Uma Proposta para Disciplina de Teoria da Computação

Claudio Cesar de Sá[1] **, Guilherme Bittencourt**[2]

[1]Departamento de Ciência da Computação (DCC)
Universidade do Estado de Santa Catarina (UDESC)
Campus Universitário Prof. Avelino Marcante, S/N
Bloco F - 3o. Piso - Sala F. Gauss
89223-100 - Joinville - SC

[2]Departamento de Automação e Sistemas (DAS)
Universidade Federal de Santa Catarina (UFSC)
88040-900 - Florianópolis - SC

claudio@joinville.udesc.br, gb@das.ufsc.br

Abstract. *This paper presents a curriculum proposal for the course of Theory of Computation, to be taught mainly in Computer Science graduate courses. The point of this proposal is to try to present the classical computation models in an unified way, based on the evaluation, structuration and solution of problems. Therefore, we proposed to introduce the basic concepts of the Theory of Computation using the presentation and discussion of the different classes of problems as a background.*

Resumo. *Neste artigo é apresentada uma proposta curricular para a disciplina de Teoria da Computação, a ser ministrada, principalmente, nos cursos de graduação de Bachalerado em Ciência da Computação. O ponto central desta proposta é uma tentativa de apresentar os principais modelos computacionais clássicos sob um ponto de vista unificado, baseado na avaliação, estruturação e solução de problemas. Assim, a partir da apresentação e discussão das diferentes classes de problemas, pretende-se introduzir os conceitos básicos da área da Teoria da Computação.*

1. Introdução

Atualmente não há um consenso a respeito de uma grade curricular adequada para o curso de Bachalerado em Ciência da Computação (BCC). Embora diversas propostas tenham sido feitas (ver, por exemplo, os resultados do grupo de trabalho da Sociedade Brasileira de Computação `https://listas.inf.ufrgs.br/mailman/listinfo/degt1-1`), o problema parece distante de ser resolvido, restando mesmo dúvidas se uma solução é possível ou mesmo desejável.

Dada a hetereogenidade das Instituições de Ensino Superior (IES), uma grade curricular única não garante um ensino equivalente, pois esta pode ser instanciada diferentemente em cada IES. Assim, um processo de *padronização* da grade curricular estaria

fadado a se tornar algo quase sempre incompleto e/ou inconsistente, restando a solução de estabelecer apenas uma diretriz curricular, permanecendo o problema em aberto até que num momento mais propício uma solução definitiva seja talvez apresentada.

Contudo, alguns resultados positivos já foram alcançados. É consenso, por exemplo, que a grade curricular de BCC apresente disciplinas como Lógica, Matemática Discreta, Cálculo Integral e Diferencial, Álgebra Linear (Álgebra Vetorial e Geometria Analítica), Sistemas Operacionais e Compiladores. Este artigo defende a inclusão da disciplina de *Teoria da Computação (TEC)* nesta lista de disciplinas básicas, seja sob forma de disciplina única ou sob forma diluída, ao longo de várias disciplinas do curso de BCC, e apresenta uma proposta curricular para tal disciplina. Esta proposta visa colaborar para o aperfeiçoamento das grades curriculares para o curso de BCC e eventualmente de outros cursos semelhantes, como o curso de Engenharia da Computação (EC).

2. A Disciplina de Teoria da Computação

Essencialmente, a disciplina de TEC em uma grade curricular visa mostrar a existência de modelos abstratos para o conceito de computação e sua relação com as máquinas reais. Assim, devem ser apresentadas na disciplina definições formais para conceitos como: computação, procedimento, algoritmo, computabilidade, etc.

Estas definições devem ser apresentadas de modo a tornar o mais clara possível a relação entre os objetos reais, o *computador* e seus *programas*, e os modelos abstratos, com seus formalismos matemáticos, com os quais muitas vezes os alunos de graduação não estão familiarizados.

Alguns pontos a serem destacados durante a disciplina:

1. Apresentar os limites fundamentais de tempo e espaço/memória aos quais os problemas reais estão restritos nas máquinas reais e sua relação com os modelos computacionais abstratos, tais como as Máquinas de Turing (MT) e outros, que geralmente contam com recursos ilimitados;
2. Enfatizar as questões decidíveis e indecidíveis da área de linguagens formais e discutir a indecidibilidade do Problema da Parada;
3. Apresentar a Hipótese de Church-Turing;
4. Aplicar o conceito de redutibilidade a outros problemas conhecidos, partir dos limites dos problemas indecidíveis;
5. Formalizar o conceito de complexidade Temporal e Espacial utilizando o modelo de MT.
6. Introduzir as classes de problemas de acordo com a Teoria da Complexidade.

Adicionalmente a este conteúdo, a disciplina de Teoria da Computação deve fornecer ao aluno subsídios para reflexão a respeito de questões como:

1. o que significa a palavra *computação* e quais as implicações deste significado;
2. o que são *algoritmos*, levando em conta os modelos computacionais abstratos estudados;
3. o pode e o que não pode ser efetivamente computado [Furtado, 2003];
4. por que, e de que forma, os problemas se distinguem entre fáceis e difíceis em relação aos critérios de complexidade [Sipser, 1996].

Muitas destes conceitos e definições da área de computação já são conhecidos pelo aluno de graduação, mas normalmente sem o aprofundamento necessário. Termos como: procedimento, algoritmo, computação, etc, são termos comuns e normalmente são apresentados ao aluno sem comprometimento quanto ao rigor matemático. Baseado neste fato sugerem-se os seguintes requisitos para a disciplina de TEC:

1. Esta deve ser apresentada a partir do $4^{o.}$ semestre ($4^{a.}$ fase) na grade curricular dos cursos de BCC e EC. Somente a partir desta fase o aluno apresenta a autonomia de estudo e a maturidade necessárias para relacionar o conteúdo teórico e prático;
2. Para um curso de duração prevista para 4 anos (8 semestres), em regime integral, a disciplina deve ser apresentada antes da $6^{a.}$ fase, pois se antes da $4^{a.}$ fase existe o problema da imaturidade, após a $6^{a.}$ fase o aluno já tem outros interesses que lhe consomem grande parte da energia, como Trabalho de Conclusão de Curso, Estágio, etc;
3. Para isto, esta tem como pré-requisitos as seguintes disciplinas: Álgebra e Cálculo (duas disciplinas distintas), Matemática Discreta, Lógica, Linguagens Formais e Máquinas, e conhecimentos sobre Paradigmas de Linguagens (funcional, lógico e imperativo);

Neste sentido, a disciplina de TEC prepara o aluno para cursos mais avançados como Métodos Formais, Inteligência Artificial, etc. A disciplina de Compiladores pode se apresentar também na $4^{a.}$ ou $5^{a.}$ fase da grade curricular.

2.1. Os Problemas

Este artigo tenta apresentar um programa unificado para a disciplina de Teoria da Computação (TEC), apesar da diversidade dos enfoques que os livros da área apresentam. Há um número considerável de livros publicados nestas últimas décadas, mas com visões diversas da área de TEC. Uma lista preliminar desse conteúdo variado pode ser visto em: [Arbib, 1969, Moret, 1998, Brookshear, 1989, Divério, 1999, Hein, 1999, McCarthy, 1963, Papadimitriou, 1998, Sipser, 1996, Engeler, 1973, Hopcroft and J.D., 1979, Davis and Weyuker, 1983]. Esta diversidade de livros se caracteriza por dois pontos:

- Os livros não apresentam uma sequência *padrão* de conteúdos. Cada autor tem sua própria visão e uma abordagem particular ao assunto;
- Uma variedade de notações que confundem um leitor sem muita experiência no assunto. Este fato é fortemente salientado pelos alunos.

Além disto, há diversos livros complementares que visam fazer uma conexão mais ampla entre a computação e as demais áreas da ciência, por exemplo: [Hoftstadter, 1999, Berlinski, 2002, Aczel, 2003, Boyer, 1991, Gersting, 2001, Penrose, 1991]. O professor deve estar preparado para o fato da disciplina apresentar um enfoque altamente multidisciplinar. Fato este que não é trivial para um iniciante na área.

No Brasil, nas principais universidades com tradição em informática e computação, a disciplina está bem estabilizada quanto ao seu conteúdo programático. Um histórico sobre o seu ensino já está bem delineado e pode ser visto através das páginas dedicadas à disciplina nas diversas universidades:

- UFRGS: http://www.inf.ufrgs.br/~hgmc/teorica/paginas/tecomp.html

- UFMG: http://www.dcc.ufmg.br/~camarao/ftc.html e http://www.dcc.ufmg.br/~nvieira/
- UFCG: http://www.dsc.ufpb.br/~lula/cursos.2003.1/tc/

Em geral, os livros textos utilizados são os mesmos, embora as abordagens sejam ligeiramente diferentes.

3. A Proposta

A proposta de um programa unificado é estruturada a partir dos *problemas* e da dificuldade inerentes em resolvê-los. Deste modo, ao final do curso o aluno deve ter uma idéia clara da área de Teoria da Computação e sua relação com o estudo de problemas computacionais, suas soluções, limites, recursos necessários, etc. Nesta proposta, a classificação dos problemas computacionais apresentada ao aluno é a mesma descrita por O. J. V.Furtado [Furtado, 2003]. Esta classificação tem por critério a *solubilidade* dos problemas e os *recursos* necessários para sua solução, e é descrita como se segue em [Furtado, 2003]:

Problemas Indecidíveis : são problemas cuja solução apresenta uma limitação teórica, o qual os torna impossíveis de serem solucionados. Isto é, para estes problemas foi provada, segundo uma determinada fundamentação matemática, a impossilidade de obter-se uma solução. O clássico *Problema da Parada* é um bom exemplo, embora em alguns sistema operacionais práticos existam programas para a detecção de *loops*, o qual num caso geral a solução é impossível;

Problemas Intratáveis : são problemas solúveis mas com com recursos ilimitados. Ou ainda, solúveis para uma quantidade limitada e finita de instâncias. O que leva à impossibilidade de solucionar certas instâncias com recursos limitados;

Problemas Tratáveis : são problemas que podem ser solucionados com recursos limitados. A maioria dos problemas que o aluno conhece são deste tipo.

Em geral, livros avançados da área de computabilidade e complexidade apresentam este enfoque [Papadimitriou, 1998, Davis and Weyuker, 1983].

Considerando que este estudo deve ser uma meta da disciplina, ou um ponto de partida para uma disciplina mais avançada na área de complexidade, é importante que os fundamentos computacionais sejam formalizados através de um modelo computacional acessível e completo. Logo, uma visão *top-down* da disciplina pode ser apresentada a partir do conceito de *problema*. Isto é, propõe-se que a disciplina de TEC seja baseada no conceito de solução de um problema através da computação em um determinado modelo.

Assim, a estrutura proposta para a disciplina é ilustrada pela figura 1, onde o *problema* é o ponto de partida. Assim, os problemas apresentam dificuldades ou complexidades intrínsecas, as quais levam a uma solubilidade ou não segundo critérios matemáticos. Para esta solução, deseja-se uma solução em hardware e/ou software. A implementação da solução necessita de um modelo computacional, o que leva ao conceito de algoritmo (procedimento forte) e, a partir da Tese de Church-Turing [Sipser, 1996], a outros modelos para esta computação, equivalentes computacionalmente [Marta Sagastume, 2003].

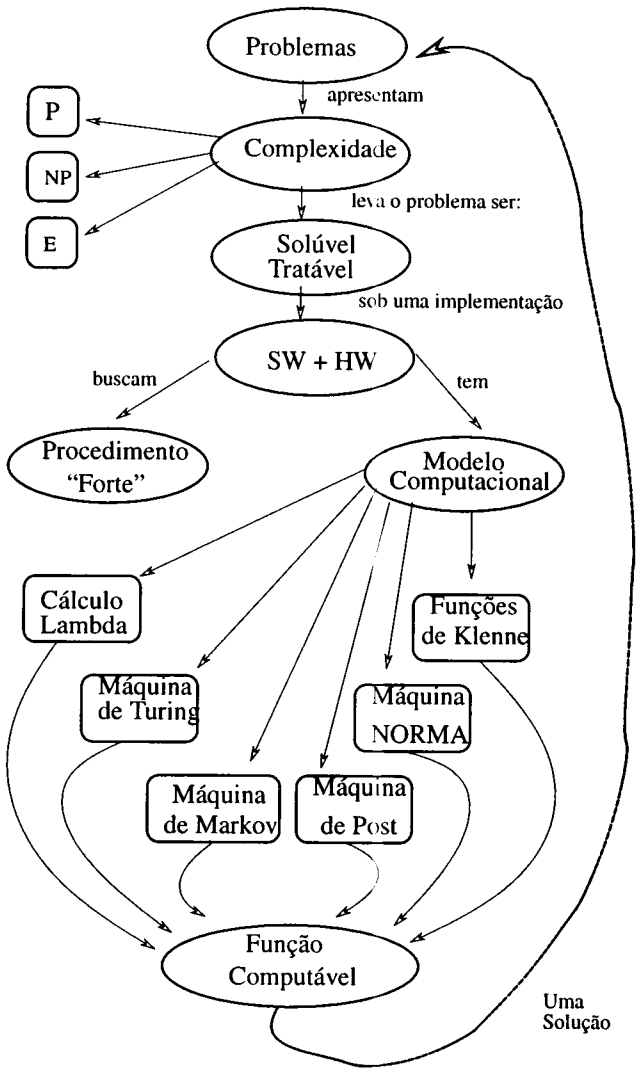

Figura 1: O problema e suas implicações

3.1. Objetivos

Os objetivos gerais e os pontos a serem evidenciados são os seguintes:

- Geral: Apresentar o conceito de algoritmo a partir de modelos formais, definindo assim computabilidade e seus limites na resolução de problemas;
- Uma proposta de ementa: Máquina de Turing. Formalização do conceito de algoritmo. Funções computáveis e a existência de funções universais. Computabilidade. Problema da Parada. A existência de problemas indecidíveis. Noções de Lambda Calculus e funções recursivas. Máquinas de Post, Markov e NORMA. A computabilidade sobre esses modelos. Relações entre os modelos de computabilidade. Complexidade. Classes de Problemas.

3.2. Conteúdo

O programa da disciplina de TEC, com uma duração de 60 a 75 horas, segundo a estruturação proposta pela figura 1, apresenta o seguinte encadeamento de assuntos e conteúdos:

Capítulo 1 (4 a 8 hrs/aula) Introdução

Conteúdo: Contextualizar a disciplina na área de CC e sua relação com disciplina afins, tais como Linguagens Formais, Compiladores, Lógica, etc. A disciplina no currículo e sua integração com outras disciplinas. A disciplina na formação do profissional. Histórico da evolução da computação (não do computador). Uma retrospectiva da evolução matemática que vai se consolidar nos formalismos necessários aos modelos computacionais propostos, mais especificamente a álgebra abstrata do século XIX. Os 23 problemas de David Hilbert e a introdução do termo *procedimento efetivo* em 1900. A formalização da lógica dos predicados por Gotlob Freege e a evidência de que esta teria problemas de inconsistências. Os paradoxos lógicos levantados por Bertrand Russel. O golpe na matemática com os Teoremas da Incompletude de Gödel. O fim da expectativa de que a lógica seria um método dedutivo perfeito para representar qualquer mecanismo formal. O abandono da busca de métodos matemáticos perfeitos. O surgimento de modelos/máquinas matemáticos que evidenciam o conceito de funções computáveis.

Comentário: um aluno de graduação, em geral, não vai estar disponível ou interessado em memorizar tantos dados históricos. Mas, o professor deve tentar motivá-lo a respeito da importância histórica dos conceitos teóricos que deram orgigem às principais idéias que fundamentam a matemática abstrata do fim do século XIX e início do século XX, que levariam ao conceito de algoritmo, computador, etc.

Capítulo 2 (12 a 16 hrs/aula) Máquinas de Turing

Conteúdo: Apresentar e discutir o conceito de *computação* com um modelo formal clássico, no caso as Máquinas de Turing (MT). Enfatizar a importância e o contexto, histórico e matemático, deste modelo computacional. Definir a Máquina de Turing padrão e sua séptupla. Exemplificar outras aplicações a partir da MT

padrão. Uso das MTs como aceitadores de linguagens, MTs multi-cabeças, MTs com fitas infinitas, MTs multi-fitas, MTs não-determinísticas, MTs como enumeradores de linguagens. Representação de problemas com sob uma fita de entrada. A Tese de Church-Turing.

Comentários:

1. Neste momento da disciplina, se faz necessário uma motivação prática ao aluno. Logo, o professor pode passar exercícios clássicos para serem implementados em simuladores de MTs. Exemplo: implementar a função soma e multiplicação unária, e avaliar as estratégias dessa programação no tocante a:
   - ✓ Uso da técnica de marcadores afim de efetivar a idéia de *contadores* numa MT;
   - ✓ Uso de movimentações do cabeçote;
   - ✓ O procedimento recursivo envolvido nos cálculos;
   - ✓ Uso do *read write* em analogia aos computadores atuais.
2. Uma prova indispensável a ser apresentada é a equivalência computacional de uma MT não-determinístca e de uma MT determinística. Esta prova pode ser feita após a apresentação da MT multi-fitas.
3. Um dos objetivos da disciplina deve ser evidenciado: o conceito de *algoritmo* e de *computação*. Ambos oriundos do modelo da MT, como base dos computadores reais.
4. O professor deve discutir o poder da representação de problemas sob uma sequência de símbolos. O livro de Sipser [Sipser, 1996], páginas 145–146, discute o problema da conectividade entre os nós de um grafo. A partir de um exemplo como este, o professor pode discutir a Tese de Church-Turing.

Capítulo 3 (8 a 10 hrs/aula) Decidibilidade

Conteúdo: Problemas de decisão em linguagens regulares e livres de contexto. O Problema da Parada da Máquina de Turing. O Método da Diagonalização. Uma segunda prova da indecidibilidade do problema da parada pela Diagonalização de Cantor (ver [Sipser, 1996], páginas (149–167). Linguagens não-reconhecíveis pelas MTs.

Comentários: Assim como o capítulo anterior tinha sua ênfase no importante resultado representado pela Tese de Church-Turing, neste capítulo a ênfase é no Problema da Parada, seus exemplos e consequências para CC. Exemplos como os citados por O.J.V. Furtado [Furtado, 2003]:

1. *Dado um programa qualquer, ele sempre tem parada garantida?*
2. *Dois programas P_1 e P_2 são equivalentes entre si?*
3. *Uma determinada solução é a melhor solução para um dado problema?*
4. *Qual o significado de um determinado programa?*
5. *Dado um programa qualquer, este programa está correto?*

Cuidar para que os exemplos sejam reais e próximo aos alunos, o que é uma tarefa dificíl.

Capítulo 4 (6 a 8 hrs/aula) Redutibilidade

Conteúdo: Conceituar e Exemplificar redutibilidade. Problemas indecidíveis da Teoria de Linguagens. Reduções via históricos da computação. Mapeamentos da redutibilidade. Máquina Universal. Exemplo de um problema indecidível (clássico): Problema da Correspondência de Post (PCP).

Comentários: Neste momento é normal que se instaure uma certa decepção junto aos alunos em face da quantidade de problemas indecidíveis da teoria de linguagens. Logo, o curso chega ao seu limite de formalismos e com um clima de: *logo agora que tudo ia tão bem*!
Cabe ao professor extrapolar para outras áreas sobre os problemas indecidíveis, como por exemplo: *Dado um inteiro I, determinar se existe um número perfeito maior que I (obs: um número é perfeito se a soma de seus divisores (exceto ele mesmo) é igual ao próprio número)* [Furtado, 2003].

Capítulo 5 (12 a 16 hrs/aula) Outros Modelos Computacionais

Conteúdo: Teoria das Funções Recursivas (Funções de Kleene). λ-Calculus. Sistemas de Produção (Máquina de Post). Sistemas de Reescritas (Máquina de Markov). Máquina NORMA. Estes modelos tem uma capacidade de computação equivalente à da Máquina de Turing.

Comentários: A atmosfera *cinza* instaurada no capítulo anterior é clareada ao mostrar outros modelos computacionais, pois, em geral, os alunos ficam aborrecidos com o excesso de provas e demonstrações.

Capítulo 6 (4 hrs/aula) Teoria da Complexidade

Conteúdo: Complexidade de tempo. Aceleração linear. Ordens de complexidade (cotas superior, inferior e média). Complexidade de espaço.

Comentários: Este assunto é visto sob diversos enfoques em várias disciplinas da área. Eventualmente o professor pode se concentrar em apenas dois conceitos com os alunos, e que não serão vistos nas demais disciplinas, são eles:
✓ Complexidade de tempo em MT;
✓ Complexidade de espaço em MT;
Ver capítulos 7 e 8 do livro do Sipser [Sipser, 1996].

Capítulo 7 (6 a 8 hrs/aula) Tratabilidade

Conteúdo: Problemas polinomiais determinísticos (Classe P). Problemas polinomiais não-determinísticos (Classe NP). Problemas NP-Hard e NP-Completos. Exemplos de problemas NP.

Comentários:
✓ O conteúdo deste capítulo pode ser apresentado junto com o conteúdo do capítulo anterior, de maneira análoga ao capítulo 7 do livro de

Sipser [Sipser, 1996]. Outros livros são complementares, alguns destes: [Bittencourt, 1996, ?, Laurière, 1990].

✓ Após uma definição mais precisa sobre complexidade espacial e temporal com a visão de MT, o professor deve retomar a proposta inicial do curso que era de questionar o quão solúvel são os problemas segundo um modelo formal de computação. Uma taxonomia é feita a partir das complexidades levantadas anteriormente, classificando os problemas em tratáveis e intratáveis.

3.3. Metodologia de Ensino

Além das necessárias aulas expositivas, com o uso eventual de retroprojetor, quando os diversos formalismos devem ser introduzidos e as provas dos resultados importantes apresentadas, propõe-se que se faça um uso intensivo de simuladores (MTs, sistemas de produção, λ-cálculo, etc.). O uso de simuladores permite que o aluno perceba as semelhanças e diferenças entre os modelos formais e os computadores reais e suas linguagens de programação. Existem disponíveis na Internet um grande número destes simuladores.

Propõe-se também que a avaliação se dê através de provas escritas e individuais, onde os conteúdos teóricos sejam solicitados, e exercícios práticos em grupo, onde os alunos devem resolver problemas utilizando simuladores de modelos formais.

Um exercício interessante é a simulação de um formalismo utilizando outro formalismo, por exemplo a implementação de uma MT em um simulador de λ-cálculo.

Propõe-se ainda a existência de uma página na Internet para a disciplina, onde os alunos podem enviar mensagens para o professor, encontrar ponteiros para textos que tratem de assuntos relacionados à disciplina, principalmente seus aspectos multidisciplinares, e para os programas simuladores, que podem assim ser instalados em computadores pessoais.

4. Reflexões Finais

Este artigo busca estruturar uma proposta de conteúdo programático para a disciplina de Teoria da Computação. Esta estruturação está focada sobre os conceitos de *problema* e *tratabilidade*. A idéia é motivar a necessidade de estudar os conceitos ligados a computação (algoritmo, decidibilidade, complexidade, etc.) de maneira formal, utilizando os problemas como um "pano de fundo"para introduzir os modelos computacionais.

Considerando o conteúdo proposto, os seguintes pontos devem ser destacados:

✓ Eventualmente o programa da disciplina pode tornar-se um tanto extenso. Neste caso, cabe ao professor focar a disciplina, omitindo questões menos relevantes, discutindo algumas provas ao invés de demonstrá-las, e ainda, fazer uso de recursos de multimídia;

✓ O professor deve apresentar uma visão ampla de como os tópicos da área da Teoria da Computação se encadeiam. Haja visto que a bibliografia é diversificada em seus propósitos. A ementa e conteúdo aqui apresentados se baseiam fortemente no livro de Sipser [Sipser, 1996]. Contudo, este tem pontos que não são tratados, como: Teoria das Funções Recursivas (Funções de Kleene), λ-Calculus, Sistemas

de Produção (Máquina de Post), Sistemas de Reescritas (Máquina de Markov) e Máquina NORMA. Logo, há uma dificuldade em se aplicar um único livro texto;

Enfim, a disciplina trata de conteúdos que têm uma relação indireta ou fraca com a parte aplicativa da informática e por isso, o aluno pode ser levado a rejeitá-la. Mas cabe ao professor fazer a ligação entre problemas fundamentais da ciência, muitos sem solução prática, com questões reais do dia-a-dia do aluno. Ou seja, ao final da disciplina, o aluno deve estar apto a diferenciar problemas que admitam soluções, daqueles já provados como indecidíveis. Isto é, sem soluções genéricas e irrestritas.

Referências

Aczel, A. (2003). *O Mistério do Alef A Matemática, a Cabala ea Procura pelo Infinito.* Editora Globo S.A. – São Paulo. título original *The Mystery of the Aleph.*

Arbib, M. A. (1969). *Theories of Abstract Automata.* Prentice-Hall Series in Automatic Computation. Prentice-Hall Inc., Englewood Cliffs, New Jersey.

Berlinski, D. (2002). *O advento do algoritmo: a idéia que governa o mundo.* Editora Globo S.A. – São Paulo. Trad. Leila Ferreira de Souza Mendes, título original *The Advent of the Algorithm.*

Bittencourt, G. (1996). *Inteligência Artificial - Ferramentas e Teorias.* 10a. Escola de Computação. Editora da Unicamp. Instituto de Computação - UNICAMP.

Boyer, C. B. (1991). *História da Matemática.* Editora Edgard Blücher LTDA., Rua: Pedroso Alvarenga, 1245 - cj. 22, São Paulo - SP, 2a. edição edition. Revista por Uta C. Merzbach. Título original: "A History of Mathematics".

Brookshear, G. J. (1989). *Theory of Computation: Formal Languages, Automata, and Complexity.* The Benjamin/Cummings Publishing Company, United States.

Davis, M. D. and Weyuker, E. J. (1983). *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science.* Academic Press, San Diego, CA.

Divério, T. A.; Menezes, P. F. B. (1999). *Teoria da Computação: Máquinas Universais e Computabilidade.* Sagra-Luzzatto, . Porto Alegre - RS. Série Livros Didáticos do Instituto de Informática da UFRGS, n.5.

Engeler, E. (1973). *Introduction to the Theory of Computation.* Academic Press, New York.

Furtado, O. J. V. (2003). Apostila de linguagens formais e compiladores. Com autorização de uso do autor, olinto@inf.ufsc.br.

Gersting, J. L. (2001). *Fundamentos Matemáticos para Ciência da Computação.* LTC - Livros Técnicos e Científicos Editora S.A., Travessa do Ouvidor, 11. Rio de Janeiro - RJ. 4a. Edição, do original: *Mathematical Structures for Computer Science* de 1999.

Hein, J. L. (1999). *Theory of Computation: An Introduction.* Jones and Barlett Plubishers, United States – USA.

Hoftstadter, D. R. (1999). *Gödel, Escher, Bach: An Eternal Golden Braid.* Basic Books, Inc., New York, 20th. anniversary edition edition. A metaphorical fugue on minds and machines in the spirit of Lewis Carroll. Pulitzer Prize Winner.

Hopcroft, J. and J.D., U. (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company, Reading. Massachusetts (2nd. edition).

Laurière, J.-L. (1990). *Problem Solving and Artificial Intelligence*. Prentice-Hall. First published in French in 1987 by Eyrolles, Paris.

Marta Sagastume, Gabriel Baum, G. M. (2003). *Problemas, Lenguages y Algoritmos*, volume 37. Publicações CLE - UNICAMP - BR. Uma 1ª. versão da I EBAI - 1988.

McCarthy, J. (1963). A Basis for a Mathematical Theory of Computation. In Braffort, P. and Hirschberg, D., editors, *Computer Programming and Formal Systems*, pages 33–70. North-Holland, Amsterdam.

Moret, B. (1998). *The Theory of Computation*. Addision Wesley, New York.

Papadimitriou, H. R. L. C. H. (1998). *Elementos de Teoria Da Computação*. Editora Artes Médicas do Sul LTDA. (Bookman Companhia Editora), Av. Jerônimo Ornelas, 670 - Porto Alegre - Br, 2a, edição edition. Título original: "*Elements of theory of computation*". Editora original: Prentice-Hall, Inc.

Penrose, R. (1991). *A Mente Nova do Rei – Computadores, Mentes e as Leis da Física*. Editora Campus. Tradução de Waltesir Dutra.

Sipser, M. (1996). *Introduction to the Theory of Computation*. PWS Publishing Company, 2nd. edition edition. Errata url: http://www-math.mit.edu/~sipser/itoc-errs1.2.html.