RCT

Vol. I
Nº. 1

A Teoria da Computação e o profissional de informática

João José Neto

Resumo

Discute-se neste trabalho a importância do domínio do conhecimento e da fluência em assuntos teóricos e abstratos como parte da bagagem conceitual dos profissionais da área de Informática. Apresenta-se um quadro panorâmico geral da área, tecendo-se comentários sobre diversos assuntos teóricos de interesse, procurando-se correlacioná-los e mostrar a sua relevância para o exercício competente da prática profissional.

Palavras-chave: *informática; Computação; Teoria da Computação; abstrações; ciência; tecnologia; fundamentos; prática; aplicação.*

Abstract

This paper discusses the importance of mastering and having detailed domain of theoretical and abstract knowledge as a part of the conceptual background for professionals in the field of informatics. A broad landscape of this field is shown, a diversity of interesting theoretical subjects are commented and interrelated, and their relevance for a competent professional practice is emphasized.

Keywords: *informatics; Computation; Theory of Computation; abstraction; science; technology; foundations; practice; applications.*

Escola Politécnica da USP — PCS

Departamento de Engenharia de Computação e Sistemas Digitais

Av. Prof. Luciano Gualberto,
Travessa 3, n. 158
Cidade Universitária
CEP 05508-900
São Paulo — SP

joao.jose@poli.usp.br

1 Introdução

Nos dias de hoje, em que o imediatismo domina a nossa sociedade, permeando muitas esferas sócio-culturais, e influindo direta e significativamente nos rumos de toda a humanidade, diversos aspectos da formação dos profissionais da área têm sido alvo de discussões e de mudanças, visando sua adequação aos nossos tempos.

Em particular, tem sido muito questionada a conveniência e até mesmo a necessidade de se estudar, conhecer e dominar assuntos abstratos, dado que o exercício profissional geralmente praticado induz ao uso mecânico e quase cego de elementos pré-fabricados, "de prateleira", parecendo até muito estranho discutir a "eventual" importância para o profissional de um reforço de formação em conhecimentos complexos, exigentes e abstratos, como aqueles propostos pela Teoria da Computação.

Identificam-se aí duas tendências tecnológicas que, embora complementares, infelizmente costumam ser interpretadas como sendo radicalmente excludentes: a primeira, sugerindo que se busque preferencialmente o (re)aproveitamento de produtos e resultados de esforços anteriores como forma de obtenção de recursos humanos produtivos e eficientes, e a segunda, priorizando o cultivo da formação científica como meio de formação de profissionais preparados para o desenvolvimento de trabalhos centrados na criatividade.

Naturalmente, sempre que levadas aos limites, tendências como essas geralmente acabam por contrapor-se, suscitando, de um lado, adeptos extremos da exploração direta de capacitações já consolidadas, para o uso imediato de soluções previamente desenvolvidas, e de outro, aqueles que radicalmente consideram legítimas apenas as tendências estritamente investigativas, puramente teóricas, talvez sem compromisso com a realidade.

Aqueles se mostram quase sempre refratários a promoverem investimentos na formação da componente conceitual dos profissionais, e mais favoráveis a investimentos de rápido retorno, que em geral correspondem a um preparo mais superficial e utilitarista, menos formal, mais informativo e preponderantemente tecnológico.

Os outros, por sua vez, por valorizarem mais a busca de conhecimentos novos, acabam priorizando a pesquisa em áreas pouco exploradas, o que exige investimentos de risco, "a fundo perdido", em pesquisa pura, os quais raramente apresentam garantias absolutas de retorno do investimento, de resultados dos desenvolvimentos na forma de produtos úteis à sociedade, do cumprimento dos prazos inicialmente estimados.

Como diz a sabedoria popular,"a virtude nunca está nos extremos", e assim, torna-se conveniente que seja identificado aquilo que cada vertente tem de melhor para oferecer, buscando-se entre elas um equilíbrio aceitável, de forma que possam cada qual contribuir a seu modo para uma formação não-polarizada de um profissional que, apoiado na solidez que lhe pode conferir o conhecimento profundo das bases fundamentais da Computação, seja capaz de, no dia-a-dia, optar, de forma competente, entre a invenção de novas soluções e a adequação, aos interesses de cada momento, de soluções previamente estabelecidas.

1.1 Motivação

Pode para alguns parecer anacrônico, na nossa era dos produtos prontos para o consumo, que conhecimentos como os oferecidos pela Teoria da Computação sejam propostos como instrumentos de trabalho para pessoas cujas formações se revelam tão diversificadas, e cujos interesses, são mesclados com os de outras áreas, como é o caso de tantos profissionais da Informática, cujos alvos mais diretos de interesse são os computadores e a computação.

Não faltaram nem faltarão argumentos de apoio ao progressivo esvaziamento de conteúdo das já tão enfraquecidas disciplinas fundamentais dos currículos de formação dos profissionais da área, em nome do pouco interesse prático que se alega existir por conhecimentos que, de tão especializados e complexos, conseguem fascinar apenas uma pequena elite.

Nem faltam justificativas para a adoção de critérios cada vez mais restritivos de apoio ao desenvolvimento da dispendiosa, demorada e incerta pesquisa básica em tantas universidades de todo o planeta, em nome da priorização daquelas pesquisas ditas "aplicadas" e "sustentáveis", potencialmente lucrativas, em vista dos resultados que podem produzir em menor prazo e com menor custo.

Argumentos dessa natureza infelizmente convencem e empolgam a muitos, consequentemente favorecendo, em vários níveis, tomadas de posturas que, em nome dos resultados imediatos e de uma tão desejada "sustentabilidade", acabam eclipsando os efeitos de sua adoção no longo prazo, muitos deles adversos, perigosos, inevitáveis, e possivelmente irreversíveis, pelas restrições que impõem às ideias e à criatividade.

Este quadro, tão hostil e tão presente nos nossos dias, motiva uma reflexão profunda acerca da importância e da necessidade de um rápido retorno ao incentivo aos estudos e à pesquisa de base, sem objetivos especulativos ou lucrativos, e que não discriminem, em suas metas, os temas mais abstratos da Teoria da Computação, os assuntos emergentes ou desconhecidos, que tipicamente não costumam estar atrelados à produção imediata de resultados ou de benefícios materiais.

Tal postura, vital para a soberania da nação no que tange à sua capacitação técnico-científica, visa suprir os subsídios essenciais à manutenção

da produção científica de vanguarda, bem como para a formação acadêmica e profissional autônoma daqueles que trabalham na área, cujo papel se mostra absolutamente essencial à gênese não apenas dos fundamentos das futuras tecnologias como também, principalmente, à preparação das próximas gerações de profissionais que, ao seu tempo, deverão estar aptos a capacitar outros, e a conduzir esses complexos desenvolvimentos sem recorrer compulsoriamente à importação de profissionais, a qual nesse quadro certamente terá sido provocada pela então provável deficiência da capacitação local.

Considerando-se a importância que ganha a cada momento o aspecto da qualidade, em suas mais variadas manifestações, levando-se em conta a existência do diversificado arsenal de ferramentas que nos oferecem as teorias conhecidas, muito poderosas e aplicáveis às mais diversas situações do dia-a-dia, pode-se constatar, pela simples observação, que muito pouca ênfase se tem dado, por ocasião da formação acadêmica do profissional, a um sério trabalho explícito de conscientização, tanto aos alunos como aos professores, relativo a toda essa conjuntura.

Torna-se, dessa maneira, muito importante formar uma consciência bem fundamentada do valor que pode agregar, tanto à qualidade do profissional de Informática como à dos produtos por ele desenvolvidos, o conhecimento profundo dos assuntos teóricos que constituem os fundamentos conceituais e científicos da Computação.

O mesmo se pode dizer em relação às indiscutíveis prerrogativas técnicas de que gozam os profissionais que amplo domínio possuam sobre esses temas em relação àqueles que os desconhecem ou que deles prescindem, pois isso também invariavelmente se reflete — nem sempre de forma imediata, porém previsível e inexorável — na qualidade apresentada pelos resultados do trabalho de cada um.

Através da identificação e da análise de diversos pontos, considerados essenciais à formação de bases firmes para a conscientização acerca da importância de uma sólida fundamentação teórica para os profissionais de Informática, o presente material pretende constituir uma pequena contribuição, tanto para reforçar convicções sobre esse fenômeno, como para promover uma possível mudança de postura em relação a esse tema que, embora para alguns se mostre bastante óbvio, explícita ou veladamente, com frequência para outros se constata tão desnecessariamente controverso.

1.2 Organização deste artigo

Após essa motivação, apoiada na observação e na análise de alguns fatos da realidade atual do mundo da Informática, o presente texto volta-se a uma discussão técnica destinada a levantar os principais pontos em que se cruzam as vias da teoria e da tecnologia.

Para as diversas situações assim identificadas, procura-se apresentar as principais razões de convivência das duas frentes, citando-se, quando oportuno, opiniões importantes a respeito de tais assuntos, emitidas por personagens ilustres da história da Computação.

Para tanto, temas-chave da Teoria da Computação são identificados, delineando-se suas dependências relativas e suas aplicações, e analisando-se a sua importância prática,para que se possa constatar mais facilmente a relevância que apresenta, para a qualidade profissional de cada um, o domínio e a fluência em cada um dos aspectos do estudo de tais assuntos.

Conclui-se o artigo com um convite a uma séria reflexão acerca do valor da Teoria da Computação na formação acadêmica dos futuros profissionais das inúmeras áreas de especialidade em que se subdividem as atividades da Informática.

2 Os Aspectos Teóricos da Computação

Apresenta-se, nesta parte do artigo, a estrutura do ambiente contextual histórico e técnico em que se desenvolve o tema de interesse central da presente matéria.

Em primeiro lugar é considerada a área mais vasta e abrangente do conhecimento, que se refere ao estudo da informação e da sua manipulação automática, e que é conhecida pelo nome de *Informática*.

Como parte nobre da Informática, identifica-se a *Ciência da Computação*, importante membro de uma grande diversidade de compartimentos do conhecimento humano associados à especialidade, os quais envolvem, entre outros, assuntos relacionados com modelagens, métodos, cálculos, análises, teorias, etc.

Analisando-se um pouco mais de perto a Ciência da Computação vislumbra-se, entre os muitos assuntos tratados, a *Teoria da Computação* [1, 2], o mais importante alicerce de suas bases conceituais, em que estão apoiados todos os ramos da Computação. Por essa razão, reconhece-se na Teoria da Computação um dos mais importantes campos de conhecimento fundamental, do qual dependem fortemente todas as demais áreas.

Sem pretensões de exaurir o assunto, busca-se neste artigo apontar uma coleção significativa de assuntos tratados na Teoria da Computação, identificando-se o inter-relacionamento existente entre diversos dos temas abrangidos, e realçando-se, quando possível, a importância de seu estudo mediante a indicação do uso que na prática se tem feito de cada um dos grupos de assuntos teóricos.

2.1 A Informática

Em toda a história, constata-se que, antes mesmo que determinadas questões, algumas das quais bastante complexas, tenham sido sequer formuladas pela primeira vez, os cientistas costumam se antecipar apresentado para elas, diversas respostas interessantes, elegantes e criativas, como resultados de suas pesquisas.

Embora nem fosse possível imaginar então a futura existência de computadores digitais, e menos ainda, que aspecto teriam ou qual importância eles viriam a ter nos dias de hoje, já nos meados do século XIX George Boole brindava a Ciência da Computação com a imensa contribuição representada pela famosa álgebra que, com justiça, leva seu nome.

Com as funções por ele idealizadas, tornou-se possível criar modelos precisos, tanto do funcionamento das volumosas, lentas e dispendiosas redes de circuitos lógicos eletromecânicos, utilizadas nos primeiros computadores, como igualmente dos ágeis, compactos e econômicos circuitos digitais microeletrônicos modernos.

O advento da modernidade trouxe a industrialização, exigindo progressos técnicos que permitissem às máquinas resolver problemas de forma melhor, mais rápida e mais confiável que os operários. Surgiram então os primeiros algoritmos, na forma de receituários que visavam preparar tais máquinas para o processamento automático de certos tipos de informação.

Importantes nomes apareceram, tais como Turing, Gödel e Church, contribuindo decisivamente nessa fase embrionária da Informática através de sua marcante atuação, referente à investigação da viabilidade ou não de se resolver certas classes de problemas por aplicação mecânica e sequencial de uma restrita variedade de operações muito elementares.

Na década de 1950, Chomsky, investigando abstrações para uso em linguística, plantou os alicerces teóricos das gramáticas gerativas, que mais tarde vie-ram a manifestar sua grande utilidade prática nessa e em diversas outras aplicações, com destaque à análise léxica e sintática de linguagens de programação, à modelagem do comportamento de organismos biológicos, ao projeto de hardware e ao processamento de linguagem natural.

Esses desbravadores anteviram, pesquisaram e deram resposta — algumas, muito antes da concepção dos primeiros computadores — a variadas e complexas questões, abrindo um caminho bem fundamentado e seguro não somente para o surgimento como também para a evolução que conduziu a Informática ao estado em que hoje se encontra.

Esta é sem dúvida uma significativa manifestação histórica da importância de um estudo teórico, profundo, da natureza de um problema computacional, precedendo atividades de implementação e até mesmo da idealização de aparatos físicos capazes de resolver automaticamente o problema em questão.

Os resultados pioneiros de pesquisas como as de Boole, Turing, Gödel, Church, Chomsky e de tantos outros, apresentados com tanta antecedência em relação ao advento das tecnologias que os viriam a utilizar, revelaram-se verdades gerais, fundamentais, independentes da tecnologia, imunes ao tempo e às áreas de aplicação, refratárias à moda, à política, às preferências e ao mercado, extremamente poderosas e perenes, continuando por essa razão em pleno uso, incontestes, válidas e atualíssimas, até os nossos dias.

2.2 Informática e a Ciência da Computação

A Informática dedica-se ao estudo do processamento lógico e automático da informação, que em geral é atualmente realizado com a ajuda de computadores digitais. Manifesta-se preponderantemente no estudo e desenvolvimento de computadores, de seus componentes mecânicos e eletrônicos, e de seus programas, bem como em diversos aspectos da concepção, realização e uso de linguagens de programação, tecnologias de desenvolvimento e softwares de aplicação.

Nos dias de hoje, a Informática se mostra virtualmente onipresente, manifestando-se em bancos, no comércio e na indústria, em componentes microeletrônicos embutidos em eletrodomésticos, na etiquetas de produtos, em telefones celulares, automóveis, jogos eletrônicos, instrumentos de medida, sistemas de comunicação, câmaras fotográficas, equipamentos de laboratório, robôs, dispositivos de sinalização, de aquisição de dados, de sensoriamento remoto e tantos outros.

É nessa vasta área que podem ser localizadas a Teoria da Informação, a análise numérica, a representação do conhecimento, a modelagem de problemas, os métodos teóricos e formais, os processos de cálculo, e entre todas essas, também a Ciência da Computação, alvo da presente publicação.

2.3 A Computação

O termo *computação* costuma ser empregado para designar o uso de algoritmos para a resolução de problemas. Como se diz formalmente, esse vocábulo alude à execução de algoritmos como meio para realizar o cálculo de funções. Foi praticada por milênios no passado, mentalmente ou por escrito, muitas vezes com o auxílio de tabelas.

Ramo da Matemática e da Ciência da Computação, a Teoria da Computação surgiu no princípio do século XX, antes da invenção dos computadores, e estuda modelos formais de computação, sua aplicabilidade e sua viabilidade prática à resolu-

ção das diversas classes existentes de problemas.

Desse estudo emergiram propostas, na forma de dife-rentes modelos de computação, representados através de diversificadas abstrações, apresentadas em variadas notações. Segundo a conjuntura denominada *Tese de Turing-Church*, todos esses modelos se equiparam uns aos outros quanto ao seu poder computacional, além de serem todos também equivalentes a um computador hipotético que oferecesse uma disponibilidade ilimitada de memória:

- A *Máquina de Turing*, que foi proposta como modelo universal de computação, e trabalha com operadores muito rudimentares sobre instruções e dados gravados em uma fita de trabalho de comprimento infinito. Dos modelos universais de computação existentes, talvez seja o mais conhecido e famoso, e é bastante aderente ao estilo de programação representado pelo paradigma imperativo;
- O *Cálculo Lambda* [3], formalismo que foi proposto e empregado na representação de programas desenvolvidos segundo o paradigma de programação adotado pelas linguagens funcionais, e que se mostra muito adequado para a representação formal de fenômenos computacionais ligados a linguagens de programação declarativas;
- As *Funções Recursivas* [2], utilizadas em Matemática desde épocas muito anteriores à dos computadores, podem também servir como um modelo computacional inspirado naquela ciência, e, ao contrário da maioria dos outros modelos em uso, têm a grande vantagem de permitir a representação e a manipulação direta de valores numéricos;
- As *Gramáticas Gerativas* [4] permitem a representação de linguagens através do uso de regras de substituição, que operam sobre sequências de símbolos, sendo muito empregadas na especificação e na representação de linguagens artificiais,como as de programação e de outras notações constituídas de cadeias de caracteres com lei de formação conhecida, tornando natural a formalização e a manipulação algorítmica de textos simbólicos.
- Os *Dispositivos Adaptativos* [5], particularmente aqueles baseados em autômatos, são modelos de computação capazes de representar fenômenos computacionais complexos através de quaisquer abstrações cujo comportamento seja descrito através de um conjunto de regras dinamicamente variável. Essa dinâmica se obtém associando-se à aplicação de cada regra uma ação que especifica alterações a serem realizadas sobre o conjunto de regras. Assim,ao executarem essas *ações adaptativas*, tais dispositivos podem ir alterando, de forma autônoma, seu próprio comportamento,em função do seu histórico de funcionamento. Com isso, um dispositivo adaptativo torna-se equivalente à Máquina de Turing, portanto poderá represen-tar o conhecimento adquirido em sua operação, tendo, no entanto, sobre a Máquina de Turing, nesse aspecto, a vantagem de concentrar, de forma localizada, em seu próprio conjunto de regras, a representação de tal conhecimento. Assim, torna-se possível identificar e acompanhar até mesmo de forma incremental a aquisição do conhecimento por um dispositivo adaptativo, através da inspeção do seu conjunto de regras e da sua variação toda vez que, durante sua operação, alguma ação adaptativa for executada. Por essa e diversas outras razões, os dispositivos adaptativos têm sido considerados como alternativas atraentes para a representação de fenômenos de aprendizagem, na área de inteligência artificial.
- Muitos outros modelos de computação têm sido propostos e podem ser encontrados com facilidade na literatura, entre os quais podem ser destacados: os *Sistemas de Post*, as *Cadeias de Markov*, as *máquinas de Mealy* e *de Moore*, diversas outras variantes das *máquinas de estados*, dos *autômatos* e *transdutores finitos* e *de pilha*, *Máquinas de Turing Universais*, *máquinas virtuais* diversas, utilizadas na execução de diversas linguagens de programação (P-system, Java byte-code, etc), *Redes de Petri*, *Statecharts*, *máquinas probabilísticas*, e inúmeras outras.

A *Teoria da Computabilidade* investiga a possibilidade de máquinas computacionais e de certos formalismos teóricos adotados como modelos de computação apresentarem ou não a capacidade de realizarem automaticamente determinados tipos de computação. Em termos do que se conhece hoje dos computadores e da computação, uma versão dessa investigação pode ser traduzida na busca de respostas para a pergunta seguinte: *é possível ou não construir um algoritmo capaz de resolver de forma automática um dado problema através da execução iterativa de passos de instruções em um dado dispositivo computacional, como, por exemplo, um computador digital ou algum outro modelo computacional?*

Mais abrangente, a *Teoria da Computação* propõe, estuda e compara modelos de computação, as classes de problemas que cada um deles consegue resolver e os limites a que cada qual está sujeito. Ilustrando, podem ser citados alguns problemas para os quais é impossível criar programas de computador que lhes sirvam de solução. São esses os problemas ditos *incomputáveis*, entre os quais se destacam: o problema da parada da Máquina de Turing; o problema da correspondência de Post; determinar se a intersecção de duas

linguagens livres de contexto arbitrárias é também livre de contexto; determinar, para uma gramática livre de contexto sobre um alfabeto não-unitário, se a linguagem que ela representa é regular;determinar se uma gramática livre de contexto arbitrária é ambígua; determinar se uma linguagem livre de contexto arbitrária é inerentemente ambígua. Muitos outros problemas incomputáveis existem, mas felizmente constata-se a existência de outra infinidade de problemas computáveis, de grande importância prática, e isso motiva o seu estudo, do ponto de vista tanto teórico como prático.

Dos problemas computáveis, alguns se mostram impraticáveis por exigirem um tempo abusivo ou uma quantidade de memória excessiva para sua operação. Para os decidíveis é possível a construção de algoritmos que sempre terminam, quaisquer que sejam os dados a eles fornecidos. Para os indecidíveis, é possível elaborar procedimentos computacionais, porém estes terminam somente quando se lhes apresentam dados "corretos", aderentes ao que normalmente se espera, podendo, no entanto, iniciar um ciclo infinito de execução em resposta aos demais dados.

O estudo da decidibilidade de algoritmos data dos anos 1930, quando os estudiosos da área tentavam demonstrar ser inviável realizar raciocínios matemáticos com a ajuda de dispositivos automáticos. Um interessante problema nessa área é o das provas automáticas dos teoremas de uma dada teoria. É possível provar que o excessivo tempo de resposta de provadores de teoremas para determinadas teorias torna impraticável seu uso com fórmulas grandes. É o caso da *Aritmética de Presburger*, que se mostra decidível, porém duplamente exponencial em complexidade.

Nos casos de problemas de alta complexidade computacional, pode ser interessante determinar para quais situações vale a pena desenvolver uma solução algorítmica e para quais outras é mais conveniente efetuar verificações extensivas de propostas de solução. O estudo das classes P e NP de complexidade computacional costuma analisar famílias de problemas que ilustram bem esta situação.

Se de um lado a Teoria da Computação busca soluções gerais e grandiosas para classes amplas de problemas, por outro lado ocupa-se também de determinar modelos de complexidade mínima que resolvam satisfatoriamente situações menos gerais, de interesse prático.

Uma forma de avaliar o poder de expressão de um modelo computacional é através do estudo da classe das linguagens formais que esse modelo é capaz de representar; o resultado dessa prática é uma taxonomia baseada na hierarquia proposta por Chomsky para as linguagens formais.

Isso tem importantes aplicações práticas no estudo de linguagens, gramáticas e autômatos: o uso extensivo de expressões regulares para a especificação de padrões de busca em cadeias de caracteres levou diversas linguagens de programação e sistemas operacionais a oferecerem recursos nativos específicos com essa finalidade. É o caso de lingua-gens como Perl e Python, e de sistemas operacionais como UNIX, Linux e similares.

No âmbito das *linguagens regulares*, é notório o uso de autômatos finitos e de máquinas de estados finitos em um grande número de aplicações importantes, tais como no projeto de circuitos sequenciais, em alguns sistemas voltados à automatização da resolução de problemas, na construção de protocolos de comunicação, sequenciadores, controladores de interfaces Web, analisadores léxicos de compiladores e interpretadores para linguagens de programação, na especificação de sistemas reativos e de tem-po real, e até mesmo na especificação da lógica de determinados tipos de programas.

Linguagens livres de contexto, as respectivas gramáticas e os autômatos de pilha ocupam igualmente seu lugar na imensidão de aplicações que fazem uso das abstrações e dos resultados da teoria da computação. Assim, formalismos gramaticais livres de contexto têm sido usados extensivamente desde a época de sua concepção na representação parcial da sintaxe de linguagens de programação [4, 6, 7]. Autômatos de pilha podem ser empregados na representação de fenômenos linguísticos envolvendo aninhamentos sintáticos, e servem assim para modelar o comportamento de programas constituídos por um conjunto de procedimentos potencialmente interdependentes e mutuamente recursivos.

Menos explicitamente exploradas, as *linguagens dependentes de contexto* estão muito mais disseminadas no mundo da computação do que pode parecer à primeira vista. Linguagens de programação são, em sua quase totalidade, linguagens dependentes de contexto, para cuja representação as gramáticas livres de contexto e os autômatos de pilha se revelam insuficientes. Embora existam diversos modelos suficientemente expressivos para representá-las, seu uso costuma ser evitado por gerar formulações extensas, complexas e obscuras. Em lugar disso, costuma-se aproximar, na prática, as linguagens de programação por meio de modelos sintáticos simplificados, livres de contexto, os quais devem ser complementados por trechos de código, funções ou procedimentos externos, que se responsabilizem pela representação das dependências de contexto, ausentes na formulação simplificada. Esta tem sido prática corrente na construção de compiladores e de outros processadores de linguagens de programação.

2.4 A Ciência da Computação

A Ciência da Computação estuda os fundamentos e a prática das computações, investigando e explorando, através de algoritmos computacio-

nais, estruturas matemáticas que modelam fatos do mundo real, permitindo assim que processos computacionais sejam formulados de maneira precisa para então manipulá-los devidamente, atendendo as necessidades das aplicações desejadas.

Para atingir esses objetivos, oferece ao profissional uma série de contribuições teóricas: proporciona bases em Linguagens Formais, na Teoria dos Autômatos e em Complexidade Computacional, oferece métodos para que seja verificado se um requisito está sendo atendido ou não, disponibiliza técnicas para o desenvolvimento de modelos, e proporciona meios com os quais se pode avaliar o poder dos modelos adotados.

Cada vez mais potentes e módicos, os computadores digitais constituem os dispositivos tecnologicamente mais importantes que materializam fisicamente os modelos abstratos oferecidos pela Ciência da Computação. Ao mesmo tempo, os programas (softwares) que neles se executam constituem a melhor concretização das abstrações lógicas que com eles pode ser realizada no dia-a-dia.

Isso abre inúmeras possibilidades para a utilização prática de abstrações computacionais, quer através do emprego de linguagens de programação bem projetadas e aderentes à aplicação, quer para a construção de compiladores ou interpretadores, permitindo a disponibilização, nos computadores, dos recursos oferecidos por essas linguagens.

Os caminhos disponíveis para a concepção de todas essas abstrações e dispositivos tecnológicos, assim como os métodos utilizados para a avaliação das aplicações da vida real que com elas são desenvolvidas contam-se entre as principais contribuições de cunho prático proporcionadas pela Ciência da Computação, as quais dificilmente teriam surgido sem os importantes fundamentos teóricos que lhes deram origem.

Pode-se identificar que uma importante característica das teorias das quais se ocupa a Ciência da Computação é que elas se apóiam em bases matemáticas muito bem estruturadas, constituindo assim um sólido corpo de elegantes fundamentos, aplicáveis a todas as atividades da área.

A motivação maior do desenvolvimento deste ramo do conhecimento gira em torno da investigação do alcance das computações, da possibilidade ou não de se efetuarem determinadas computações, culminando no importante conceito de algoritmo, cuja possibilidade de existência é fortemente associada à viabilidade prática de materialização das computações realizáveis.

O estudo sério dos assuntos científicos da computação proporciona ao interessado um duradouro lastro de formação, de largo espectro, a respeito dos assuntos em questão, que lhe garante segurança conceitual ao longo de toda sua vida profissional.Tal base técnica dificilmente pode ser adquirida quando a formação do profissional estiver afastada da teoria, mas centrada na digestão frequente de generalidades, de atualidades e de informação de divulgação.

Embora para a área mercadológica, estratégica e empresarial, esse tipo de preparo seja essencial, para a formação de um profissional com perfil mais técnico tal esquema geralmente se traduz somente em uma cultura fugaz, específica e temporária, acerca dos fatos, do estado da arte e da tecnologia em moda em cada momento histórico, e isso costuma mostrar-se insuficiente para o exercício competente dos aspectos profissionais técnicos de alcance mais profundo.

Em lugar de simplesmente informar o interessado, preparando-o a produzir rapidamente para con-sumo imediato no mercado de trabalho, a formação acadêmica e profissional baseada em fundamentos sólidos é capaz de conceder-lhe algo muito mais versátil e perene: o embasamento de que necessita para dominar os assuntos que constituem as mais profundas raízes do conhecimento da área, que resistam ao tempo, à moda e à tecnologia.

Isso não deve ser entendido como uma apologia à alienação quanto a avanços tecnológicos, mas como um alerta para que os atraentes acenos do imediatismo não ofusquem o profissional exatamente naquele que seria seu melhor momento para priorizar sua formação conceitual em temas abrangentes e universais, reduzindo – sem anular – a prioridade da assimilação rápida de atualidades e da absorção imediata de treinamentos específicos em ferramentas tecnológicas particulares, de interesse imediato e efêmero, mas restrito e localizado.

Com uma formação menos restrita e específica, o profissional de computação sempre terá à sua disposição uma gama considerável de conhecimentos oriundos da Ciência da Computação, os quais virtualmente poderão ser utilizados no desenvolvimento de variadas aplicações, em qualquer área de sua atuação profissional, bastando para isso que, inicialmente, adquira habilidades para modelar adequadamente suas aplicações usando o ferramental teórico assim assimilado, de modo que possa, num segundo momento, convertê-los em programas executáveis, ou seja, representados na forma de algoritmos.

Aqui se pode avaliar bem a importância de um profissional da área conhecer e dominar profundamente, em teoria e na prática, todo o processo de elaboração de algoritmos, desde a sua concepção, passando posteriormente pelos procedimentos de elaboração, materialização, verificação, e implantação, tanto de algoritmos representados por pequenos trechos de código como dos atualmente frequentes sistemas de software, de porte cada vez maior.

Não basta que, durante o seu período de preparação acadêmica para o mercado, o profissional se detenha apenas no estudo de um ou outro desses elementos, pois todos eles se mostram essenciais à sua formação integral. Mostra-se, assim, de suma

importância que os programas dos cursos universitários da área levem em consideração essas ponderações, equilibrando a ênfase que dão a essas diversas componentes formativas de seus alunos.

2.5 A Teoria da Computação

No vasto campo da Ciência da Computação, pode-se dizer com segurança que a Teoria da Computação ocupa um merecido lugar de destaque, devido à sua importância como sustentáculo conceitual de todos os conhecimentos da área, tanto dos mais fundamentais e abrangentes como dos mais particulares e específicos [8–10].

Eliminadas as peculiaridades dos problemas específicos e a influência de idiossincrasias apresentadas pelas tecnologias específicas em uso, a Teoria da Computação investiga, estabelece e interpreta propriedades intrinsecamente associadas aos sistemas computacionais, daí tirando conclusões universais de ampla aplicabilidade, e que se provam independentes de casos e de especificidades.

Os resultados da Teoria da Computação, na sua maioria obtidos na primeira metade do século passado,comprovaram-se totalmente refratários à história. Ilustrando através de um caso particular, na área da programação pode-se dizer que tais resultados não foram afetados pelas evoluções tecnológicas refletidas no paradigma de programação adotado, valendo indiferentemente para programas desenvolvidos em linguagens funcionais, imperativas, lógicas, orientadas ou não a objetos, e explorando ou não o paralelismo ou a concorrência.

Devido à universalidade de que são dotadas, as poderosas ferramentas teóricas proporcionadas pela Teoria da Computação nem sempre são direta e imediatamente aplicáveis a casos particulares, a um produto especial ou a alguma tecnologia específica, e esse fato infelizmente leva muitos profissionais – por falta de base ou desinteresse – a afastar-se delas, a desconsiderar e até a desvalorizar sua importância.

Não é de se esperar que profissionais despreparados sejam capazes de apreciar o alcance dos recursos teóricos, pois para isso deveriam possuir habilidades que os capacitassem à análise de situações práticas, permitindo-lhe identificar, destacar e utilizar, dentre tantos resultados teóricos disponíveis, aqueles que se aplicam ao seu caso prático específico, mas isso só se consegue com um bom preparo técnico prévio, que lhes proporcione amplo domínio sobre os fundamentos dessa teoria.

Reconhecendo a imensa importância desse ramo do conhecimento, Lewis e Papadimitriou [2] acertadamente recomendam seu estudo acadêmico precoce, como forma de proporcionar não somente a assimilação de habilidades matemáticas, essenciais ao futuro profissional, como também as bases conceituais para um aproveitamento significativamente melhor de todas as disciplinas da área.

É preciso lembrar que, como seria de se esperar, a Teoria da Computação se fundamenta em métodos e técnicas da Matemática, em particular, na teoria dos conjuntos. Scheurer [11], alerta que se deveria dar mais atenção aos conhecimentos de teoria dos conjuntos ministrados aos futuros profissionais da área, de modo que estes possam empregá-los de forma tão natural quanto os programadores utilizam com fluência as suas linguagens de programação.

Recorda ainda esse autor que os processos de modelagem formal e matemática independem do uso ou não de computadores. Afirma ainda que, didaticamente, este importante fato deveria ser considerado com muita ênfase ao se ministrar conhecimentos básicos aos futuros programadores, permitindo-lhes assim assimilar em primeiro lugar os fundamentos, para, só depois de adquirida a devida habilidade de modelagem, começarem a utilizar o computador para a implementação dos modelos.

É instrutivo constatar que a Teoria dos Conjuntos e a Lógica Matemática funcionam como alicerces para muitas áreas da computação, merecendo destaque a engenharia de software, o software básico e as linguagens de programação, nos quais se pode identificar o uso extensivo dos seus conceitos no estudo e na prática de métodos algébricos tradicionais, de métodos formais, da semântica axiomática para linguagens imperativas, da semântica denotacional para linguagens declarativas, e em tantas outras situações.

Outra área, que merece atenção pelo amplo uso que faz de assuntos teóricos fundamentais, envolve as *Linguagens de Programação* [4, 6, 7, 12–16] e os correspondentes *Paradigmas* [17, 18]: imperativo, funcional, lógico, orientado a objetos, paralelo, concorrente, etc., cada qual extensivamente explora um ou mais temas associados, da Teoria da Computação.

Na área da modelagem de dados e na teoria dos *Bancos de Dados*, identificam-se muitos conceitos que se fundamentam em conceitos matemáticos básicos oriundos de temas tais como a teoria dos conjuntos, as relações e funções, a álgebra e o modelo relacional.

Muitas outras áreas interessantes poderiam ser aqui listadas cujo lastro teórico e fundamental está na Teoria da Computação, mas as que foram anteriormente apresentadas constituem uma amostra significativa do alcance exibido por essa importante teoria fundamental da Ciência da Computação, e do papel desempenhado por ela no desenvolvimento e no aprimoramento técnico das aplicações computacionais.

3 Conveniência do Estudo de Teoria

Em seu prefácio ao livro de F. S. Beckman [19], já em 1980 o corpo editorial da série escreve que o campo do software básico surgiu a partir da soma dos esforços esparsos de inúmeros profissionais de diversas especialidades, pressionados a produzirem artesanalmente programas de sistema que fossem práticos e funcionais.

Adiante, em seu próprio prefácio, esse autor reconhece a vastidão dos temas que compõem o assunto do livro,e afirma que em suas 443 páginas não se aprofunda suficientemente nos aspectos formais da matéria, e que por essa razão, o seu livro não seria recomendável para estudantes sérios de matemática em cursos de graduação ou pós-graduação devotados ao tema.

É razão para uma reflexão preliminar um confronto entre a opinião emitida pelo autor acerca de seu próprio texto e o notório excelente nível apresentado pela publicação em pauta, especialmente se a compararmos com tantos textos hoje adotados em cursos superiores da área.

Neste ponto, cabe tecer mais algumas considerações sobre alguns dos óbvios porquês do estudo dos assuntos teóricos da Computação, os quais lamentavelmente até mesmo nos dias de hoje, apesar das evidências, continuam sendo considerados por tantos – seguramente por desconhecimento do assunto – como desnecessários, difíceis, desagradáveis e até mesmo inúteis.

Uma exposição desses profissionais, sem preconceitos, às inúmeras belas e profundas questões conceituais que permeiam essa área do conhecimento torna-se a estratégia mais eficaz para contornar os tantos malefícios advindos desse fenômeno.

Reynolds [20], em seu livro, lembra que, em busca de atividades remuneradas permanentes que os satisfaçam, os profissionais costumam, principalmente no início de suas carreiras, mudar de emprego com uma certa frequência, e isso reforça a necessidade que têm de uma base conceitual sólida e abrangente, especialmente na área da Computação.

Segundo Greenlaw [21], o conhecimento da teoria auxilia o profissional até mesmo em aspectos de natureza humanística, como é, por exemplo, o caso da utilização das complexas técnicas da criptografia para garantir privacidade no computador, e o domínio desse conhecimento influi decisivamente na competência com que o profissional da computação exerce sua ocupação.

Referindo-se aos projetos de software da sua época, Wulf, Shaw e Hilfinger afirmavam [22], já em 1981, que as deficiências principais observadas nos programas de então não eram decorrentes de limitações físicas das máquinas, e sim, consequência frequente do despreparo, por parte dos profissionais, na compreensão, em profundidade, da natureza dos seus projetos.

Reynolds [20] alerta também que de um profissional da área se espera não apenas competência nas especificidades do assunto de sua especialidade, mas que domine muito bem cada um dos temas a ela relacionados, em especial os princípios científicos subjacentes de maior profundidade.

Enfim, convém ainda notar que um conhecimento fluente da teoria aguça o juízo estético do profissional, dando-lhe habilidade para priorizar em seus projetos os aspectos mais naturais, espontâneos e elegantes dos processos computacionais, apurando ainda seu senso estético, favorecendo assim a obtenção de produtos com maior beleza estrutural.

No mundo de hoje, que tende a priorizar um contínuo e imediato processo de captura e memorização das novidades de uma tecnologia em rápida evolução, a cultura assim adquirida mostra-se efêmera e insuficiente, motivando que seja resgatada a habilidade técnica do profissional através do incentivo ao uso freqüente e sistemático do raciocínio abstrato para a recuperação e expansão plena dos seus processos mentais.

Num cenário como esse, torna-se indispensável o exercício da habilidade de raciocinar, e não apenas de memorizar, e neste papel é insubstituível o estudo da teoria, dele resultando uma preciosa desenvoltura na formulação precisa e clara de idéias, na capacitação ao julgamento do alcance e das limitações dos aparatos tecnológicos em uso, e na habilitação para uma boa avaliação técnica de propostas de soluções algorítmicas para os problemas.

3.1 Temas-chave da Teoria da Computação

Permeando todos os estudos relacionados com os aspectos teóricos da Computação, encontram-se os assuntos usualmente conhecidos como *Matemática Discreta* [23,24] e *Lógica Matemática* [25,26].

Truss [27] afirma que, independentemente de eventuais aplicações, somente por sua complexidade e beleza a matemática já é em si própria fascinante e significativa, merecendo atenção e estudo.

Diz também ser um verdadeiro prêmio o fato de a matemática ser também útil, embora isso decorra da compreensão profunda da lógica e elegância subjacentes.

Completa sugerindo que não se veja a matemática como uma coleção complicada e incompreensível de abstrações, mas como um interessante e prazeroso tema de estudo, vivo e poderoso.

Entre os tópicos mais comumente estudados da Teoria da Computação, merecem destaque particular os seguintes:

- matemática discreta, teoria dos grafos, teoria dos conjuntos e teoria das funções recursivas;
- linguagens formais e autômatos;
- máquinas universais, computabilidade, algoritmos;
- complexidade, intratabilidade;
- máquinas finitas sequenciais com entrada e saída, transdutores, máquinas de Mealy e de Moore;
- lógica matemática, gramáticas formais, modelos matemáticos, teoremas;
- aspectos teóricos subjacentes a redes neurais, computação evolutiva e sistemas nebulosos.

Além de darem uma visão que permite avaliar o alcance e as limitações dos computadores, uma observação menos superficial permite assegurar que, longe do que seria intuitivo, as complexas estruturas conceituais estudadas na Teoria da Computação gozam, na prática, de uma extraordinária aplicabilidade.

Assim, as ferramentas conceituais oferecidas pela teoria viabilizam a obtenção de soluções para muitos complexos problemas da prática, e apontam formas elegantes e econômicas para a realização tecnológica de tais soluções.

Destacam-se, entre muitos outros, os seguintes casos ilustrativos:

- a álgebra booleana e a teoria das máquinas sequenciais fornecem um substrato conceitual essencial para a descrição de muitos fenômenos computacionais, sendo em particular extensivamente utilizadas no projeto lógico de computadores e de sistemas digitais em geral, de codificadores, de sistemas de comunicação, de controladores e de robôs;
- a teoria das relações e a álgebra relacional constituem fundamentos teóricos nos quais se baseiam muitas técnicas e métodos hoje disponíveis, que nas aplicações práticas proporcionam formas metódicas e muito adequadas para a formulação rigorosa e confiável de sistemas de software que operam sobre complexos bancos de dados;
- autômatos, transdutores e outras máquinas de estados clássicas, formuladas com base em estados internos, funções de transição e funções de saída, têm auxiliado significativamente na elaboração de pré-processadores, de compiladores, e no processamento simbólico de cadeias de símbolos. Essas abstrações constituem o alicerce de inúmeros softwares, de interfaces entre o ser humano e os computadores, em particular os de acesso às redes de computadores, de processos de automação industrial e de protocolos de comunicação digital;
- expressões regulares, que são formas gramaticais muito populares de descrição de linguagens regulares, têm sido modernamente adotadas como notação muito conveniente e prática para a especificação de leis simples de formação de cadeias simbólicas, e, cada vez mais, utilizadas no dia-a-dia de inúmeras atividades profissionais, estando disponível como recurso nativo de muitas linguagens de programação, aplicando-se principalmente em análise léxica, na categorização de cadeias, no alinhamento de textos e no reconhecimento e identificação de padrões sequenciais simples;
- gramáticas gerativas, originalmente estudadas e propostas para aplicações em lingüística, representam formas construtivas para a especificação de linguagens em geral, encontrando freqüentes aplicações, não apenas na sua área inicial de interesse, como também na análise de lingua-gens complexas, e na especificação formal de notações para a expressão de programas nãoconvencionais, tais como ocorre no projeto de linguagens para programas aplicativos especiais;
- a teoria da complexidade computacional [28] e de algoritmos, que se ocupa do estudo de problemas que permitem a obtenção de soluções algorítmicas, ou seja, representáveis através de programas de computador. Estudos nesses domínios têm propiciado o surgimento de métodos e técnicas destinados a identificar a viabilidade de construção de algoritmos cujo tempo de resposta não seja intrinsecamente abusivo, e que não apresentem exageradas exigências de memória de trabalho. Em outra vertente, procuram garantir que algoritmos de criptografia se beneficiem exatamente da dificuldade computacional apresentada por alguns problemas especiais, os quais têm constituído o fundamento de muitos dos protocolos criptográficos modernos;
- a adaptatividade [5] é outro aspecto que merece especial consideração, pelo poder computacional que pode proporcionar, e pela compatibilidade que apresenta com os formalismos clássicos estudados na Teoria da Computação. A ideia principal em que se baseia é a possibilidade que tem um dispositivo conceitual adaptativo de automodificar dinamicamente seu comportamento. Essa propriedade lhe confere o mesmo poder computacional da Máquina de Turing. Podem ser identificadas diversas manifestações históricas deste e de outros conceitos afins. Entre as principais, destacam-se: (i) diretamente na máquina – muitos programas antigos, originalmente produzidos em linguagem de baixo nível, e até mesmo em linguagem de máquina, cujo código se automodi-

ficava para melhor aproveitar a memória do computador, ou então para realizar operações não disponíveis nos comandos básicos da máquina; (ii) nos sistemas de programação – programas cujo código era organizado como um conjunto de overlays, que se revezam dinamicamente na ocupação de uma mesma área de memória, viabilizando assim a execução de programas que de outra forma não caberiam na memória física do computador; (iii) nas linguagens de programação – as linguagens extensíveis, que permitem ao programador definir novos comandos na linguagem, para depois utilizá-los na codificação de seus programas; (iv) nas metalinguagens – as gramáticas de dois níveis, que permitem criar, em duas etapas, gramáticas específicas das sentenças particulares de linguagens dependente de contexto.

- as teorias dos conjuntos difusos, da computação evolutiva e das redes neurais constituem outras contribuições recentes aos fundamentos da inteligência artificial moderna, que nos últimos anos tem se disseminado e evoluído de forma extraordinária.

Os temas mais importantes cobertos pela Teoria da Computação, discutidos nos tópicos estudados a seguir neste artigo, não esgotam a área, mas fornecem um panorama bastante significativo dos assuntos estudados, proporcionando uma boa visão de sua importância prática.

3.1.1 Linguagens Formais e Autômatos

Este é um assunto que vem sendo estudado desde a década de 1950, e tem destacada importância na Teoria da Computação, fundamentando muitos outros temas da área e oferecendo inúmeros subsídios para aplicações práticas. Ubíquo em currículos acadêmicos da área, por sua tão grande importância conceitual e prática costuma-se recomendar que seja apresentado o mais precocemente possível aos estudantes das áreas da computação e afins [29–35]. Historicamente, constata-se que seu estudo foi inicialmente direcionado para questões linguísticas associadas à representação e tratamento de lingua-gem natural, mas acabou conquistando a expressão que hoje apresenta ao se notar sua poderosa praticidade em aplicações voltadas para a compilação ou interpretação de linguagens de programação em computadores digitais.

Essa disciplina concentra sua atenção nas linguagens, apreciadas especialmente do ponto de vista de especificação e de reconhecimento, e também no estudo de modelos constituídos por abstrações gerativas e cognitivas a elas associadas, consideradas tais linguagens segundo sua estrutura, propriedades, características, classificações e inter-relacionamento.

Proporciona um variado e poderoso ferramental conceitual de extraordinária aplicabilidade a outras disciplinas, habilitando o profissional a desenvolver argumentações matemáticas, formais e rigorosas, e proporcionando uma grande familiaridade com os fundamentos e princípios da Ciência da Computação.

Muitas aplicações deste ramo do conhecimento podem ser identificadas. Dentre as mais clássicas, destacam-se: o processamento de linguagens de programação textuais, a representação de processos, estruturas e protocolos de comunicação, do fluxo da lógica dos programas, e de máquinas cujo funcionamento depende de sucessivas alterações do seu estado interno.

Arto Salomaa [36], cita Alfred Aho, que em uma palestra teria feito referência à teoria de linguagens formais como sendo a *flor da ciência da computação*, cujas pétalas seriam a complexidade, as linguagens de programação, os sistemas de programação e os compiladores.

Atualmente, inúmeras aplicações modernas têm explorado densamente as ferramentas conceituais estudadas nesta disciplina. Entre elas destacam-se a computação gráfica, em particular as animações computadorizadas, os hipertextos e as hipermídias, as interfaces Web, e as linguagens visuais e não-lineares em geral.

A importância do estudo desse assunto se manifesta de várias maneiras:

- proporciona uma visão panorâmica das bases científicas da computação;
- proporciona fundamentos teóricos para a área, através do estudo de assuntos tais como decidibilidade, computabilidade e complexidade computacional;
- fornece um sólido lastro para o desenvolvimento de inúmeras aplicações computacionais, entre as quais o processamento de linguagens, o reconhecimento de padrões e a modelagem de sistemas;
- estabelece um forte elo entre a teoria e a prática computacional;
- permite que os conceitos e os resultados da teoria referente às linguagens, seus geradores, reconhecedores e analisadores possam ser aplicados de forma rígida;
- é um dos temas que se mostra mais eclético e menos divorciado dos demais assuntos estudados na Teoria da Computação.

Segundo uma das grandes autoridades mundiais no assunto, Arto Salomaa [36], é curioso observar que o tema das linguagens formais e autômatos, enquanto para tantos representa o principal tópico da pesquisa na Ciência da Computação, para outros constitui apenas algo de importância muito duvidosa. O autor jocosamente insinua ser isso apenas reflexo da época e da cultura dos pareceristas.

Comentando sobre a receptividade dessa disciplina pelos seus alunos, Peter Lintz menciona em sua obra [37] que os estudantes por vezes a rotulam como superfluamente abstrata e inútil na prática. Sugere que, em resposta a essa tendência, o professor oriente seu ensino através de exercícios, e que continuadamente conscientize os alunos dos interessantes desafios que tais disciplinas vão impondo à sua persistência e inventividade na manipulação de problemas de difícil solução.

Hopcroft e Ullman afirmam em seu livro clássico da área [38] que esta disciplina compõe uma importantíssima sub-área da Ciência da Computação.

Lembram que Noam Chomsky criou em 1956 um modelo matemático na forma de gramática, com o qual se propôs a estudar a sintaxe de línguas naturais, mas alguns anos mais tarde seu modelo revelouse expressivo em outra área, ganhando impulso por adequar-se perfeitamente à formalização da lingua-gem de programação ALGOL 60.

O estudo teórico da relação, hoje inseparável, entre gramáticas e os correspondentes autômatos acabou levando à criação da importantíssima ideia dos compiladores dirigidos por sintaxe, e também do conceito de compilador de compiladores, ou seja, dos geradores automáticos de compiladores, ambos até hoje muito utilizados [39].

É digno de uma cuidadosa e responsável atenção o alerta, proclamado por Hopcroft e Ullman [38],lembrando que é virtualmente impossível fazer qualquer estudo sério na área da Computação no qual não seja priorizada a fluência do estudante nas técnicas e nos resultados emanados da teoria das linguagens formais e dos autômatos.

3.1.2 Linguagens de Programação

Outro assunto, que se revela de imensa importância teórica e prática na vida do profissional de computação, refere-se às linguagens de programação. Este artigo procura não se reportar às lingua-gens específicas propriamente ditas, atendo-se, por questões de escopo, somente ao aspecto teórico e conceitual do estudo de linguagens de programação em geral.

Os temas teóricos mais relevantes, associados ao estudo das linguagens de programação, incluem:

- O *Cálculo Lambda* e a *Teoria dos Combinadores*, teorias matemáticas que fundamentam a programação funcional;
- A *Lógica de Floyd-Hoare*, sistema formal usado para provar correção de programas imperativos. É preciso lembrar que os métodos formais têm adquirido importância crescente, permeando estudos sérios em *Engenharia de Software*, e fundamentando muitas de suas práticas contemporâneas;
- A relativamente recente *Teoria de Objetos*, estabelecida por Abadi e Cardelli, que fundamenta toda a área da *programação orientada a objetos*, atualmente muito disseminada e em expansão;
- As *Semânticas Formais* clássicas, voltadas à especificação formal da interpretação das lingua-gens de programação: semântica operacional, semântica denotacional e semântica axiomática.

O estudo teórico das linguagens de programação vem recebendo contribuições de diversas origens, e em muitas direções, compreendendo assuntos fundamentais, relativos à composição, aos domínios, aos escopos, às ligações, às transições, às regras de inferência associados aos programas que com elas podem ser desenvolvidos.

Razões não faltam para que seja reconhecida a necessidade de um conhecimento profundo da teoria das linguagens de programação por parte de qualquer profissional que atue na área de computação:

- A radicalização da aplicação do conceito de reaproveitamento de código tende, progressiva e desnecessariamente, a levar os programadores a se afastarem das tarefas mais criativas e originais da profissão, reduzindo aos poucos a nobre atividade de programação a um mero processo, cada vez mais mecânico e menos intelectual, de seleção e reorganização de componentes préfabricados.
- As linguagens de programação modernas incorporam inúmeros conceitos não-triviais, como por exemplo, sofisticados sistemas de tipos, e por isso mesmo seu uso competente exige não apenas programadores "com prática", mas profissionais que apresentem um mínimo de intimidade com os conceitos e fundamentos abstratos que lhes são subjacentes, para que possam utilizar essas linguagens em todos os seus recursos.
- A necessidade crescente do desenvolvimento de sistemas de software de grande porte torna a especificação dos programas uma tarefa cada vez mais difícil de realizar por parte de profissionais que não possuam um profundo conhecimento e grande fluência na utilização do indispensável ferramental conceitual exigido em tais projetos.
- Para o desenvolvimento de softwares de segurança, nos quais os requisitos de qualidade se manifestam em níveis muito superiores ao que se pratica usualmente no desenvolvimento de programas convencionais, tornase necessário o uso de métodos rigorosos de verificação e de validação formal, cuja aplicação competente exige do profissional profundos conhecimentos teóricos, e não apenas uma habilidade puramente pragmática.
- Todo sistema de software que oferece a seus usuários uma interface, através da qual sejam feitas as comunicações entre o programa

e seu operador, requer algum tipo de processador de linguagem, para cuja concepção e projeto são necessários profundos conhecimentos e habilidades teóricas.

- Sistemas que se utilizam extensivamente de conceitos e conhecimentos teóricos relativos às linguagens de programação estão presentes em toda parte, destacando-se os sistemas de comunicação, os sistemas de manipulação simbólica em geral, os processadores de texto, os sistemas operacionais e muitos outros.

3.1.3 Ciência da Programação

Em seu prefácio ao livro de David Gries [40], o conceituado prof. Edsger W. Dijkstra manifesta sua preocupação com a resistência mental que surge toda vez que se propõe o uso de técnicas do pensamento científico em uma nova área de conhecimento, e recomenda um cuidado especial ao se procurar convencer neófitos de que, embora à primeira vista isso não seja evidente, aqueles "inúteis" formalismos abstratos não apenas são muito úteis, como também indispensáveis.

Ainda nesse livro, o autor sugere que esse fenômeno assim ocorre devido a uma série de razões:

- Muitas vezes, pessoas que desconhecem o assunto costumam colocar objeções injustificadas a seu uso, em função da dificuldade que sentem em compreender e conviver com notações matemáticas, formalismos, técnicas rigorosas e provas formais.
- Alegando serem esses formalismos desnecessários no dia-a-dia, argumentam que, em vista do demorado e oneroso preparo prévio exigido de um profissional para que esteja apto a usá-los fluentemente, torna-se lícito e prático aceitar como satisfatória a qualidade de programas pragmaticamente resultantes de desenvolvimentos artesanais, efetuados sem qualquer ajuda de métodos rigorosos.
- Outro argumento, infundado e recorrente, alega ser difícil e até inviável o emprego de técnicas rigorosas para desenvolver e implementar programas de grande porte, cuja ocorrência se mostra cada vez mais freqüente na prática.
- Por incrível que pareça, alguns profissionais da área alegam ser muito mais importante que se disponha de uma especificação bastante precisa para um software a ser desenvolvido do que o correspondente programa estar, ele próprio, correto.
- No extremo, outros alegam ainda que o mundo real dispensa provas formais, e que, para todos os efeitos, na prática é suficiente apenas que o programa desenvolvido cumpra sua tarefa.

Aludindo a essas posições, professadas por tantos profissionais, cita ainda o autor um trecho do Dicionário Oxford, que contrasta os conceitos de *ciência* e de *arte*: aquela, fundamentada em princípios e na constante utilização dos mesmos; esta, sustentada pelo conhecimento de tradições e pelas destrezas que se desenvolvem através da prática.

Gries ainda afirma que a programação só poderá se converter em uma ciência através do cultivo de uma profunda maturidade matemática, acompanhado de uma intenção real de empregar esse tipo de recurso na prática do dia-a-dia.

3.1.4 Computabilidade

De forma análoga ao que se pode observar em outras especialidades científicas, uma grande variedade de importantes descobertas desta fascinante área do conhecimento foi obtida teoricamente, muito antes de sua prática tornar-se tecnologicamente viável.

Infelizmente, tal fato induziu, e ainda hoje induz muitos, à dissociação entre os fatos ligados ao estudo e ao uso dos computadores e os ensinamentos proporcionados pela Teoria da Computabilidade.

Kfoury, Moll e Arbib [41] afirmam, no entanto, em seu livro que, por sua importância, a Teoria da Computabilidade pode e deve ser considerada como sendo, apesar de tudo, o coração da Ciência da Computação.

Nessa linha, tem-se procurado determinar e desenvolver um alicerce científico para o estudo de assuntos relacionados com a Informática de modo geral, e em particular,de assuntos ligados ao processamento de dados através de algoritmos, ao projeto de computadores digitais e ao projeto de programas desenvolvidos em linguagens de programação.

Desde meados do século passado, em diversos aspectos se tem observado um crescimento significativo do desenvolvimento da Computação, como por exemplo: a sofisticação dos processos; o barateamento dos componentes eletrônicos e o violento aumento de sua capacidade e complexidade; a viabilização de equipamentos eletrônicos muito poderosos e a custo acessível; os avanços nas metodologias de programação; a drástica redução do tempo necessário para a obtenção de programas complexos e confiáveis; o desenvolvimento de técnicas avançadas de especificação rigorosa de programas, processos e equipamentos.

Nesse contexto, a grande aplicabilidade da Teoria da Computabilidade pode ser facilmente identificada, considerando-se que abrange, entre muitos outros assuntos:

- Aspectos da sintaxe e da semântica de lingua-gens de programação
- Estudo da enumeração e da universalidade das funções computáveis

- Técnicas de computabilidade e estado computável de problemas
- Metodologia de programação e prova de correção de programas
- Semântica denotacional
- Programas recursivos
- Regras de prova para propriedades dos programas
- Auto-referência em computabilidade e teorema da recursão
- Conjuntos, conjuntos recursivos, conjuntos recursivamente enumeráveis, teorema de Gödel
- Máquina de Turing e formulações alternativas da teoria da computabilidade.
- Automatização da prova de teoremas

Monty Newborn, em seu livro sobre provadores automáticos de teoremas [15], comenta que os computadores modernos têm evoluído muito rapidamente, ficando, cada vez mais, velozes, compactos e baratos, e já permitiram a construção de programas capazes de provar teoremas e até de vencer campeões mundiais de xadrez.

3.1.5 Complexidade computacional

Em artigo recentemente publicado [42], o conceituadíssimo prof. Peter Denning, reportando-se a observações da realidade do dia-a-dia das principais aplicações dos computadores, resume, em poucas palavras, a relevância e o impacto que tem o tema Complexidade Computacional sobre as atividades do profissional da área:

> ... over 3,000 common problems in science, engineer-ng, and business are so difficult to solve that even the fastest supercomputers would take centuries on simple versions.
>
> (Peter J. Denning *The profession of IT* Communications of the ACM, v.51, n. 8, aug. 2008, p. 21)

De origem relativamente recente, datada da segunda metade da década de 1970,a Teoria da Complexidade Computacional tem recebido uma extraordinária atenção dos pesquisadores da Ciência da Computação, em vista de sua grande importância para a área [2, 28].

O estudo da Complexidade Computacional exige a compreensão de fenômenos não triviais, resultantes da interação da computação, da lógica e das aplicações, para investigar as razões de alguns problemas apresentarem soluções computacionais tão onerosas.

Enquanto a Teoria da Computabilidade se ocupa de determinar a possibilidade ou não de uma dada classe de problemas poder ser resolvida algoritmicamente, a complexidade computacional se detém nos aspectos da viabilidade prática da execução de algoritmos considerados adequados para a tarefa.

Assim, os problemas computáveis conseguem motivar os estudiosos da Computabilidade não apenas por representarem curiosos desafios a serem resolvidos pelo programador, mas também por constituírem por si próprios interessantes objetos de estudo:

- variados formalismos estão disponíveis para sua modelagem e estudo
- uma vez abstraídos e formalizados, podem ser analisados de maneira independente do particular formalismo com o qual foram representados
- ao estudarem as propriedades dos problemas, os pesquisadores buscam para eles soluções que preferencialmente sejam ao mesmo tempo rápidas e compactas
- para essas propostas de solução, são consideradas particularmente convenientes aquelas que apresentem requisitos polinomiais em suas exigências de área de trabalho, e de tempo de resposta
- as soluções mais adequadas na prática são aquelas que apresentam, no espaço e no tem-po,comportamento polinomial de ordem baixa (de preferência, linear)

Muitos tópicos costumam ser considerados dentro da temática da Complexidade Computacional:

- *Algoritmos*: modelos de computação conhecidos podem ser convertidos algoritmicamente uns nos outros, com uma perda polinomial de eficiência, exceto nos casos não-determinísticos, de resposta exponencial. *Assimetrias* e *não-determinismos* criam atraentes possibilidades para a classificação de problemas quanto à sua complexidade computacional.
- *Máquina de Turing: Computabilidade e Decidibilidade*: de aparência à primeira vista obscura, e dispondo de operações básicas muito primitivas, a Máquina de Turing mostra-se capaz de representar algoritmos sem significativa perda de eficiência, o que a tem consagrado como modelo de computação. Computacionalmente muito poderosa, a Máquina de Turing é capaz até mesmo de computar fatos não-triviais sobre os algoritmos que representam.Todavia, há muitos problemas, ditos incomputáveis, aos quais não existe qualquer possibilidade de se associar um algoritmo, ainda que ineficiente. Máquinas de Turing estritamente representam, portanto, problemas computáveis. Estuda-se, para problemas computáveis, o fenômeno da decidibilidade: alguns dos problemas computáveis, ditos indecidíveis, são representados por Máquinas de

Turing que iniciam um processamento infinito em resposta a certos dados de entrada, enquanto os demais (problemas decidíveis) garantidamente terminam o seu processamento qualquer que sejam os dados a elas fornecidos. Naturalmente, quando possível, os últimos são os preferidos, em termos práticos.

- *Lógica booleana* e *Lógica de Primeira Ordem*: Estas formulações permitem criar sistemas de provas gerais e compreensíveis, por meio de comandos matemáticos detalhados cuja semântica associa cada sentença à respectiva aplicação matemática, permitindo expressar fatos sobre computações, o que propicia o surgimento de problemas lógicos indecidíveis.
- O estabelecimento de *Classes de Complexidade* e sua inter-relação é um assunto complexo e com inúmeros pontos ainda em aberto, e isso tem motivado diversos trabalhos na área, e dado origem a muitos assuntos de investigação, constituindo assim um tema bastante adequado para pesquisa atual em Ciência da Computação.
- *Reduções* e completude: O uso da lógica se revela muito interessante quando, através das técnicas de redução, se torna possível estabelecer pontos em comum acerca da dificuldade apresentada pelos problemas pertencentes a toda uma classe de complexidade. O estudo da *NP-completude* e de temas afins suscita importantes métodos de análise da Complexidade Computacional.
- A dificuldade apresentada por determinados problemas pode demonstrar-se útil e não indesejável como ocorre na maioria dos casos práticos. É o que ocorre no estudo da *Criptografia*, no qual a complexidade dos algoritmos é justamente aquilo que se busca e se necessita ter sob controle. Curiosamente, nessas exatas situações em que a complexidade é mais almejada, é que se mostra mais difícil obter a sua manifestação.
- *Computação paralela* é uma área de interesse na qual se mostra mais necessário e mais árduo o estudo da complexidade computacional, especialmente quando tal estudo envolve simultaneamente numerosos elementos como agentes paralelos.

4 Formação dos profissionais da área

Complementando as diversas alusões já feitas anteriormente neste artigo, cabem algumas observações adicionais sobre a maneira como se formam ou como deveriam ser formados os profissionais que trabalham com Computação. Inicialmente, transcreve-se a seguir um trecho de uma entrevista [43] dada pelo conceituado prof. Denning:

> The four core practices of computing professions are programming, systems, modeling, and innovating.To be a complete computing professional, you should know the principles and be competent in the four practices.
>
> (Entrevista do prof. Peter J. Denning à Ubiquity)

Tem-se observado uma queda cada vez mais pronunciada na qualidade da formação dos futuros profissionais da Computação no que tange à concepção e à implementação de programas, especialmente os de porte menor.

A formação que eles recebem tem, nesses casos, negligenciado os essenciais aspectos microscópicos da problemática dos projetos pequenos, em favor de uma forte polarização para a gestão de grandes projetos de software ou para o desenvolvimento de grandes sistemas.

Como resultado,nota-se,da parte do profissional, uma perda sensível de habilidade de projeto de programas e algoritmos, uma dificuldade de compreensão de diversos assuntos ligados às lingua-gens de programação e à sua utilização como instrumento para o desenvolvimento de programas. Observa-se ainda uma notória deficiência em sua criatividade para a resolução autônoma de problemas, sem recorrerem a repositórios de programas pré-existentes.

Em relação a isso, já em 1981 o renomado David Gries alertava [40] que, no mínimo, os programas grandes podem ser vistos como uma coleção organizada de programas menores, e que ninguém se pode considerar especialista em produzir programas grandes de qualidade enquanto não se provar um perito no desenvolvimento de bons programas pequenos.

Pode-se acrescentar que é temerário considerar capacitado para gerir um dado projeto um profissional não dotado de um pleno domínio sobre todos os aspectos do mesmo, pois dificilmente costumam ser passivamente acatadas as diretrizes emanadas por gestores que não estejam aptos a demonstrarem competência nos assuntos a que se refere o projeto que pretendem conduzir.

Outro assunto de importância para a presente reflexão, refere-se à influência da adoção de métodos baseados em formulações teóricas na qualidade dos produtos gerados pelos profissionais de computação.

Alvo tradicional da pesquisa teórica da computação,a resoluçãode problemas formulados de maneira precisa e rigorosa através de formalismos matemáticos bem estabelecidos mostra-se essencial para assegurar a construção de programas de alta qualidade.

Tais formulações constituem, na prática, um grande desafio, representando um importante auxílio a atividades de projeto, e, por suas propriedades, podem ser exploradas no estabelecimento de métodos de avaliação objetivos.

As métricas assim desenvolvidas se empregam em instrumentos de avaliação que utilizam índices de mérito do software, o que proporciona formas objetivas interessantes não apenas para avaliações individuais como também para comparações entre diferentes softwares e para uso em sistemas de auxílio à tomada de decisões.

As principais metas que se costuma perseguir quando se utilizam tais métodos são: embasamento matemático sólido para os modelos desenvolvidos; fundamentação científica para as decisões tomadas; alta qualidade técnico-científica para os produtos desenvolvidos; anteposição de um planejamento científico à aplicação de práticas tecnológicas; expurgo conseqüente da prática de reapro-veitamento míope de peças pré-fabricadas no desenvolvi-mento de produtos de software de alta qualidade.

5 Conclusão

Para atender às necessidades de uma época em que se faz necessário exigir do profissional uma capacitação cada vez melhor e mais ampla, é preciso um constante esforço de reciclagem e atualização, tanto da parte dos profissionais como das instituições de ensino, para que seja possível bem trabalhar com os atuais sistemas computacionais, de complexidade e calibre sempre crescentes.

É consenso ser fator distintivo dos mais capacitados profissionais da Informática o conhecimento e domínio das teorias, abstrações e conceitos que fundamentam suas atividades técnicas, e essa distinção se evidencia progressivamente à medida que, simultaneamente, crescem as dimensões dos sistemas computacionais contemporâneos.

Na área acadêmica, verifica-se um crescente afastamento de metas entre disciplinas formativas e as de caráter mais pragmático, e isso é outro preocupante motivo de atenção.

Cabe assim um alerta sobre a necessidade de resgate das importantes relações entre elas existentes, manifestas nas afinidades entre os modelos abstratos estudados na teoria e as atividades, métodos e técnicas práticas de programação que neles devem sempre se apoiar.

Flávio S. C. da Silva e Ana C. V. de Melo defendem em seu livro [17] a existência de uma forte afinidade entre cada paradigma de programação e algum dos modelos de computação existentes, sugerindo que para cada paradigma seja utilizado o modelo de computação que lhe for mais expressivo. Assim, por exemplo, seriam usadas máquinas de Turing para o paradigma imperativo, enquanto Funções Recursivas e Cálculo Lambda seriam empregados para o paradigma funcional e outros paradigmas declarativos.

Do ponto de vista pedagógico, para que possam ser adequadamente fundamentadas as diferentes atividades ligadas à programação, aconselham os autores que se dê certa uniformidade à ênfase com que é efetuado o estudo dos diferentes modelos de computação disponíveis.

Com a formação conceitual abrangente, assim proporcionada, os profissionais terão condições de gerar soluções simples e transparentes para seus problemas, e como decorrência natural, de produzir para eles programas eficientes, elegantes e corretos.

É muito atual e oportuno meditar muito seriamente sobre o cenário científico e tecnológico em que trabalha o profissional da computação nos dias de hoje.

De um lado, é possível adquirir conhecimentos para formar consciência da importância que têm no dia-a-dia os temas difíceis e abstratos de que tratam as disciplinas teóricas da área da Computação, e é gratificante a certeza da utilidade de uma sólida formação, resistente à moda, ao tempo e à tecnologia, que paire inabalável acima de tendências políticas e mercadológicas.

De outro,a observação dos fatos do nosso mundo traz muitas incertezas sobre a maneira como serão tratados esses assuntos no futuro.

É angustiante a constatação de que a influência do desconhecimento e da falta de preparo, aliados ao imediatismo, venha de há muito produzindo seus efeitos, especialmente através de decisões tomadas com base em prioridades distorcidas, que acabam criando importantes barreiras a uma boa formação profissional.

Desse quadro, os maus efeitos que ainda não vieram à tona seguramente se manifestarão em futuro próximo, quando, com certeza, uma reforma para a reversão dessa situação deverá mostrar-se muito difícil, demorada e one-rosa, se não inviável.

Referindo-se em particular ao futuro das atividades de programação, em entrevista concedida a Edward Feigenbaum [44], o respeitadíssimo prof. Donald Knuth faz um grave alerta que convida à reflexão, dada sua veracidade e as severas consequências não mencionadas, dela decorrentes:

> I'm worried about the present state of programming. Programmers now are supposed to mostly just use libraries. Programmers aren't allowed to do their own thing from scratch anymore. They're supposed to assemble reusable code that somebody else has written. There's a bunch of things on the menu and you choose from these and put them together. Where's the fun in that? Where's the beauty in

that?

(Commun. of the ACM, v.51, n.8, aug. 2008, p. 35)

Fica então esta pergunta para ser respondida: se as atuais tendências continuarem como as observamos hoje, com que deverá parecer-se a nossa profissão daqui a uns quinze ou vinte anos? Não apenas no que tange à atividade da programação, mas igualmente a todas as outras especialidades da área da Computação...

Referências

[1] D. I. A. Cohen, *Introduction to Computer Theory*. Wiley, 1997. 2nd ed.

[2] H. R. Lewis and C. H. Papadimitriou, *Elementos da Teoria da Computação*. Porto Alegre: Bookman, 2004.

[3] H. Barendregt, *The Lambda Calculus: Its Syntax and Semantics, volume 103 of Studies in logic and the foundations of mathematics*, vol. 155. 1984.

[4] R. Backhouse, *Syntax of Programming Languages*. London: Prentice-Hall International, 1979.

[5] J. J. Neto, "Adaptive rule-driven devices — general formulation and case study," in *CIAA* (B. W. Watson and D. Wood, eds.), vol. 2494 of *Lecture Notes in Computer Science*, pp. 234–250, Springer, 2001.

[6] F. G. Pagan, *Formal Specification of Programming Languages: A Panoramic Primer. Englewood Cliffs, New Jersey 07632*. Prentice-Hall, Inc., 1981.

[7] K. Slonneger and B. Kurtz, *Formal Syntax and Semantics of Programming Languages: A Laboratory Based Approach*. Addison-Wesley, 1995.

[8] C. L. Lucchesi and et al, *Aspectos Teóricos da Computação*. Impa – CNPq, 1979.

[9] J. C. Martin, *Introduction to Languages and the Theory of Computation*. New York: McGraw-Hill, 2003.

[10] M. Sipser, *Introdução à Teoria da Computação*. Editora Thomson Learning, 2007.

[11] T. Scheurer, *Foundations of computing: system development with set theory and logic*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1994.

[12] M. Gordon, *Programming Language Theory and Its Implementation*. Englewood Cliffs: Prentice-Hall, 1988.

[13] A. D. McGettrick, *The Definition of Programming Languages*. Cambridge University Press, 1980.

[14] B. Meyer, *Introduction to the Theory of Programming Languages*. Englewood Cliffs: Prentice Hall, 1990.

[15] M. Newborn, *Automated theorem proving: theory and practice*. Springer Verlag, 2001.

[16] G. Winskel, *The Formal Semantics of Programming Languages*. Cambridge, Massachusetts: MIT Press, 1993.

[17] F. S. C. Silva and A. C. V. Melo, *Modelos Clássicos de Computação*. Thomson, 2006.

[18] R. W. Sebesta, *Concepts of Programming Languages*. Redwood City, Calif.: Wiley, 2002.

[19] F. Beckman, *Mathematical Foundations of Programming*. Boston: Addison-Wesley, 1980.

[20] J. Reynolds, *Theories of programming languages*. Cambridge University Press, 1998.

[21] N. Greenlaw and H. Hoover, *Fundamentals of the Theory of Computation*. San Francisco: Morgan Kauffman, 1998.

[22] W. A. Wulf, M. Shaw, and P. N. Hilfinger, *Fundamental Structures of Computer Science*. Reading, Massachusetts: Addison Wesley Publishing Company, 1981.

[23] E. R. Schneiderman, *Matemática Discreta – uma introdução*. Thomson, 2003.

[24] N. L. Biggs, *Discrete Mathematics*. Oxford Science Publications, Clarendon Press, revised ed., 1989.

[25] A. G. Hamilton, *Logic for mathematicians*. 1990. revised edition.

[26] E. Mendelson, "Introduction to mathematical logic," *Van Nostrand-Rein-hold Co., New York*, 1964.

[27] J. K. Truss, *Discrete mathematics for computer scientists*. Addison-Wesley, 1994.

[28] C. Papadimitriou and C. Papadimitriou, *Computational complexity*. Addison-Wesley Reading, MA, 1994.

[29] J. Carroll and D. Long, *Theory of Finite Automata with an Introduction to Formal Languages*. Prentice Hall, 1989.

[30] P. G. et al., *Teoría de autómatas y lenguajes formales*. México: Alpha Ômega, 2001.

[31] M. Harrison, *Introduction to formal language theory*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1978.

[32] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*. Reading, MA: Addison Wesley, 1979.

[33] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 2001.

[34] R. Muñoz and L. Villodre, *Lenguajes, gramáticas y autómatas: Curso básico*. México: Alpha Ômega, 2002.

[35] G. Revesz, "Introduction to Formal Language Theory," 1983.

[36] A. Salomaa, *Jewels of formal language theory*. Computer Science Press, Incorporated, 1981.

[37] P. Linz, *An Introduction to Formal Languages and Automata*. Boston: Jones and Bartlett Publishers, 2006.

[38] J. E. Hopcroft and J. D. Ullman, *Formal languages and their relation to automata*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1969.

[39] A. Aho, *The Theory of Parsing, Translation, and Compiling*. Englewood Cliffs: Prentice-Hall, 1972.

[40] D. Gries, *The Science of Programming*. Springer-Verlag, 1981.

[41] A. Kfoury, R. Moll, and M. Arbib, *A Programming Approach to Computability*. Springer-Verlag, 1982.

[42] P. J. Denning, "The profession of IT," *Communications of the ACM*, vol. 51, Aug. 2008.

[43] J. Gehl, "Ubiquity Interview: Great Principles of Computing. Ubiquity Editor John Gehl interviews Peter J. Denning about the progress of the Great Principles Project," *Ubiquity – An ACM IT Magazine and Forum*, vol. 7, November 2006. `http://www.acm.org/ubiquity/interviews/v8i22_denning.html` – last visited 30^{th} December 2007.

[44] E. Feigenbaum, "Interview with Donald Knuth: A life's work interrupted," *Communications of the ACM*, vol. 51, pp. 31–35, aug 2008.

Mini-currículo do autor: João José Neto é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975), doutor em Engenharia Elétrica (1980) e livre-docente (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo e coordena o LTA — Laboratório de Linguagens e Tecnologia Adaptativa do PCS — Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Ciência da Computação, com ênfase nos Fundamentos da Engenharia da Computação, atuando principalmente nos seguintes temas: dispositivos adaptativos, tecnologia adaptativa, autômatos adaptativos,e em suas aplicações à Engenharia de Computação,particularmente em sistemas de tomada de decisão adaptativa, análise e processamento de linguagens naturais, construção de compiladores, robótica, ensino assistido por computador, modelagem de sistemas inteligentes, processos de aprendizagem automática e inferências baseadas em tecnologia adaptativa.