GAM - Um Simulador para Auxiliar o Ensino de Linguagens Formais e de Autômatos

Anibal S. Jukemura[1], Hugo A. D. do Nascimento[1], Joaquim Q. Uchôa[2]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
CEP 74001-970, Caixa Postal 131, Goiânia, GO

[2]Departamento de Ciências da Computação, Universidade Federal de Lavras (UFLA),
CEP: 37200-000, Lavras, MG

{anibal,hadn}@inf.ufg.br, joukim@comp.ufla.br

Abstract. *A Finite Automaton is a mathematical model of a finite-state machine that reads symbols from a tape and accepts or rejects its input. The Theory of Finite Automata is important for modeling real machines with a finite set of states, for compacting dictionaries, and for several others applications. This paper describes a tool for helping students and teachers to study the Theory of Finite Automata. The tool provides conditions for simulating different types of abstract machines in order to perform several operations and conversions that are common in this area of study.*
Keywords: *finite automata, simulator, learning tool.*

Resumo. *Um Autômato Finito é um modelo matemático de uma máquina de estados finitos que lê símbolos de uma fita e aceita ou rejeita sua entrada. A Teoria de Autômatos Finitos é importante para a modelagem de máquinas reais com um conjunto finito de estados, para compactação de dicionários, e para várias outras aplicações. Este artigo descreve uma ferramenta para auxiliar estudantes e professores a estudarem a Teoria dos Autômatos Finitos. A ferramenta fornece condições de simular diferentes tipos de máquinas abstratas a fim de executar diversas operações e conversões que são comuns nessa área de estudo.*
Palavras-chave: *autômatos finitos, simulador, ferramenta de aprendizagem.*

1. Introdução

O estudo de Linguagens Formais e Autômatos é imprescindível para a formação de profissionais em Ciências da Computação e em áreas de conhecimento afins. As definições e os problemas vistos nesse estudo são a base para compreender conceitos mais complexos em Teoria da Computação, tais como decidibilidade e complexidade de problemas. Além disso, a Teoria de Autômatos tem aplicações diretas na modelagem de vários sistemas físicos de estados finitos e na construção de compiladores e interpretadores para linguagens de programação. Até mesmo processadores de texto fazem uso de autômatos finitos para realizarem a busca de *substrings* (sub-cadeias de

caracteres) ou a verificação ortográfica de palavras de uma forma eficiente [Kowaltowki et. al 1998, Unitex 2005].

Embora a Teoria de Linguagens Formais e de Autômatos seja amplamente pesquisada, percebe-se que há uma escassez de ferramentas de apoio ao aprendizado desse tema.

As ferramentas computacionais específicas podem ser utilizadas efetivamente para auxiliar no aprendizado de conteúdos da Computação. Em geral, o emprego de tais ferramentas motivam o aluno para o estudo da disciplina, permitem testar rapidamente novas idéias ou conceitos já ensinados, além de oferecerem um meio para correlacionar aspectos teóricos e práticos do conteúdo.

O objetivo do presente trabalho é, portanto, apresentar uma ferramenta chamada GAM (*Ginux Abstract Machine*), que auxilie no estudo de Linguagens Formais e Autômatos nos Cursos de Ciências da Computação, suprindo assim a necessidade por um ambiente de ensino para essa disciplina. Note que existem ferramentas similares para tal fim. Contudo, nenhumas dessas ferramentas possuem a flexibilidade de extensão vista na GAM, como será discutido posteriormente neste artigo.

O restante deste documento está organizado como segue: a Seção 2 descreve conceitos teóricos básicos sobre autômatos. A Seção 3 apresenta a ferramenta GAM. Na Seção 4 é feita uma comparação das características da GAM com outros sistemas similares. Finalmente, na Seção 5, são feitas algumas considerações sobre a aplicabilidade e extensibilidade funcional da GAM.

2. Definições básicas sobre a Teoria de Autômatos

As definições encontradas nesta seção são baseadas nos trabalhos de Aho [Aho 1973], Hopcroft [Hopcroft 1979], Menezes [Menezes 2000] e Sudkamp [Sudkamp 1997].[*]

Um *autômato finito* é um modelo matemático de uma máquina computacional de estados finitos que lê uma seqüência de símbolos (uma *string*) de uma fita e aceita ou rejeita essa entrada. Tal modelo é composto de três partes:

a) **Uma Fita** – consiste de um dispositivo de entrada que contém a informação (símbolos) a ser processada;

b) **Uma Unidade de Controle** – é composta por uma leitora (cabeça de leitura) que acessa uma célula da fita de cada vez e movimenta-se para uma nova posição na fita;

c) **Um Programa ou Função de Transição "delta"** – é uma função que, dado um estado atual da máquina e um símbolo lido da fita, determina uma mudança de estado.

Um autômato finito começa em um ou mais estados iniciais e lê uma seqüência de símbolos da fita. Durante o processo de leitura, o autômato muda de estado de acordo

[*] Definições adicionais em Teoria de Autômatos podem ser encontradas nas referências citadas.

com a sua função programa. Uma vez lida a seqüência de símbolos, o autômato pára sua execução, e emite uma mensagem dizendo se aceita ou rejeita essa seqüência. O sinal de aceitação é emitido, se e somente se o autômato tiver parado em um estado previamente definido como final.

O conjunto de todas as *strings* aceitas por um autômato formam a linguagem aceita por essa máquina.

Os autômatos normalmente estão classificados nos seguintes grupos:

- Autômatos Finitos Determinísticos (AFD): são máquinas que apresentam uma função programa "delta" que, para cada estado e cada símbolo, possui uma única transição bem definida. Além disso, o autômato começa em um único estado inicial e a leitora da máquina avança imediatamente para a próxima posição da fita após a leitura de um símbolo.
- Autômatos Finitos Não-Determinísticos (AFND): são máquinas semelhantes às da classe determinística, diferenciando-se na função programa "delta" que, ao processar uma entrada composta pelo estado corrente e um símbolo lido, tem como resultado um conjunto de novos estados. A máquina pode ter também mais de um estado inicial. O funcionamento de um autômato não-determinístico pode ser simulado através de uma multiplicação da unidade de controle de um autômato determinístico.
- Autômatos Finitos Não-Determinísticos com Movimentos Vazios (AFND-ε): são máquinas semelhantes às da classe não-determinística, com o adicional de poderem realizar uma transição de estados sem a leitura de um símbolo da fita.
- Autômatos de Pilhas ou Autômatos *push-down* (APD): são máquinas análogas aos Autômatos Finitos anteriores, mas com a inclusão de uma pilha que funciona como memória auxiliar.
- Máquinas de Turing (MT): é o formalismo mais poderoso dentre as quais foram citadas. Na realidade os autômatos desta classe são capazes de executar quaisquer tarefas que sejam efetivamente computáveis.

Existem algumas operações básicas que são realizadas sobre autômatos, entre elas destacam-se:

1. Análise e validação de cadeias de entrada: ao se analisar uma *string* de entrada, a máquina de estados é capaz de determinar se essa cadeia de caracteres é um elemento ou não da linguagem representada.
2. Conversão de um AFND para um AFD e vice-versa: essa operação comprova que essas duas classes são equivalentes.
3. Minimização de um AFD: o objetivo da minimização é gerar um autômato finito equivalente ao autômato original, mas com o menor número de estados possível.

Mas qual é a relação entre Linguagens Formais e Autômatos? Uma linguagem, de uma forma geral, é uma reunião de um conjunto finito de símbolos e um conjunto de regras de expressões que, devidamente formuladas, representam uma forma clara de comunicação. Em Ciências da Computação, a Teoria de Linguagens Formais representa o estudo dos modelos matemáticos que possibilitam a especificação, o reconhecimento, a classificação, as propriedades e o inter-relacionamento entre as linguagens. Tais modelos matemáticos são formalmente representados pelos autômatos.

3. A Ferramenta GAM

A *Ginux Abstract Machine* (GAM) é uma ferramenta *opensource* (código aberto) que pode ser utilizada em sala de aula ou em laboratório, como parte de um material de apoio para o aprendizado de conceitos em Teoria de Linguagens Formais, Teoria da Computação e Compiladores. Ela oferece os seguintes recursos:

1. módulos para a entrada e teste de máquinas formuladas por alunos e pelo professor;
2. suporte a diversos tipos de autômatos;
3. interface gráfica para visualização da máquina de entrada;
4. possibilidade de acompanhar passo-a-passo a execução da função de transição “delta”; e
5. operações de conversão e minimização de autômatos.

A GAM se originou de um trabalho de Especialização do Departamento de Computação da Universidade Federal de Lavras, Minas Gerais [Jukemura 2004a]. A ferramenta pode ser usada em ambientes GNU/Linux, Windows e em sistemas operacionais que suportem GTK e GTKmm, estando disponível para *download* em *http://www.ufgvirtual.ufg.br/~anibal/ginux*.

3.1. Estrutura Interna

A estrutura interna da ferramenta GAM consiste de seis elementos distintos (ver Figura 1):

- **Usuário**: é quem fornece os dados necessários para compor um autômato, e por realizar operações de simulação, conversão e minimização do mesmo.
- **Módulo principal de visualização (interface gráfica)**: este módulo apresenta os autômatos graficamente como um grafo direcionado. O módulo de visualização atua também como uma interface gráfica, permitindo ao usuário modificar o autômato e interagir com os demais recursos do sistema.
- **Módulo dos autômatos**: este módulo é composto pela classe que armazena funcionalmente os autômatos que serão manipulados pela GAM.
- **Módulo de minimização de AFDs**: é chamado pelo usuário para minimizar um autômato AFD. O resultado, o autômato mínimo, é apresentado na tela através do módulo principal de visualização.

- **Módulo de conversão de um AFND para um AFD**: este é o módulo responsável por aplicar o algoritmo de conversão de um AFND para um AFD, com a opção de expor quais estados foram convertidos.
- **Módulo de Entrada/Saída**: este módulo apresenta funções para salvar e recuperar os autômatos através do disco rígido.

A versão atual da GAM possui, contudo, algumas limitações. São elas:

- os símbolos para os alfabetos de entrada, saída e auxiliar foram limitados a uma quantidade máxima de três e as máquinas a uma quantidade de até dez estados;
- o quadro de desenhos é estático, portanto, ele não fornece liberdade ao usuário para manipular diretamente os componentes das máquinas;
- os símbolos das transições, em algumas situações, ficam superpostos, ocasionando dificuldades na leitura. Essa situação, porém, é contornada com o uso da interface para expor a função programa que mapeia todas as transições.
- as classes APD e MT são apenas expostas graficamente. A GAM ainda não consegue ler e interpretar *strings* de entrada para essas duas classes.

É importante ressaltar que as limitações da ferramenta são conseqüências do tempo limitado durante o seu desenvolvimento como trabalho de Especialização do Departamento de Computação da Universidade Federal de Lavras. Portanto, sua complementação funcional será realizada futuramente.

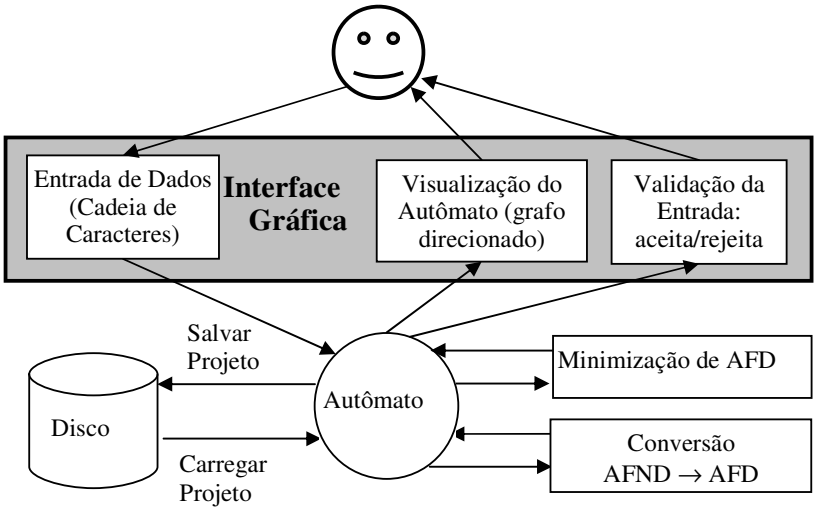

Figura 1. Estrutura funcional da GAM

3.2. Interface Gráfica

Ao abrir a ferramenta, o usuário irá encontrar a janela principal do projeto que está dividida em quatro blocos: o menu principal, a janela de desenhos, a caixa de análise de entradas e a barra de ferramentas (como ilustrado na Figura 2).

Através do menu principal, o usuário seleciona o tipo de máquina a ser simulada. As operações de minimização de AFDs e conversão de AFNDs para AFDs estão também acessíveis através deste bloco (como pode ser visto na Figura 3).

O usuário fornece os dados necessários para a confecção do autômato, como o estado inicial, o estado final, o alfabeto de entrada e a função programa "delta" através da barra de ferramentas (ver Figura 2).

O bloco de janelas de desenhos ilustra os autômatos como grafos direcionados.

A caixa de análise de entradas reúne os elementos necessários para o usuário executar passo-a-passo a validação de uma cadeia de caracteres pelo autômato por ele montado.

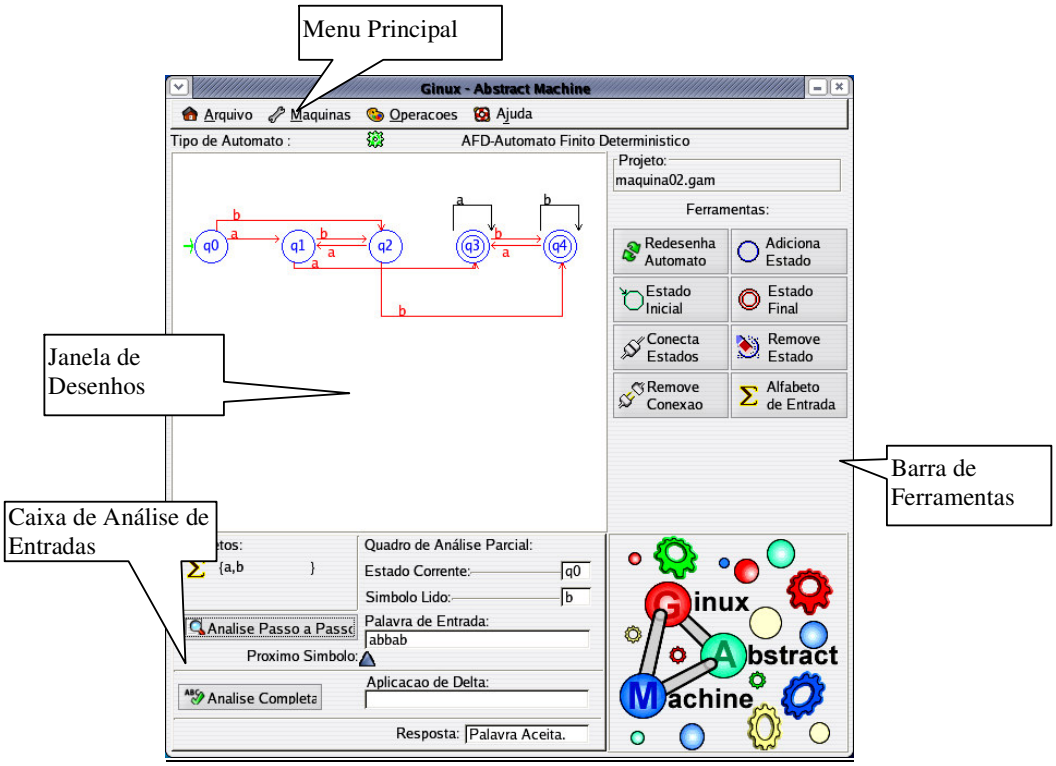

Figura 2. A interface gráfica da GAM

Figura 3. Escolha do tipo de máquina e operações

3.3. Exemplo de Funcionamento da GAM

As Figuras de 4, 5 e 6 apresentam imagens que ilustram o funcionamento da GAM. Inicialmente, seleciona-se o tipo de máquina a ser estudada (ver Figura 3). O usuário deve agora usar o grupo de botões do painel de ferramentas para montar graficamente o autômato (ver Figura 2, apresentada anteriormente). Assim, o usuário define o alfabeto de entrada, fornece o estado inicial, define quais são os estados finais, e monta a função programa "delta", conectando os estados.

Caso a máquina seja um AFND, o usuário pode optar por acessar o menu de "Operações" e clicar em "Conversão de AFND para AFD", a fim de obter um AFD como resultado (ver Figura 4).

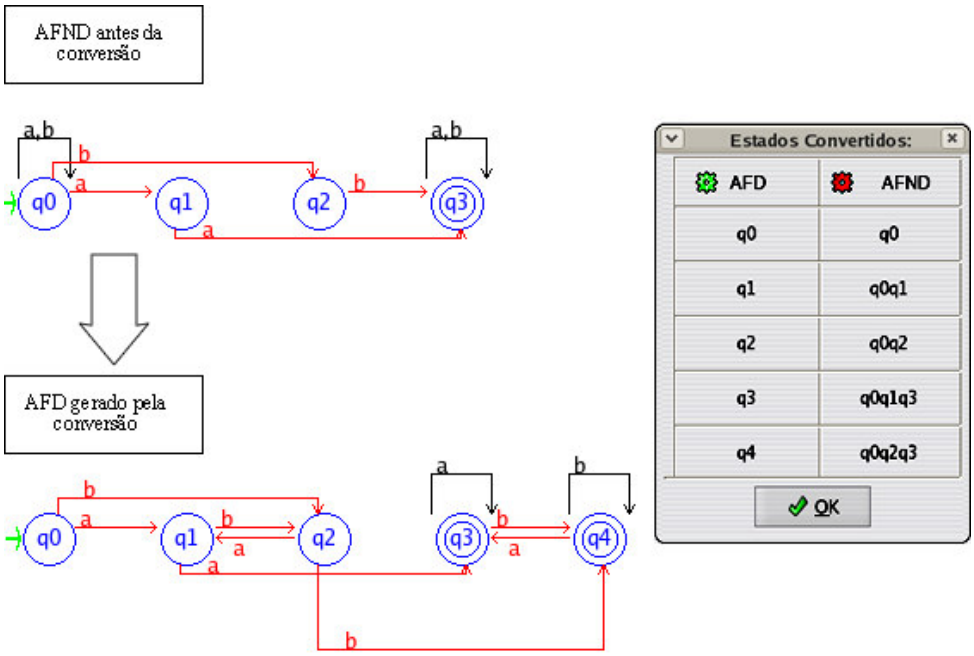

Figura 4. Conversão de um AFND para um AFD

No caso da máquina ser um AFD e, mesmo depois de uma conversão de um AFND para um AFD, o usuário pode executar a operação de minimização, acessando no menu principal a opção "Minimização de AFD" (ver Figura 3).

Ao minimizar um AFD, o resultado é automaticamente ilustrado na janela de desenhos e, adicionalmente, é mostrada uma outra janela contendo as informações sobre os estados minimizados do AFD original (como ilustra a Figura 5).

A qualquer momento, o usuário pode digitar uma cadeia de entrada e selecionar o botão de análise na caixa de análise de entradas, para receber a resposta de aceitação ou de rejeição da cadeia (ver Figura 6).

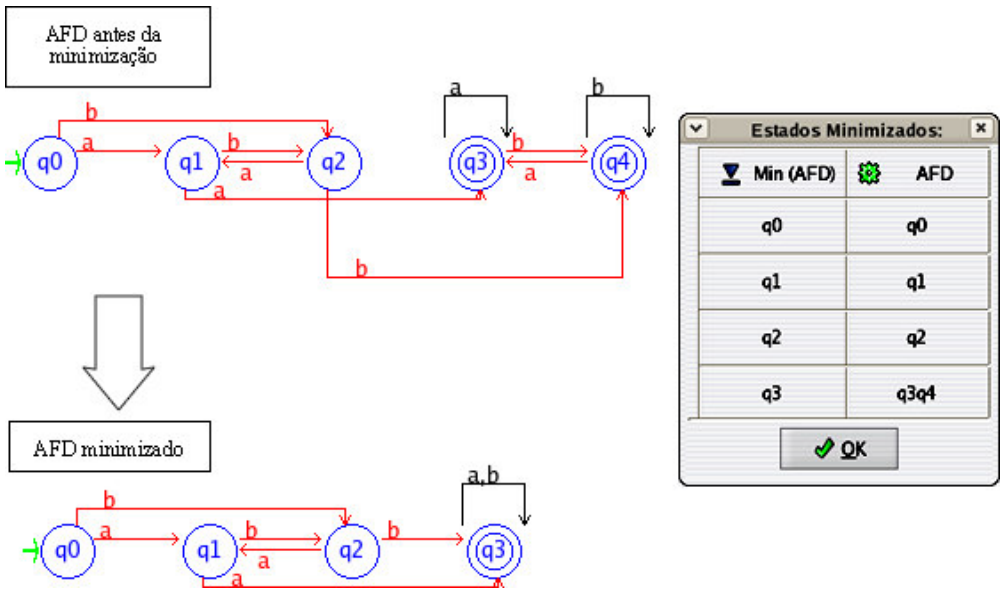

Figura 5. Minimização de um AFD

Ressalta-se que as simulações de cadeias de caracteres de entrada para os autômatos são atividades que demandam muito tempo. Professores que optam pelo uso de procedimentos normais em sala de aula para demonstrar tais simulações acabam por desperdiçar um tempo valioso.

A ferramenta GAM pode ser usada para simular uma grande quantidade de cadeias de entrada em um curto espaço de tempo, além de facilitar o estudo de máquinas com um número de estados razoáveis e *strings* com até 20 caracteres de comprimento. A contribuição à melhoria do aprendizado é, portanto, fator de considerável importância.

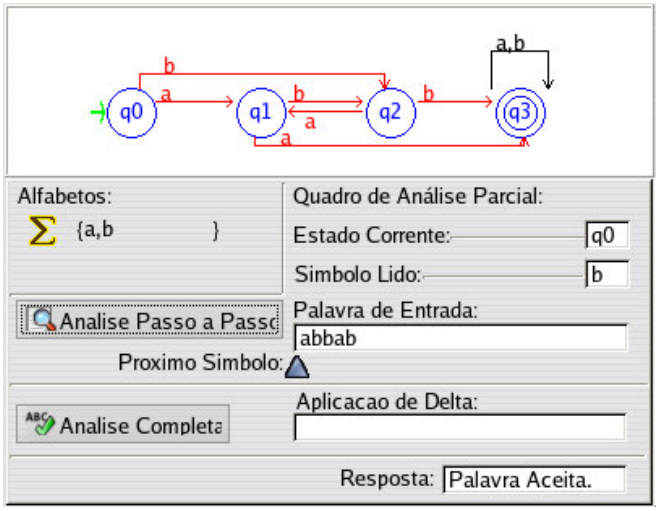

Figura 6. Análise de uma entrada por um AFD

4. Avaliação da Ferramenta

Para testar a funcionalidade, usabilidade e o papel didático da ferramenta GAM, foi elaborado um experimento prático com usuários reais, conforme é descrito a seguir.

4.1. Planejamento do Experimento

Vinte e cinco alunos do terceiro e quarto anos do curso de Ciências da Computação do Instituto de Informática (INF) da Universidade Federal de Goiás que estão cursando a disciplina de Linguagens Formais e Autômatos tomaram parte da avaliação da ferramenta.

O experimento foi dividido em duas baterias de testes com dez alunos na primeira bateria e quinze alunos na segunda, utilizando-se o laboratório GNU/Linux de pós-graduação do INF. Os testes foram distribuídos em três estágios:

Estágio 1: nessa parte do experimento, os alunos responderam a uma lista de cinco exercícios contendo questões relacionadas com a construção de AFNDs, conversão entre AFNDs e AFDs, análise de palavras de entrada e minimização de AFDs. Para essa parte, foi previsto um tempo de 10min.

Estágio 2: uma vez respondida a lista de exercícios, cada aluno teve acesso à ferramenta GAM, a fim de validar suas respostas.

Estágio 3: esse estágio compreendeu o preenchimento de um questionário sobre a utilidade e a usabilidade da ferramenta GAM pelo aluno. Foi planejado um período de 15 minutos para a realização dessa atividade.

4.2. Resultados Encontrados

Para o Estágio 1 do experimento, os alunos gastaram em média, 12min para resolver todos os exercícios propostos. É interessante observar que todos demonstraram domínio do conteúdo da disciplina, até porque já haviam estudado o assunto da lista de exercício no bimestre anterior.

O Estágio 2 durou de 15 a 20 minutos e, para preencherem a ficha no Estágio 3, os alunos gastaram de 10 a 13 minutos.

Pelas respostas obtidas através do experimento, percebeu-se uma unanimidade com relação à facilidade de uso do software, visto que todos não tiveram dificuldades de aprenderem por si próprios a manipularem suas funções (lembrando que não houve um treinamento prévio). Isso tudo fez com que o Estágio 2 do experimento não se prolongasse tanto.

Com relação à instalação da ferramenta em ambiente GNU/Linux,

- 56% acharam ser de fácil instalação;
- 28% acharam o procedimento difícil;
- 4% acharam muito difícil;
- 12% não opinaram.

Os alunos relataram que a ferramenta funcionou sem nenhum problema. Somente um aluno encontrou um "*bug*" (falha) no software com relação à aceitação de palavras contendo caracteres não pertencentes ao alfabeto.

Todos foram unânimes em afirmar que a ferramenta apresenta um manual bem descrito e que mensagens de erro apresentadas durante o seu uso direcionam bem o usuário no momento de aprender a utilizar a aplicação. A maioria dos usuários afirmou que a interface é agradável e intuitiva.

Os alunos também utilizaram efetivamente a ferramenta para verificar conceitos de Teoria da Computação e de Linguagens Formais e Autômatos. Foi observado por exemplo, que praticamente todos empregaram bem a ferramenta para confirmar as respostas da lista de exercícios.

Os alunos afirmaram que o projeto deveria ser continuado, a fim de que novas funcionalidades possam ser agregadas.

Com base nos resultados obtidos, percebe-se que a ferramenta pode ser utilizada para que:

- os alunos corrijam seus próprios exercícios;
- os alunos verifiquem se conseguem realmente minimizar AFDs, converter AFNDs para AFDs e aplicar a função "delta" para a aceitação ou rejeição de palavras de entrada pelo autômato construído;
- possa, de uma forma geral, sanar as dificuldades encontradas durante o primeiro contato com a teoria de autômatos;
- motive os alunos no estudo de Linguagens Formais e Autômatos.

5. Ferramentas Similares

É importante observar que existem outras ferramentas disponíveis na literatura para o estudo de linguagens formas e autômatos. Dentre elas, encontram-se: DFApplet, VAS (*Visual Automata Simulator*) e *Visual Turing*. A seguir, é apresentada uma descrição sucinta de cada uma dessas ferramentas:

1. **DFApplet** [Chiristin 2005] – foi escrita em Linguagem JAVA pela Universidade de Berkeley; possui somente a função de simulação de um AFD, e é distribuída através de uma licença GPL.
2. **VAS** [Bovet 2005] – é uma ferramenta de código livre feita em Linguagem JAVA pela Universidade de São Francisco (USFCA). Com relação à GAM, apresenta adicionalmente, recursos para simular Máquinas de Turing, porém não executa minimização de AFDs.
3. **Visual Turing** [Cheran 2005] – é uma ferramenta feita em Linguagem JAVA que apresenta somente simulação de Máquinas de Turing. O Visual Turing apresenta uma versão comercial completa e uma versão limitada através de uma licença GPL.

Essas ferramentas não possuem tantas funcionalidades como a GAM. A Tabela 2 traz uma comparação entre as mesmas.

Tabela 2. Quadro Comparativo

Características		Ferramentas			
		DFApplet – 1.0	**VAS**	**Visual Turing**	**GAM**
Sistema Operacional	Windows	√	√	√	√
	GNU/Linux	√	√	√	√
	MAC/OS	√	√	√	
Licença	GPL	√	√	√	√
	Comercial			√	
Funções	Simular AFD	√	√		√
	Simular AFND		√		√
	Simular AFND-ε				*
	AFD → AFND		√		√
	AFD → ER^+				*
	ER → AFND-e				*
	Minimizar AFD				√
	Simular APD				*
	Simular MT		√	√	*

(*) - Essas funções serão implementadas futuramente

(+) - Expressões Regulares

6. Conclusão

A GAM foi implementada com as funções de simulação de AFDs e AFNDs, com opções de análise passo a passo e análise total para *strings* de entrada com até vinte caracteres. Com o auxílio da interface gráfica que expõe a função programa, fica evidente a facilidade de estudar essas classes de uma forma muito clara e prática.

A principal vantagem da ferramenta encontra-se nas funções de simulação de AFD e AFND, juntamente com as operações de conversão de AFNDs para AFDs e minimização de AFDs, as quais fornecem uma excelente opção prática à exposição teórica do estudo desses itens, além de economizar muito tempo em simulações de cadeias de caracteres de entrada para os autômatos durante demonstrações em sala de aula.

Adicionalmente, o projeto foi desenvolvido sob licença GPL. Com a sua continuidade, em conjunto com a divulgação à comunidade livre, espera-se o surgimento de muitas contribuições para o desenvolvimento das próximas versões. Propõe-se de imediato a implantação das seguintes operações em uma nova versão:

- análise de cadeia de entrada para os modelos de APDs e MTs;
- simulação passo a passo da pilha auxiliar para os APDs; e
- simulação completa de AFND-ε, APDs, MTs.

Outra interessante função que pode ser incluída nessa ferramenta é a alteração do módulo de desenho gráfico das máquinas abstratas para que ele suporte mais estados e também possa fornecer liberdade de manipulação de componentes pelo usuário.

Referências

Aho, A. V., *Currents in the Theory of Computing*, Prentice Hall, 1973

Bovet Jean*, Visual Automata Simulator.* [on-line]. Disponível na internet via http://www.cs.usfca.edu/~jbovet/vas.html. Arquivo capturado em janeiro de 2005.

Cheran Cristian, *Visual Turing*. [on-line]. Disponível na internet via http://sourceforge.net. Arquivo capturado em janeiro de 2005.

Christin Nicolas, *DFApplet-1.0*. [on-line]. Disponível na internet via http://www.sims.berkeley.edu/~christin/dfa/. Arquivo capturado em janeiro de 2005.

Hopcroft, John E., *Introduction to automata theory, languages and computation*. USA: Addison-Wesley Publishing Company, Inc. 1979. P.1-170.

Jukemura, Anibal S., *GAM – Ginux Abstract Machine*. Monografia de Especialização.Lavras, MG. Ed. UFLA-FAEPE. 2004.

Keller, Rodrigo dos Santos; Schereiber, Jacques N. C.; (Artigo Completo) *GEO-3D: A Realidade Virtual como Suporte ao Ensino da Geometria Espacial*. Anais do WISE99 - Workshop Internacional Sobre Educação Virtual. 9-11 de Dezembro de 1999. Fortaleza – CE.

Kowaltowki, Tomasz; Lucchesi Cláudio L.; Stolfi, Jorge. *Finite Automata and Efficient Lexicon Implementation*. Relatório Técnico IC-98-2, Instituto de Computação - Unicamp, Janeiro de 1998. 13 páginas.

Menezes, Paulo F. Blauth. *Linguagens Formais e Autômatos*. Porto Alegre: Ed. Sagra Luzzatto, 2000. 160p.

Sudkamp, Thomas A., *Languages and Machines*. USA: Addison-Wesley Publishing Company, Inc. Segunda Edição, 1997. P.155-295

Unitex-PB, *Compress*. [on-line]. Disponível na internet via http://www.nilc.icmc.usp.br:8180/unitex-pb/compactador.html. Arquivo capturado em fevereiro de 2005.