da função de transição estendida se dá considerando os estados q_0

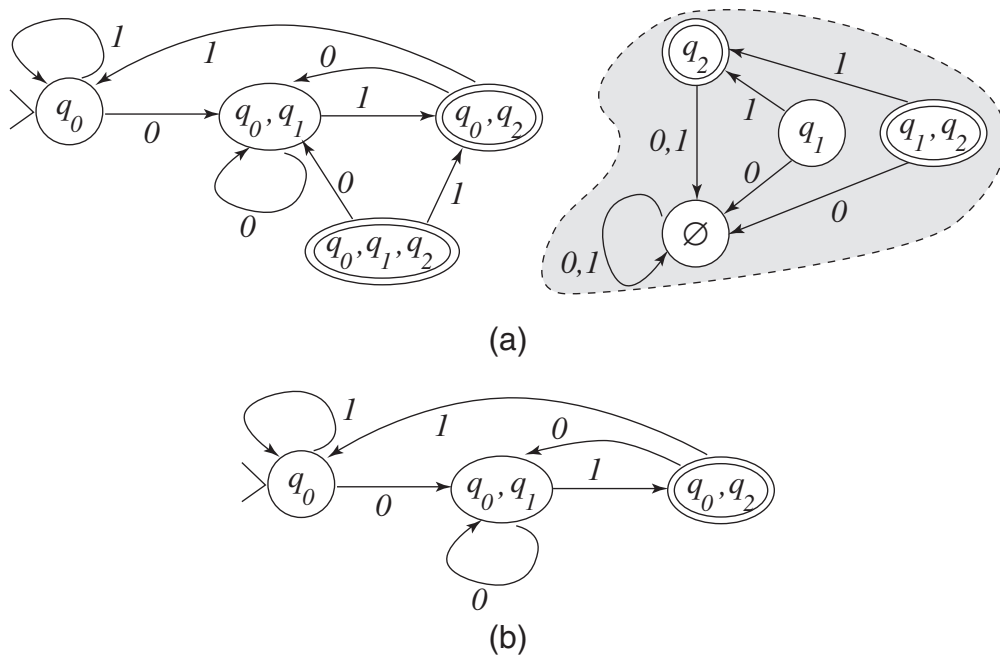


Figura 6.8: Autômato determinístico reconhecedor de strings contendo 01 no final: (a) representação com estados inatingíveis; (b) representação final.

e q_1 separadamente. De acordo com o que já foi constatado para q_0 , temos que $\delta^*(q_0, 0) = \{q_0, q_1\}$. Analogamente podemos dizer que estando em q_1 e lendo o símbolo 0 não há nada o que ser feito a não ser ficar em q_1 . Assim, a união de todos estes possíveis estados atingíveis é dada por $\{q_0, q_1\} \cup \{q_1\} = \{q_0, q_1\}$.

A Figura 6.8(a) apresenta a função de transição estendida representada de modo gráfico, que já representa um esboço do DFA desejado. Tal como descrito nas propriedades da conversão de um NFA em DFA, o estado inicial do autômato continua sendo q_0 , enquanto que os estados finais são todos aqueles que contêm q_2 , isto é, $\{q_2\}$, $\{q_0, q_2\}$, $\{q_1, q_2\}$ e $\{q_0, q_1, q_2\}$.

Nesta forma de visualização também se pode executar a tarefa de ignorar os estados não atingíveis do DFA. Observe que se o autômato se inicia no estado q_0 , não há como alcançar o trecho em evidência dado pelos estados $\{\emptyset\}$, $\{q_1\}$, $\{q_2\}$ e $\{q_1, q_2\}$. Sob esta ótica, o trecho em questão será eliminado. Além disto, o estado $\{q_0, q_1, q_2\}$ também é um estado que nunca pode ser alcançado, dado que só existem setas de transição saindo dele. Assim, este estado também será eliminado. A Figura

6.8(b) apresenta o autômato finito determinístico em sua versão final.

6.4 Minimização de Autômatos

Neste momento, parece sugestível que não há um algoritmo que permita a construção de um autômato qualquer que implemente a linguagem L a ser representada. A construção de autômatos envolve um processo eminentemente criativo, que tende a ser facilitado com a prática e o estudo. Durante o processo de criação, o projetista do autômato, cria um arranjo de estados e transições, sem que necessariamente esteja preocupado com a quantidade dos mesmos. Em geral, problemas desta natureza, objetivando uma otimização, são deixados para uma etapa posterior.

Na seção 6.3 foi apresentado o processo de conversão de um NFA em um DFA. Neste processo, observa-se que também não há uma preocupação formal com a redução da quantidade de estados. Existe apenas um ensaio de otimização quando se efetua a verificação de estados inatingíveis e posteriormente realiza-se a eliminação.

Poder calcular o autômato mínimo é uma questão importante na construção de ferramentas baseadas em autômatos finitos tais como: protocolos de comunicação, processamento de texto, análise de imagens, lingüística computacional, entre outras. Isto reduz a complexidade da engenharia do software, e em última medida, torna o processamento da aplicação mais leve, e seu custo financeiro, menor.

Um autômato com m estados e que aceite uma linguagem L é dito mínimo quando, para todo DFA de n estados que aceite a mesma linguagem L , se a relação entre estas quantidades de estados é dada por $m < n$. Dois estados q e q' são ditos equivalentes (\equiv) se $\forall \sigma, \delta(q, \sigma) \in F \Leftrightarrow \delta(q', \sigma) \in F$. Alternativamente, diz-se que dois estados são distintos ($\not\equiv$) se $\exists \sigma, \delta(q, \sigma) \in F \wedge \delta(q', \sigma) \notin F$, ou vice-versa.

Observe o autômato da Figura 6.9(a), cuja linguagem aceita é dada por $(a, b)(a, b)^*$. Analisando este autômato, percebe-se que a transição de q_0 para q_2 aponta para a construção de strings do tipo $a(a, b)^*$. Ainda nesta mesma figura, as transições que passam pelo estado q_1 permitem a construção de strings do tipo $b(a, b)^*$ de um modo pouco convencional. Isto ocorre porque a passagem por q_1

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.