CONSTRUÇÃO DE SIMULADORES GRÁFICOS PARA TEORIA DA COMPUTAÇÃO: UMA PROPOSTA PARA O ENSINO DO CONCEITO DE MÁQUINAS DE TURING

Gerson Pastre de Oliveira[1]
Marco Aurélio Seraphim da Silva[2]

Resumo

Este trabalho relata a construção de um sistema que utiliza uma interface gráfica para simular o conceito de máquina de Turing. A interface, construída em Java, é baseada em técnicas de programação de jogos e tem como finalidade principal auxiliar no processo de ensino-aprendizagem em disciplinas como Teoria da Computação. Além dos aspectos técnicos pertinentes ao funcionamento da aplicação, este artigo procura abordar as vantagens do uso de simulações no ensino superior, principalmente em processos que envolvam conceitos complexos.

1. INTRODUÇÃO

Por que usar simuladores para o ensino de teoria da computação?

A pergunta que dá título a esta seção suscita uma série de reflexões, todas elas bastante relevantes, envolvendo duas áreas do conhecimento, as quais têm freqüentemente aparecido interconectadas na atualidade: educação e ciência da computação. Na tentativa de colaborar com a construção de conhecimentos que permitam trabalhar com estas duas frentes, surge este artigo, também como resultado de um esforço de investigação, realizado no ano de 2006, no âmbito do Programa de Iniciação Científica da Faculdade Politécnica de Jundiaí.

O caráter complexo do conhecimento pertinente já foi assinalado por Morin (2002), ao indicar que a linearidade e a simplicidade não se coadunam com a multidimensionalidade que constitui a construção dos saberes contemporâneos. Em meio ao caos gerado pela diversidade das fontes de dados, da multiplicação exponencial dos meios de acesso e das formas de conexão com as informações e do fluxo incontrolável de produção destes elementos, dados e informações, não há como pretender que o conhecimento, uma construção pessoal refinada, se submeta a parcelização em disciplinas não comunicantes. Ao contrário, diante dos desafios da complexidade, a interdisciplinaridade parece colaborar de forma definitiva para que as iniciativas que visem

[1] Professor da Faculdade Politécnica de Jundiaí (Anhanguera Educacional); Doutor em Educação (USP)
[2] Faculdade Politécnica de Jundiaí.

subsidiar este processo de construção de conhecimentos tenham êxito – principalmente no meio acadêmico.

Além disso, múltiplos meios de assimilação de dados estão disponíveis na atualidade, permitindo que as sínteses pessoais que resultam em informações possam ser feitas para além das dimensões primárias da oralidade e do texto escrito, ainda que estas sejam formas muito importantes e válidas neste processo. Entre esses outros, em adição aos tradicionais meios de comunicação de massa, estão as novas tecnologias de informação e comunicação (NTIC). As possibilidades das NTIC são amplas, incluindo comunicação e ensino a distância através de diversos paradigmas. Entretanto, tais tecnologias abrem novas possibilidades também no ensino presencial, de forma a permitir que a estratégia pedagógica do professor receba o auxílio de programas computacionais que pretendam dar suporte ao processo de aprendizagem, por parte dos alunos, de conteúdos complexos. A perspectiva do computador em sala de aula não é a da substituição de professores por máquinas.

> *As ferramentas computacionais, utilizadas como auxiliares do processo de ensino-aprendizagem – portanto devidamente encaixadas na estratégia pedagógica do curso – rendem largas oportunidades para a construção crítica do conhecimento. Não realizam o papel do professor, não ensinam, não resolvem todos os problemas das diversas dimensões da escola, mas podem oportunizar, no contexto acanhado da sala de aula e para além dele, a dinâmica da experimentação. (...) Para o aluno, surge a oportunidade de intervir, de usar da tecnologia para tornar-se co-autor* (Oliveira, 2004).

Para o professor, então, o computador configura-se como um auxiliar valioso, desde que utilizado com o planejamento devido. Mas os discentes, para os quais existem conteúdos e estratégias, não estão esquecidos.

> *Diante da possibilidade de ampliar seu papel, o aluno encontra conexões e implementa, efetivamente, uma interdisciplinaridade que lhe traz sentido e que é mais do que um simples projeto de interligação de conteúdos de disciplinas diversas. Além disso, amplia-se o espaço interacional, composto por pessoas que aprendem juntas e pelo próprio conhecimento – não apenas informação* (*idem*,*ibidem*).

Em uma sociedade na qual a interatividade está cada vez mais presente e na qual os dispositivos que potencializam o uso dos sentidos constituem de forma crescente a vida das pessoas, inclusive para aprender, propor que os computadores sejam usados para ensinar uma teoria complexa parece muito razoável. Uma possibilidade aberta para o uso dos computadores com fins educacionais é a da simulação, definida por Oliveira como

> *(...) a utilização de alguns aspectos de determinado universo, testados através de inserções feitas em um modelo, admissível neste mesmo universo. Trata-se de método utilizado na tecnologia educacional, geralmente chamado de simulação baseada em computador, e que, na definição de Laurillard (1995), é um programa que assume algumas instâncias de um aspecto do mundo, permitindo que o usuário faça entradas no modelo, execute-o e mostre os resultados* (Oliveira, 2000, p.44-45).

Como pode ser visto mais adiante neste trabalho, o conceito de máquinas universais, e em especial o de máquinas de Turing, no âmbito do ensino de disciplinas como Teoria da Computação, é bastante complexo, principalmente no que diz respeito à operacionalização de máquinas específicas, dadas a partir de expressões regulares ou de funções. A construção de um programa que efetue a simulação deste processo pode auxiliar no teste de hipóteses formuladas por professores e alunos, de modo a consolidar o aprendizado através da experimentação. O caráter interdisciplinar da proposta surge ao lançar uma visão mais ampla em relação ao uso de simuladores deste tipo: o caráter de subsídio à aprendizagem lógica tem interfaces importantes e integradoras com disciplinas como algoritmos, estrutura de dados, matemática discreta, entre outras.

2. MÁQUINA DE ESTADO FINITO E MÁQUINA DE TURING

2.1 Máquina de estado finito

A definição formal de máquinas de estado finito pode ser dada de acordo com a proposição seguinte (Gersting, 2001, p.399):

> *$M = [S, I, O, f_e, f_o]$ é uma máquina de estado finito se S é um conjunto finito de estados, I é um conjunto finito de símbolos de entrada (o alfabeto de entrada), O é um conjunto finito de símbolos de saída (o alfabeto de saída) e f_e e f_o são funções, onde f_s:S x I $\rightarrow$ s e f_o:S $\rightarrow$ O. A máquina sempre começa inicializada em um estado inicial fixo s_0.*

Ao falar sobre este conceito, Gersting (*op.cit.*, p.398) indica que as mesmas constituem um modelo capaz de assimilar características de computadores atuais. Em termos gerais, o que se pode entender sobre tais máquinas é que as mesmas possuem, assim como os computadores digitais, *operações sincronizadas* por pulsos discretos, têm *ações previsíveis*, ou seja, funciona de forma determinística, respondem a *dados de entrada* e produzem *dados de saída*, mediante processos aplicados aos dados de entrada e de acordo como a seqüência de *estados*, de ordem finita, que a máquina pode assumir.

O conceito de estado é extremamente importante. Para Gersting (*idem*, *ibidem*), "em qualquer momento, a máquina está em exatamente um desses estados. Em qual estado ela estará a seguir depende tanto do estado atual quanto dos dados de entrada". Ou seja, trata-se

de uma máquina conceitual, a qual, a cada ciclo de funcionamento, recebe um dado de entrada, que, por sua vez, alterará o estado no qual a máquina se encontra, de forma a emitir uma saída compatível.

Como estado, ainda, pode-se entender uma memória que armazena um número representativo do ponto do processamento no qual a máquina se encontra. O número de estados deve ser finito, e a especificação de cada um deles conhecida no âmbito da máquina em questão.

Os dados de entrada da máquina são conhecidos e dados através de um alfabeto *I*, que deve conter um conjunto finito de símbolos, entre os quais pode constar o símbolo e o sentido matemático de vazio (∅).

Em Gersting (2001, p.399), tem-se que a descrição particular de uma máquina de estado finito compreende a definição dos conjuntos e funções pertinentes. As funções f_s e f_o podem ser definidas com o uso de tabelas de estados ou de grafos de estados, conforme exposto nas figuras seguintes. Para as máquinas ali representadas, os conjuntos envolvidos seriam $S = \{s_0, s_1, s_2\}$, $I = \{0,1\}$, $O = \{0,1\}$.

Estado atual	**Próximo estado** **Entrada Atual** **0**	**1**	**Saída**
S_0	S_1	S_0	**0**
S_1	S_2	S_1	**1**
S_2	S_2	S_0	**1**

Tabela 1. Tabela de estado (adaptado de Gersting, 2001, p.399)

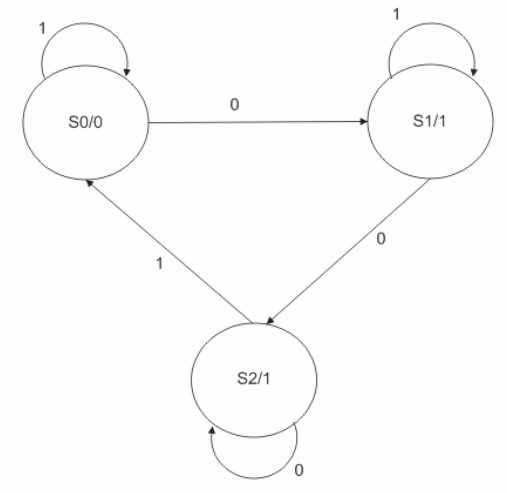

Figura 1. Grafo de estado (adaptado de Gersting, 2001, p.399)

2.2 Máquina de Turing

Em 1936, o matemático britânico Alan M. Turing fez uma proposta que se mostrou, posteriormente, muito avançada em termos científicos para a época: o dispositivo conceitual que, em sua homenagem, ficou conhecido como *máquina de Turing*. Trata-se de uma máquina de estado finito que agrega possibilidades adicionais, como a de reler os dados de entrada, além de escrever e apagar por cima dos mesmos (Gersting, 2001, p.419). Este modelo representa um avanço em relação às máquinas de estado finito tradicionais, de modo a superar as limitações encontradas em tais modelos, principalmente em uma característica que amplia de forma exponencial suas perspectivas: a máquina de Turing possui memória auxiliar ilimitada (*idem*, *ibidem*).

Uma máquina de Turing contém um conjunto de quíntuplas que definem seu comportamento e uma fita, dividida em células, cada qual contendo apenas um símbolo válido de acordo com um alfabeto finito disponível. Esta fita tem seu tamanho ilimitado, e pode ser lida e escrita por uma cabeça de leitura móvel (conceitual), a qual, a cada ciclo, está parada sobre uma das células da fita. Um símbolo especial, geralmente *b*, permite identificar as células em branco. Além disso, a quantidade de células que não está em branco em determinado momento é sempre finita (*idem, ibidem*). A direção do movimento é dada por *D* (direita) ou *E* (esquerda).

Como definição formal, tem-se:

Sejam S um conjunto finito de estados e I um conjunto finito de símbolos para a fita (o alfabeto da fita), incluindo um símbolo especial b. Uma ***máquina de Turing*** *é um conjunto de quíntuplas da forma (s,i,i´,s´,d), onde s, s´ ∈ S; i,i´ ∈ I; e d ∈ {D,E}, tais que duas quíntuplas distintas nunca começam com os mesmos símbolos s e i (Gersting, 2001, p.420).*

s	***i***	***i'***	***s'***	***d***
estado atual	*símbolo atual*	*símbolo impresso*	*próximo estado*	*direção do movimento*

Figura 2. Quíntupla para máquina de Turing

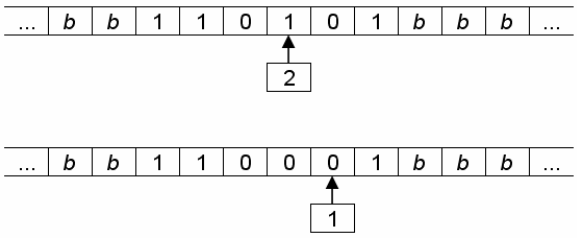

Figura 3. Exemplo de fita para Máquina de Turing e comportamento da mesma para a quíntupla (2,1,0,1,D)

3. PROTÓTIPO

A opção metodológica adotada neste trabalho previu, desde o início, a construção de um protótipo em linguagem computacional que permitisse simular os conceitos relacionados à máquina de Turing. Para a criação do mesmo, foi escolhida a linguagem de programação Java, dada sua independência de plataformas, e por permitir que um mesmo aplicativo possa ser executado tanto em ambiente *desktop* quanto na Internet, na forma de *applet*.

Como forma de manter a portabilidade, todo o projeto foi criado como um componente, no qual existe uma classe (*TuringMachine*) que encapsula todas as funcionalidades. Todas as interações entre usuário e máquina são realizadas através desta classe. Desta forma, para portar a aplicação de um ambiente *desktop*, para um ambiente *web*, é necessário apenas criar uma classe do tipo *JApplet* e instanciar a classe *TuringMachine*, inicializando a mesma de forma adequada.

Com o propósito de simulação para o aprendizado, a tela principal do protótipo foi criada para ser intuitiva e de fácil utilização, podendo ser dividida em 5 partes básicas: *menu de opções*, *estado atual*, *quíntuplas*, *fita* e *próximo passo*. O *menu*, situado na parte

superior da tela, procura proporcionar facilidade no acesso às funcionalidades do protótipo. Na área de *estado atual*, situada na parte superior esquerda da tela, é possível visualizar o estado atual da máquina. As *quíntuplas cadastradas* são apresentadas na parte direita da tela, em forma de lista, permitindo que uma grande quantidade das mesmas seja exibida. A *fita*, situada na parte inferior da tela, possui um quadrado vermelho, com o texto em negrito no centro, simbolizando a cabeça de leitura. Já o símbolo de cor vermelha, que aparece ao lado da cabeça de leitura (à esquerda ou à direita, dependendo do caso), simboliza o último símbolo alterado. Na funcionalidade de iniciar a próxima transição, existe o botão *próximo passo*, situado na parte inferior central da tela.

Implementado na forma de depurador, o protótipo fica aguardando a instrução por parte de seu operador para que seja executado o próximo passo das transições da máquina, o que é feito através de um botão. A interação do usuário com o protótipo é feita através da solicitação para a realização da próxima transição, o que pode ser considerado como a simulação do pulso discreto que faz com que a máquina funcione.

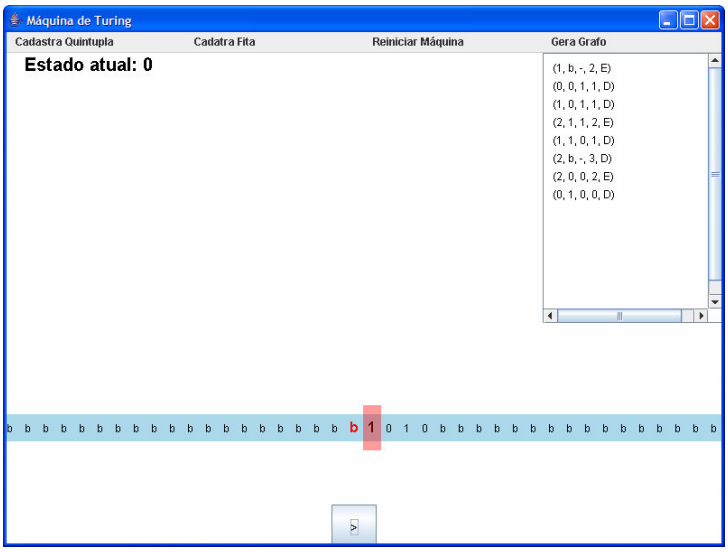

Figura 4. Tela principal do protótipo

Tratando-se de uma aplicação que utiliza geração de imagem para a criação da interface, foram utilizadas técnicas de programação para jogos, como "*Double Buffer*" para melhoria de performance, além do uso de bibliotecas gráficas básicas.

3.1. Controle da Fita

Tendo como base a teoria de Máquina de Turing, que define a fita como ilimitada, e sempre preenchida o símbolo *b* à esquerda e à direita, adotou-se a opção de não armazenar os valores em branco. Desta forma, foi escolhida, como estrutura de dados responsável pela representação da fita, uma lista ligada de objetos do tipo *string*, na qual serão elementos da mesma apenas os valores diferentes de branco. Assim, não existem limitações para o tamanho da fita, sendo este limite definido apenas pela capacidade de armazenamento de informação na memória do computador utilizado na simulação.

O algoritmo criado é responsável por remover os símbolos nas extremidades esquerda e direita da fita quando marcados como branco, e também criar novos símbolos na lista quando o *b* da extremidade for alterado para um outro símbolo do alfabeto válido. No caso de alteração da informação escrita, quando não for branco, o símbolo presente é alterado pelo novo símbolo informado na quíntupla, sem que sejam realizadas outras operações.

Com o intuito de facilitar a interação com o usuário, o protótipo dispõe de uma tela exclusiva para inserção e deleção de novos elementos na fita, conforme exibe a próxima figura.

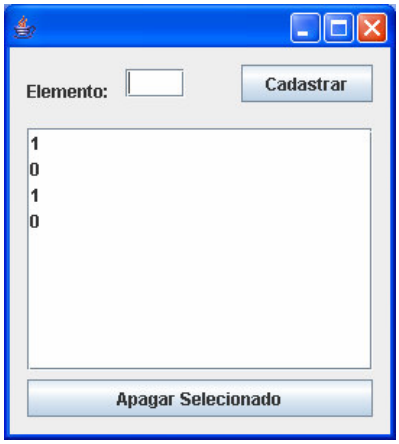

Figura 5. Tela de edição de fita

3.2. Quíntuplas

As quíntuplas são modeladas no protótipo como uma classe na qual são informados os valores para cada um dos cinco atributos. Todos os valores são representados como *strings*, exceto o sentido do movimento, que será representado pelo tipo *char*, sendo aceitos, no caso, apenas os caracteres D e E (direita e esquerda, respectivamente).

Como forma de facilitar a escrita da quíntupla na tela, foi sobrescrito o método *toString()*, para que, ao ser chamado, o mesmo retornasse um texto no formato apresentado pelo quadro seguinte.

(e, i, i', e', s) onde: e – Estado atual i – Símbolo lido i' – Símbolo a imprimir e' – Próximo estado s – Sentido do movimento

Quadro 1. Modelo de exibição das quíntuplas pelo método *toString()*

Visando uma melhor performance durante a execução do protótipo, e também na verificação para evitar ambigüidades, o que faria a máquina parar, as quíntuplas, são armazenadas na forma de tabela *hash*, implementada com a classe *HashMap*. Como chave, foi definido que seriam utilizados o *estado atual* e o *símbolo lido*, separados por vírgula.

A tela de edição de quíntuplas, da mesma forma que as demais, foi concebida como facilitadora para a interação entre usuário e aplicação, de forma que aquele possa incluir, alterar, consultar ou excluir os dados necessários à operacionalização do modelo.

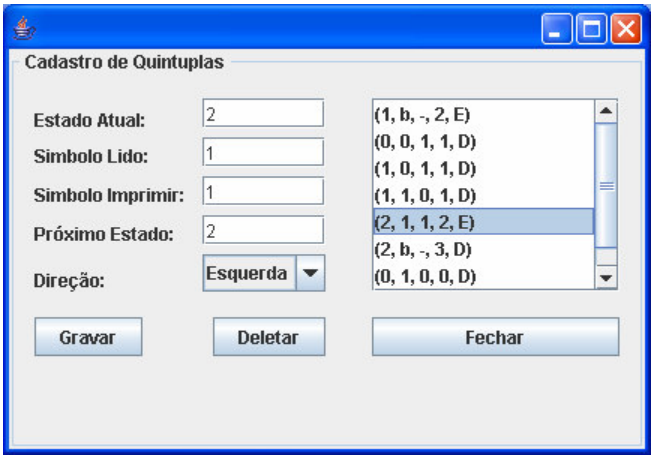

Figura 6. Tela de edição de quíntuplas

3.3. Transições

Para realizar as transições, a tabela *hash* é pesquisada, com a finalidade de encontrar a quíntupla que atenda ao estado atual e ao símbolo lido. Desta forma, retorna-se o objeto

representando a quíntupla que satisfaça a condição de “estado atual x símbolo lido”, sendo tal retorno armazenado para que seja utilizado na realização da transição.

De posse da quíntupla necessária para realizar a transição conhecida, o protótipo realiza a modificação do elemento que está sendo apontado na fita. Neste movimento, o protótipo verifica qual será o novo estado, realizando a alteração do estado atual de acordo com este dado. Terminados todos os passos, a aplicação verifica se o movimento da fita será para a esquerda ou para a direita.

Caso não exista uma quintupla que satisfaça a relação “elemento lido x estado atual”, o valor da quintupla a ser utilizada será vazio, não sendo possível realizar a transição, caracterizando o travamento da máquina (ocorrência prevista no modelo original).

```
Inicio
    quintupla := encontra_quintupla(estado_atual, simbolo_lido);
    fita.simbolo_atual := quintupla.símbolo_atual);
    estado_atual := quintupla.próximo_estado;
    SE quintupla.movimento = ‘E’ ENTÃO
        move_fita_esquerda();
    SENÃO
        move_fita_direita();
Fim;
```

Quadro 2. Pseudocódigo da transição

3.4. Movimento da Fita

O movimento da fita é dado pelo índice do símbolo da fita que está posicionado sob a cabeça de leitura, chamado de *itemAtual*. Desta forma, quando é realizado um movimento para a esquerda, o *itemAtual* é decrementado em uma unidade, para que a cabeça de leitura aponte para o símbolo a esquerda do atual, podendo receber valor negativo caso a cabeça de leitura esteja posicionada sobre o branco na extremidade esquerda da fita. Caso o movimento seja realizado à direita, o *itemAtual* será incrementado em uma unidade, fazendo com que a cabeça de leitura aponte para o símbolo à direita do atual, podendo o valor do índice ser igual à quantidade de elementos da fita (considerando que o índice é iniciado em zero), quando a cabeça de leitura estiver posicionada sobre o branco na extremidade direita da fita.

3.5. Geração do grafo de estados

Para geração do grafo de estados, foi criado um algoritmo que, primeiramente, percorre todas as quíntuplas cadastradas, armazenando todos os valores de estado atual e próximo estado que ainda não tenham sido cadastrados, dando um índice a cada novo estado encontrado.

Tendo todos os estados mapeados, uma matriz quadrada é criada para armazenar as adjacências, com o tamanho igual à quantidade de estados encontrados. Aqui, o índice das linhas e das colunas refere-se ao índice dado ao estado na primeira etapa do processo. A varredura das quíntuplas é realizada novamente, sendo verificado o estado atual e o próximo estado das mesmas. Cada quintupla é, então, inserida na posição referente à coordenada de linha referenciada pelo índice do estado atual e coordenada de coluna referenciada pelo índice do próximo estado.

No processo de geração da imagem, os estados são distribuídos na tela de forma aleatória, sendo interligados com retas.

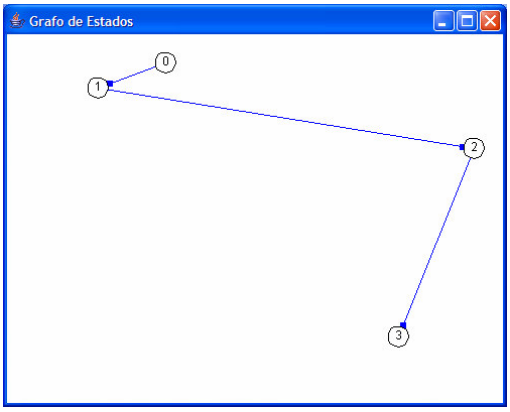

Figura 7. Grafo de estados gerado pelo protótipo

Indicando o estado de destino, é inserido um quadrado, que tem suas coordenadas centrais informadas através do pseudocódigo do próximo quadro. Utilizando o algoritmo proposto, é necessário verificar se o estado de destino está posicionado à direita do estado de origem. Em caso positivo, deve-se realizar uma normalização na equação para que o quadrado seja desenhado do lado oposto, pois o resultado do arco da tangente, neste caso, é um valor que compreende a metade direita da esfera do estado, e no caso de o estado de

destino estar a direita do estado de origem, o quadrado deverá ser desenhado do lado esquerdo. Neste último caso, é necessário adicionar 180º ao grau encontrado.

```
Inicio
      m = (y_fim – y_inicio)  / (x_fim – x_inicio);
      grau = ArcTan(m);

      SE (x_inicial - x_final < 0) ENTÃO
            grau = grau + 180;

      cosseno = Cosseno(grau);
      seno = Seno(grau);
      x = x_fim + (cosseno * 12);
      y = y_fim + (seno * 12);
Fim;
```

Quadro 3. Pseudocódigo para geração do quadrado indicador de direção

4. DISCUSSÃO DOS RESULTADOS

De forma a buscar subsídios para afirmar que o simulador aqui descrito pode representar um recurso adicional que auxilia de forma efetiva a estratégia pedagógica do professor, o mesmo foi submetido ao uso por parte dos alunos do sexto e oitavo semestres do curso de ciência da computação da Faculdade Politécnica de Jundiaí, de forma não-estruturada. Semelhantes testes se mostraram promissores, já que os alunos demonstraram entender o funcionamento da máquina e assimilar a teoria subjacente. Isto indica a possibilidade de implementar o uso do protótipo em situações reais de ensino, no âmbito de palestras, seminários e aulas. O uso em laboratório também foi bastante elogiado pelos discentes, uma vez que em semelhantes situações, há a possibilidade de interagir diretamente com o sistema, simulando situações encontradas em exercícios de disciplinas como Teoria da Computação. Além disso, no laboratório, a idéia de simulação se completa através da interação com o dispositivo, o que pode ser feito de forma individual ou em grupo. No segundo caso, percebeu-se a possibilidade de promover um aprendizado em colaboração consistente, uma vez que os resultados obtidos com o uso do programa podem ser amplamente discutidos, criticados e expostos, na forma de dúvidas ou comentários, ao professor e aos demais colegas.

Em um segundo momento, o protótipo foi testado através do uso por dez alunos de outra instituição, também estudantes de ciência da computação. Tais alunos foram selecionados aleatoriamente entre um grupo de estudantes que apresentava dificuldades na

compreensão dos conceitos trabalhados na disciplina “Teoria da Computação”, componente do sexto semestre (o curso todo tem oito semestres). Após utilizarem o sistema para estudo, questões equivalentes às relacionadas em provas nas quais os alunos em questão tinham encontrado dificuldades foram elaboradas pelo professor. Apuradas as respostas, constatou-se que:

- Nenhum aluno obteve resultado menor do que aquele conseguido na prova oficial;
- Três alunos obtiveram resultados semelhantes aos obtidos na avaliação anterior;
- Três alunos melhoraram o desempenho em relação aos conceitos obtidos na prova oficial, mas ainda permaneceram abaixo do conceito mínimo para aprovação (aumento médio de 31% nos acertos);
- Quatro alunos ampliaram seus desempenhos pessoais, atingindo conceitos iguais ou superiores aos necessários para aprovação (aumento médio de 39% nos acertos).

Ou seja, nos testes iniciais, as impressões e os dados recolhidos permitem afirmar que o protótipo pode auxiliar no ensino e na aprendizagem do conceito de máquina de Turing, já que a maioria dos usuários revelou progressos no entendimento do assunto em foco.

5. CONSIDERAÇÕES FINAIS

A primeira consideração necessária é a de que o protótipo deve passar por mais amplos testes, de modo a recolher impressões de um número maior de usuários. Os resultados iniciais foram promissores, permitindo supor que o refinamento do sistema, com a inclusão de outros tópicos da mesma área de conhecimento e de áreas correlatas, tem amplas perspectivas de sucesso na composição de uma ferramenta de suporte ao trabalho docente. Este refinamento pode tocar tanto os algoritmos que determinam o funcionamento da máquina, quanto as interfaces que promovem a interação com alunos e professores. Além disso, a eficiência do emprego do mesmo como instrumento auxiliar do processo de ensino-aprendizagem, apesar das indicações mencionadas, pede, também, uma testagem mais intensa, talvez com um número maior de sujeitos e com metodologias experimentais.

A experiência de construção de um simulador que tem por finalidade implementar o uso de um modelo cuja teoria não se furta à complexidade deixou a impressão que o uso de dispositivos computacionais como parte da estratégia pedagógica nos cursos superiores é um campo a ser explorado largamente. Inúmeras possibilidades acenam para a confecção de outros tantos aplicativos que venham a lidar com teorias em relação às quais surgem inúmeras dificuldades no âmbito dos processos de ensino-aprendizagem. Os cursos de ciência da computação têm diversos assuntos que podem ser encarados desta forma, mas isso não é muito diferente nas disciplinas referentes às engenharias ou à matemática, por exemplo, para as quais o conceito aqui implementado poderia ser expandido.

O percurso para a criação de semelhantes modelos demonstrou, pelo menos no que diz respeito a esta investigação, constituir-se em um desafio, tamanho o número de descobertas e de novas necessidades que se interpunham entre a teoria em si e sua implementação no dispositivo. Estes impasses fizeram com que a teoria fosse constantemente revisitada, o que levava a um trabalho adicional de construção de códigos em Java – e de testes exaustivos dos mesmos, até que uma solução satisfatória fosse encontrada. E isso tudo em relação a uma tese apresentada em 1936! Este fato não deixa de causar a impressão de que, dado o caráter incontrolável da expansão das possibilidades de conhecimento nos dias atuais, as simulações computacionais possam vir a ter uma maior relevância na busca por compreensão de fenômenos, teses e teoremas de relevo para a formação do estudante contemporâneo.

Não é, sobretudo, um trabalho simples, nem tampouco fácil. Mas a experiência é gratificante, à medida que se percebe que a teoria, tantas vezes incompreendida no âmbito da sala de aula, pode surgir com novos contornos, apresentada de uma forma que venha a possibilitar outros meios para sua assimilação.

6. BIBLIOGRAFIA

DIVÉRIO, Tiaraju A; MENEZES, Paulo B. *Teoria da computação: máquinas universais e computabilidade*. Porto Alegre: Sagra-Luzzato, 2000.

FONSECA, Ijar. Máquinas de Estado Finito. Disponível em: http://www2.dem.inpe.br/ijar/MaqFin1.doc Acesso em: 01 dez. 2005.

GERSTING, Judith L. *Fundamentos matemáticos para ciência da computação*. 4.ed. Rio de Janeiro: LTC, 2001.

HORSTMANN, Cay; trad. FURMANKIEWICZ, Edson. Big Java. Bookman: Porto Alegre: 2004.

Javadoc. Sun Microsystem. Disponível em: <http://java.sun.com/j2se/javadoc/>. Acessado em: 11 set. 2006.

MORIN, Edgar. *Os sete saberes necessários à educação do futuro*. 5.ed. São Paulo: Cortez; Brasília, DF : UNESCO, 2002.

MORELLI, Ralph; WALDE, Ralph; *Java, Java, Java: Object Oriented Problem Solving*.3 ed. Estados Unidos: Prentice Hall.

OLIVEIRA, Gerson Pastre. Teoria da Computação (mimeo). Jundiaí: FPJ, 2005.

OLIVEIRA, Gerson P. Construção coletiva do conhecimento através de uma experiência de incentivo à autonomia dos estudantes no aprendizado de matemática discreta. *Anais do VII Encontro Paulista de Educação Matemática*, p. 136-137, 2004.

OLIVEIRA, Gerson P. O Uso de Sistemas Computacionais na Avaliação Formativa de Estudantes. *Dissertação de Mestrado*. Universidade São Francisco: Bragança Paulista, 2000.