Melhorias de um framework para o estudo de elementos de Teoria da Computação

Cléberson Alessandro de Assis[1], **Emerson Tavares**[1]

[1]Departamento de Ciência da Computação – Universidade Presidente Antônio Carlos (UNIPAC)
Rua Palma Bageto Viol S/N – Barbacena – MG – Brasil

cleberson_assis@hotmail.com, emersontavares@yahoo.com.br

Abstract. *The paper presents the implementation of improvements to an educational tool geared to help in the teaching of the theory of computation. The tool aims to provide students to create and validate mechanisms of recognition through an interactive graphical user interface and easy usability, trying to ease the difficulties encountered by students when they start the study of theory of computing.*

Resumo. *O trabalho apresenta a implementação de melhorias em uma ferramenta educativa voltada para o auxílio no ensino da disciplina de Teoria da Computação. A ferramenta objetiva proporcionar ao aluno a criação e validação de mecanismos de reconhecimento através de uma interface gráfica interativa e de fácil usabilidade, tentando assim amenizar a dificuldade encontrada pelos alunos quando iniciam o estudo de teoria da computação.*
Palavras chave: autômatos, expressões regulares, gramáticas, linguagens formais, reconhecedores de padrão.

1. Introdução

A disciplina Teoria da Computação ou também conhecida como Teoria das Linguagens Formais, foi originalmente proposta na década de 1950 e teve como objetivo desenvolver teorias relacionadas com as linguagens naturais. No entanto, logo foi verificado que esta teoria era importante para o estudo de linguagens artificiais e em especial, para as linguagens originárias da Computação e da Informática [MENEZES 2005].

Este trabalho apresenta a adição e alteração de algumas funcionalidades presentes na ferramenta de auxilio no ensinamento da disciplina de Teoria da Computação. O software conta com uma interface gráfica de fácil usabilidade, que permite ao usuário maior facilidade e um melhor entendimento na sua utilização. A ferramenta proposta tem como principal finalidade auxiliar no processo de ensino na disciplina de Teoria da Computação, fazendo o uso de uma interface gráfica para simular o conceito de maquina de Turing.

1.1. Justificativa

A disciplina de Teoria da Computação é uma matéria em que os alunos geralmente apresentam dificuldade no aprendizado, devido ao alto grau de abstração exigido. Uma proposta pra tentar amenizar esta dificuldade, seria o uso de um software com o propósito educativo [DIVERIO and MENEZES 2000], que simule a criação de autômatos DFA - Autômatos de estados finito determinísticos (Deterministic Finite Automaton) ou NFA - Autômatos de estados finito não determinístico (Nondeterministic Finite Automaton) e a

partir destes autômatos ser possível gerar a Expressão Regular correpondente, realizar a conversão de um autômato NFA para um autômato DFA, contruir a matriz de transição e a gramática do autômato, validar se uma dada sequencia é valida, dentre outras funcionalidades.

A implementação das modificações na ferramenta de auxilio no ensinamento da disciplina de Teoria da Computação, tem como finalidade melhorar e simplificar a interface do sistema com o usuário, fazendo com que os componentes do software sejam facilmente compreendidos por quem o utiliza.

1.2. Análise e Objetivos

O seguinte trabalho consistiu na analise e aprimoramento do software Conversor e Validador de Sequências de Autômatos [SANDY 2010]. Este software contém as seguintes características.

- a ferramenta permite gerar um autômato DFA ou NFA a partir de um arquivo de configuração criado manualmente pelo usuário;
- exibe separadamente as propriedades do autômato, tais como:
  - Alfabeto;
  - Estados;
  - Estado Inicial;
  - Estados Finais.
- gera a ER(Expressão Regular) para o autômato;
- gera a Matriz de Transição;
- visualiza a imagem do autômato;
- faz a conversão da matriz de transição de NFA para DFA;
- validação do autômato por uma sequência de entrada;
- visualização da gramática a partir do autômato.

Com o estudo do software analisado verificou-se a necessidade e a possibilidade de aprimorar a ferramenta em alguns pontos, como:

- uma melhor interação entre o usuário e o software através de uma interface gráfica de fácil usabilidade;
- permitir a manipulação de alfabeto vazio no autômato;
- otimização do algoritmo de conversão de NFA para DFA;
- otimização do algoritmo de geração da Expressão Regular;
- otimização do algoritmo de criação da gramática;
- permitir salvar a imagem do autômato.

2. Revisão de bibliográfia

A disciplina de teoria da computação explica conceitos e princípios que ajudam a entender a natureza geral da computação através de modelos computacionais abstratos.

Para a modelagem do hardware de um computador é utilizado o conceito de autômato. Autômato é uma Construção que possui todas as características indispensáveis de um computador. Um Autômato funciona como um reconhecedor e serve para modelar uma maquina ou mesmo um computador simples. Através de reconhecedores é possível identificar se a cadeia de símbolos pertence ou não a uma determinada linguagem.

Linguagem formal é uma abstração das características gerais de uma linguagem de programação, onde possui um conjunto de símbolos, regras de formação de sentenças, etc. [CAMPANI 2009]

Segundo a hierarquia de Chomsky, a linguagem formal é a forma mais simples que permite o desenvolvimento de algoritmos de reconhecimento ou de geração com baixa complexidade, que possuem grande eficiência e fácil implementação.

Linguagens regulares podem ser representadas por um DFA, NFA e por uma Expressão Regular (ER).

2.1. Autômato Finito

Um autômato finito é um modelo matemático (máquina abstrata) de um sistema com entradas e saídas discretas. Pode ser determinístico (DFA) ou não-determinístico (NFA), no qual o não-determinístico significa que mais de uma transição para fora de um estado pode ser possível para o mesmo símbolo de entrada. Tanto os autômatos finitos determinísticos quanto os não determinísticos são capazes de reconhecer precisamente os conjuntos regulares. [MENEZES 2005]

Podemos representar um autômato DFA ou NFA por uma matriz de transição, cujas linhas correspondem a estados, e cujas colunas correspondem aos símbolos de entrada.

2.1.1. Autômato Finito Determinístico (DFA)

Um DFA é um sistema de estado finito, adicionado de 3 elementos conceituais: FITA, UNIDADE DE CONTROLE e FUNÇÃO DE TRANSIÇÃO DE TRANSIÇÃO DE ESTADOS.

- FITA - Dispositivo de entrada que contém a informação a ser processada.
- UNIDADE DE CONTROLE - Reflete o estado corrente da máquina.
- FUNÇÃO DE TRANSIÇÃO DE TRANSIÇÃO DE ESTADOS - Função que comanda as leituras e define o estado da máquina.

2.1.2. Autômato Finito Não Determinístico (NFA)

Um NFA é uma generalização do DFA que permite múltiplas transições sobre um dado por estado x símbolo ou transição em vazio.

O domínio de reconhecimento de um NFA é o mesmo do DFA, ou seja, o conjunto das linguagens regulares.

O processamento de um NFA é semelhante ao processamento de um DFA, apenas com uma diferenciação - um DFA processa em linha, enquanto um NFA processa através de uma árvore de possibilidades, em que se um dos ramos alcançarem um estado final, este será o caminho do processamento. [MENEZES 2005]

2.2. Expressões Regulares - ER

Formalismo algébrico para representação de linguagens regulares. Tem o mesmo poder de representatividade de um DFA ou NFA, podendo inclusive obter um modelo de reconhecimento à partir do outro.

Uma ER é definida recursivamente a partir de conjuntos (linguagens) básicos e operações de concatenação e união. [MENEZES 2005]

Podemos considerar uma Expressão Regular como uma composição de símbolos, caracteres com funções especiais, que, agrupados entre si e com caracteres literais, formam uma seqüência, uma expressão. Essa expressão é interpretada como uma regra, que indicará sucesso se uma entrada de dados qualquer casar com essa regra, ou seja, obedecer exatamente a todas as suas condições. [JARGAS 2009]

Assim uma ER pode ser definida como, um método formal de se especificar um padrão de texto.

2.2.1. Transformação NFA para DFA

Um NFA, ao encontrar o caminho a ser seguido durante uma execução, exige de um hardware qualquer desempenhar atividades de difícil sistematização. Apesar de serem de mais fácil compreensão e com notação menos densa, os NFAs não são apropriados para serem transformados em programas de computadores. Por este motivo, existe uma necessidade de converter um autômato não determinístico em um determinístico. O segredo desta conversão é que cada estado de um NFA corresponde a um conjunto de estados do DFA correspondente. [LEWIS and PAPADIMITRIOU 2001]
Passos para montagem da tabela para conversão:
Gerar a tabela de transformação seguindo os seguintes passos:

1 - Lançar o estado inicial na tabela de transição e avaliar;

2 - Transportar os novos estados obtidos e ainda não avaliados para novas linhas e avaliar;

3 - Repetir o passo anterior até que não haja estados sem avaliações.
Construir o DFA a partir da tabela obtida.
O estado inicial do DFA será o mesmo do estado inicial do NFA.
Os estados finais serão todos os estados que possuírem indicação de estados do NFA.

2.2.2. Gramática

Uma gramática serve para definir qual o subconjunto de sentenças que faz parte de uma determinada linguagem. Ela é um dispositivo formal para especificar uma linguagem potencialmente infinita de uma forma finita.

- Notação Formal de Gramática
  Uma gramática G é definida por uma quádrupla G = (N; T; P; S) onde:
  - N - conjunto finito de não-terminais;
  - T - conjunto finito de terminais;
  - P - conjunto finito de regras de produção;
  - S - símbolo inicial da gramática.

3. Trabalhos Relacionados

Dentre as ferramentas com a finalidade de auxílio no ensino da disciplina de teoria da computação, podemos citar:

- Auger [SZYMANSKI and GARCINDO 2004] - Ferramenta de apoio ao ensino de autômatos Finitos, desenvolvida por Charbel Szymanski e Luiz Alfredo Soares Garcindo. A ferramenta disponibiliza as seguintes funcionalidades:
  - Edição gráfica de um autômato finito.
  - Representação do autômato como uma matriz de transição.
  - Processo de minimização, que pode ser acompanhado pelo usuário passo a passo.
  - Reconhecimento de uma sentença.

  A ferramenta Auger também possui uma ajuda interna, onde o usuário terá um referencial teórico sobre autômatos finitos, e também um manual de auxílio na utilização do software.
- Um Ambiente Simulador para Máquinas de Estados Finitos [PEREIRA et al. 2009] - Criadores: Luciano Machado Pereira, Adriana Postal, Josué Pereira de Castro.

  Implementado em Delphi o simulador possui um ambiente visual para a criação de simuladores de autômatos. O usuário possui total interação com o diagrama criado, podendo manipulá-lo facilmente. O simulador possui um quadro que descreve o caminho percorrido ate chegar em um estado de aceitação ou rejeição, assim ele descreve todos os estados e transições por onde o fluxo da execução passou até ser finalizado.
- Language Emulator [VIEIRA 2002] - Por Luiz Filipe Menezes Vieira, Marcos Augusto Menezes Vieira, Newton José Vieira.

  A ferramenta desenvolvida em Java possui varias funcionalidades, como:

  - Minimizar DFA.
  - Transformar NFA em DFA.
  - Transformar uma gramática regular em um NFA.
  - Transformar NFA em uma gramática regular.
  - Transformar NFA em uma expressão regular.
  - Determinar se uma palavra pertence a uma gramática regular.
  - Transformar uma máquina Mealy em uma máquina de Moore.
  - Transformar uma máquina de Moore em uma máquina Mealy.
  - Dar a saída de uma máquina de Moore para uma entrada dada.
  - Salvar e Carregar DFAs.
- JFLAP (Java Formal Languages and Automata Package) [RODGER 2009]: é uma ferramenta que pode ser usada na disciplina de fundamentos da teoria da computação, mais especificamente, na parte de teoria dos autômatos. Algumas funcionalidades da ferramenta:

  - Criar:
    - DFA, NFA, Gramática Regular e Expressões Regulares;
  - Conversões:
    - NFA para DFA, Minimizar DFA, NFA para Expressão Regular, Expressão Regular para NFA, NFA para Gramática Regular e Gramática;
  - Gera Gramática livre de contexto.

- EduLing [DOGNINI 2003] - ferramenta desenvolvida em Object Pascal, voltada para o auxílio e aprendizagem de linguagens regulares. O EduLing apresenta funções como: criação e validação de uma ER e de um DFA, transformação de ER para um DFA ou NFA e a de um DFA para uma ER. A ferramenta apresenta as seguintes características:

  - Interface gráfica interativa para criação e manipulação de autômatos;
  - Tutorial sobre a disciplina que auxilia o aluno na compreensão da mesma;
  - Permite salvar e carregar os autômatos criados;
  - Permite a visualização passo a passo da validação de um DFA;
  - Apresenta limitações na parte de expressões regulares e autômatos;
  - Não permite a criação e validação de um NFA.

- SisLR (Sistema de Linguagens Regulares) [ASSIS 2007] - desenvolvido para plataforma Windows usando o paradigma de programação orientada a objetos da linguagem Object Pascal do Delphi. Software educativo voltado para o auxílio no ensino das linguagens regulares. O SisLR permite ao aluno a criação e validação dos autômatos através de uma interface gráfica interativa e de fácil usabilidade. Abaixo descritas as principais características do SisLR:

  - Interface gráfica para criação e manipulação de autômatos;
  - Opera com NFA, DFA e ER;
  - Apresenta tutorial da Disciplina;
  - Ajuda do Sistema;
  - Salvar Autômatos.

4. A Ferramenta

A ferramenta denominada Conversor e Validador de Sequências de Autômatos, utiliza na sua implementação a linguagem de programação Java, linguagem orientada a objetos desenvolvida na década de 90 pela Sun Microsystem, foi desenvolvido no IDE(ambiente de desenvolvimento integrado) NetBeans, projeto Open Source fundado pela Sun Microsystem no ano de 2000.

A interface dá ferramenta pode ser visualizada na Figura 1 apresentada abaixo.

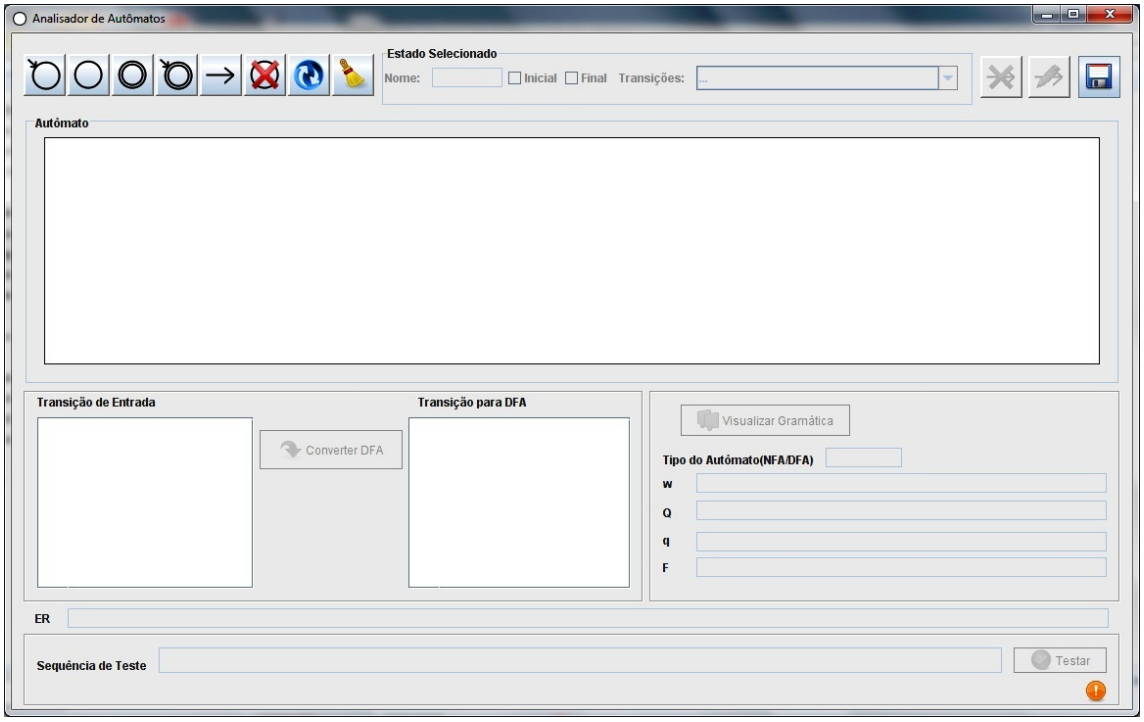

Figura 1. Tela Principal.

Na ferramenta se encontra todos os componentes necessários para a criação e manipulação do autômato.

Para isto o software conta com uma barra de botões onde são encontrados um botão para cada tipo de estado possível: Inicial, Comum, Final e Inicial Final. Também possui o botão para adicionar a transição entre os estados, para está ação se deve selecionar o estado de origem no painel, clicar no botão de transição e selecionar o estado de destino, então forneça o alfabeto q esta transição ira conter na popup que será aberta.

Os botões Excluir Estado, Atualizar e Limpar também compõem a barra de botões. O botão Excluir Estado remove o estado selecionado e todas as transições ligadas a ele. O botão Atualizar verifica se o autômato é valido e executa as funcionalidades de gerar a Matriz de Transição, gerar a ER e preenche em tela todas as informações do autômato. O botão Limpar, remove tudo que foi inserido na tela.

Um estado selecionado no painel tem suas informações mostradas em tela, onde é possível alterar suas características e ainda editar ou excluir suas transições.

A partir do autômato montado e clicando no botão Atualizar na barra de botões é feita uma validação, verificando se o autômato possui um estado inicial e pelo menos um estado final. Passando está validação a ferramenta analisa se o autômato montado é um DFA ou um NFA, gera-se então a Matriz de Transição, a ER e para autômatos NFA permite a conversão da Matriz de Transição para DFA. É possível também exibir a gramática do autômato através do botão Visualizar Gramática.

Depois de criado o autômato é possível testar sequências de entrada pelo campo Sequencia de Teste, sequências estás compostas por símbolos/caracteres que estejam presentes nas transições do autômato. Para que uma sequência seja dada como valida é anali-

sado cada símbolo, começando a validação pelo estado inicial e ao termino dos símbolos deve-se estar em um estado final para que a sequência seja aceita.

* Abaixo é apresentado um exemplo da criação de um autômato pela ferramenta, e demonstrado por partes, todas as funcionalidades presentes no software.

As Figuras 2 e 3 exibe a área de Edição Gráfica da ferramenta, onde é possível o usuário manipular um autômato a partir de componentes predefinidos, permitindo assim uma maior interação do usuário com a ferramenta.

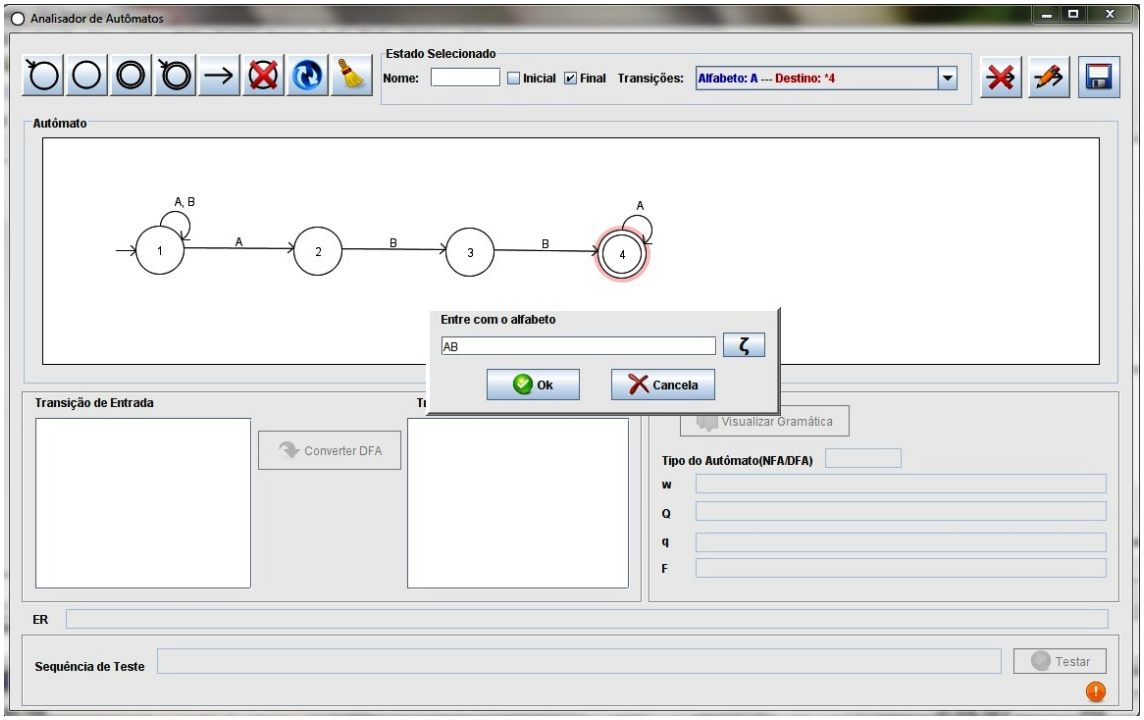

Figura 2. Edição Gráfica. Adicionando alfabeto a transição.

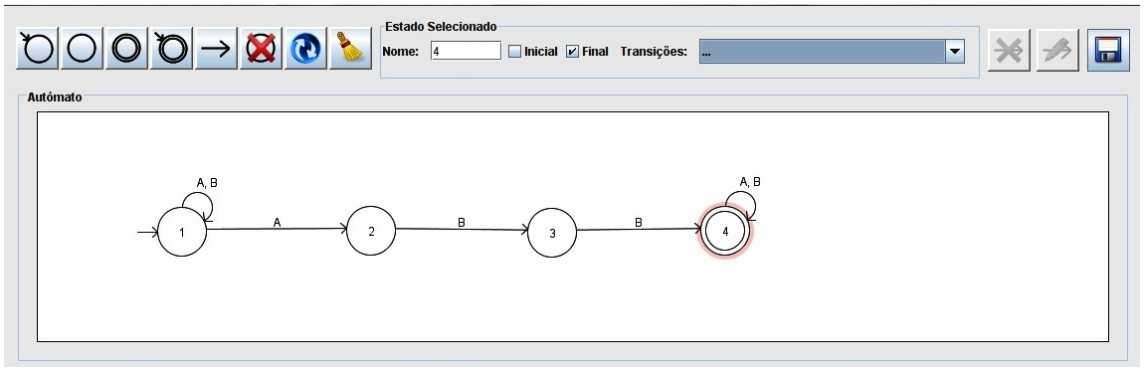

Figura 3. Edição Gráfica.

A Figura 4 exibe a área de Transição, onde é montada a matriz de transição do autômato criado exibida na área ”Transição de Entrada”e sendo o autômato um NFA permite fazer a conversão para DFA da matriz de transição, como pode ser vista na área ”Transição para DFA”.

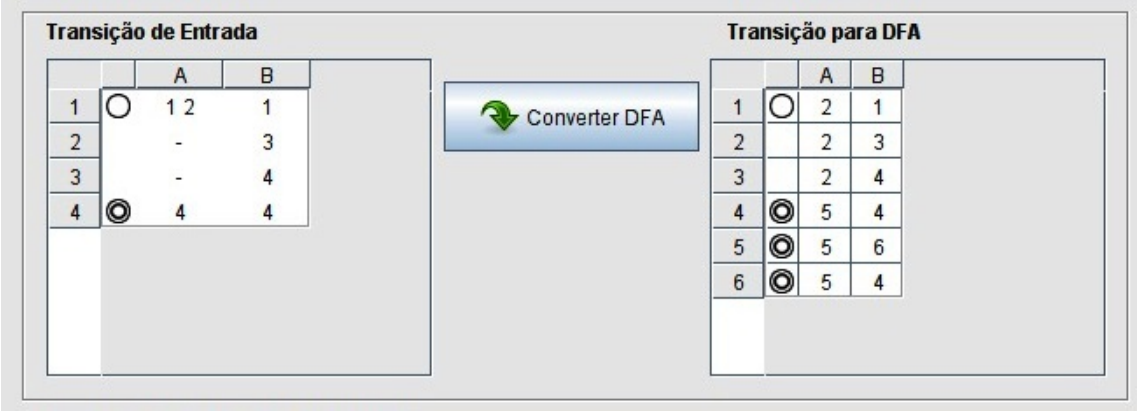

Figura 4. Área de Transição.

A Figura 5 demonstra as informações contidas no autômato criado, como:

- o tipo do autômato (DFA ou NFA);
- as suas propriedades:
  - W = Alfabeto
  - Q = Estados
  - q = Estado Inicial
  - F = Estados Finais
- e permite também gerar a gramática através do botão ”Visualizar Gramática”.

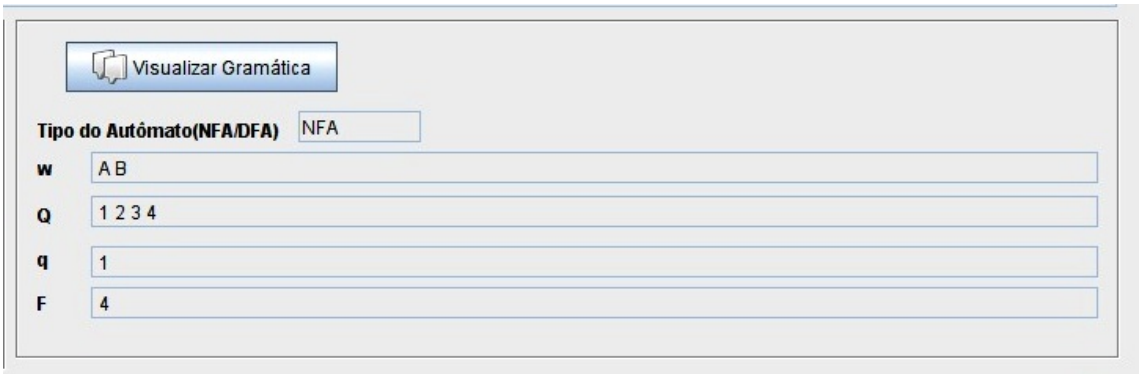

Figura 5. Informações do Autômato.

A Figura 6 exibe a área onde é gerada a Expressão Regular referente ao autômato criado e a área de sequência de testes aonde o usuário fornece uma entrada de caracteres validos a ser processado e validado no autômato.

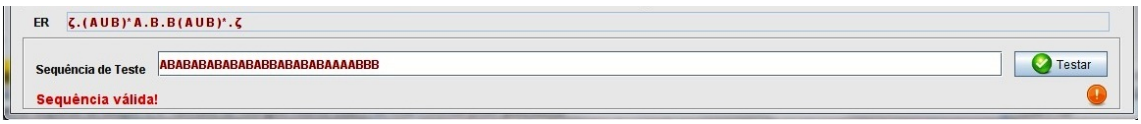

Figura 6. Expressão Regular e Sequência de Teste.

Na Figura 7 é mostrado a imagem que é gerada e salva pela ferramenta a partir do autômato criado.

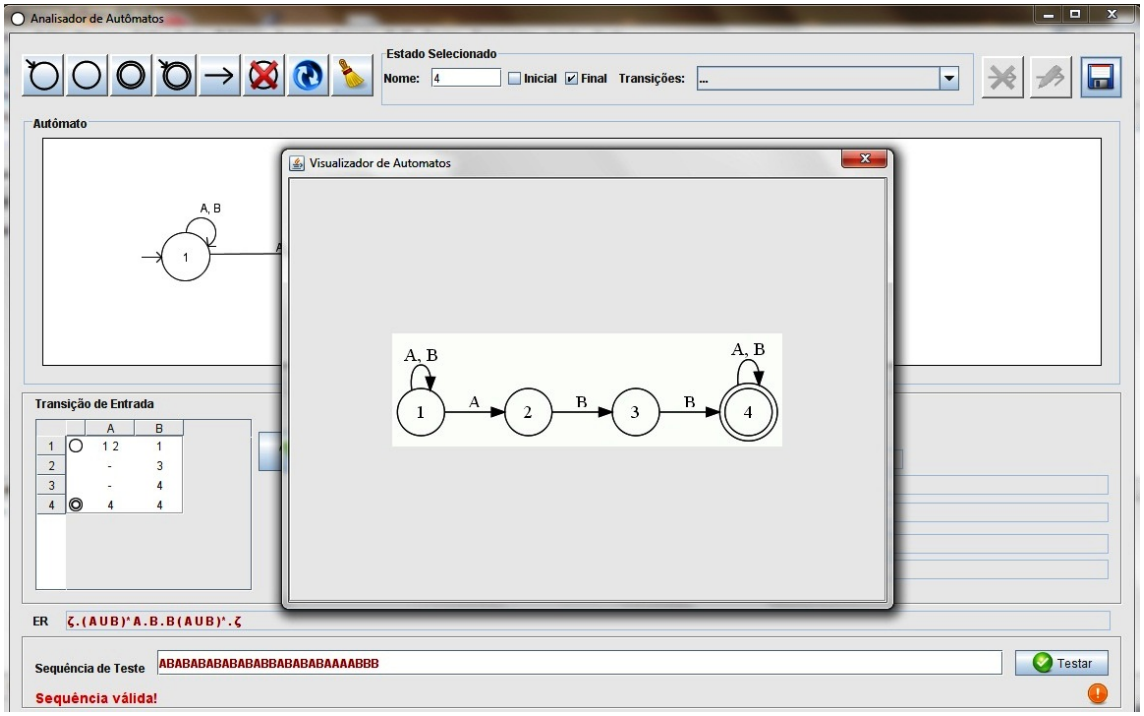

Figura 7. Imagem Salva.

E por fim, como é possível ver na Figura 8 é exibido a gramática gerada pela ferramenta a partir do autômato criado pelo usuário.

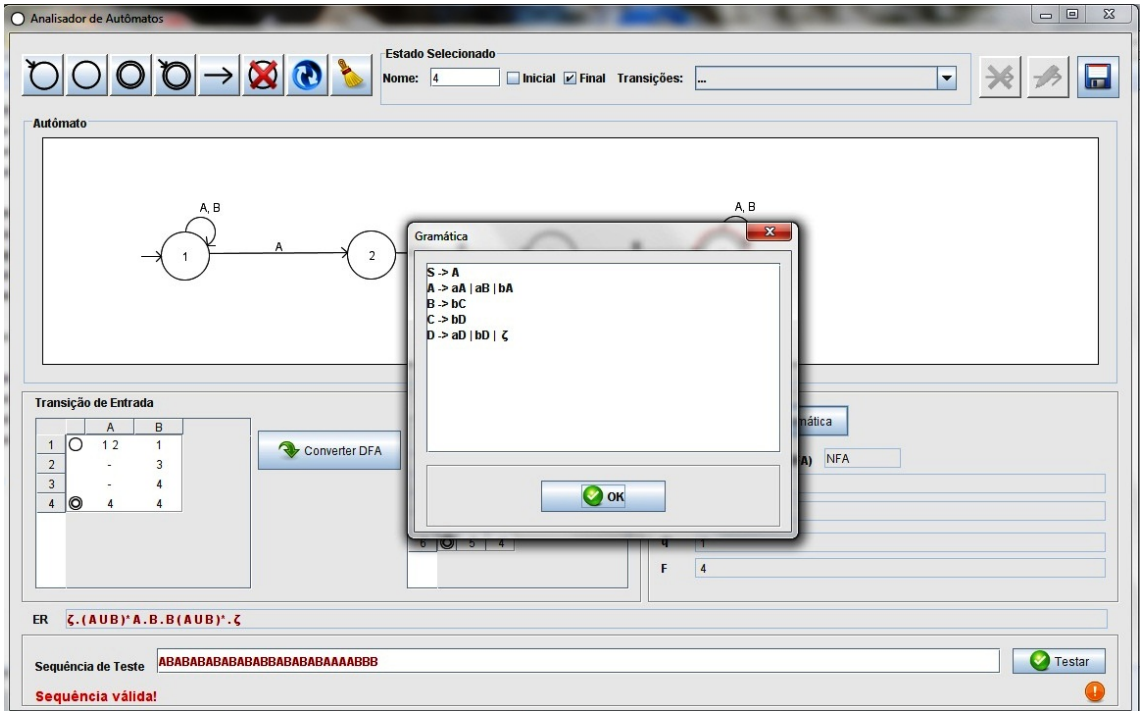

Figura 8. Gramatica.

5. Conclusão e Trabalhos Futuros

Neste trabalho foi possível concluir que a ferramenta aperfeiçoada, Conversor e Validador de Sequências de Autômatos, possui facíl entendimento e uma melhor interface entre usuário e o software, diferente das ferramentas apresentadas na seção 2 deste artigo, que possuem um maior grau de abstração para quem está começando os estudos sobre autômatos, e pelo fato de ser desenvolvido em um ambiente multiplataforma como Java, permite uma maior distribuição do software.

Para a realização de trabalhos futuros, ficam as seguintes sugestões:

- Implementção da normalização da gramática;
- Inserção de um menu contendo as funcionalidades da ferramenta;
- Implementação da conversão de NFA para DFA graficamente;
- Permitir aalvar o trabalho realizado de forma a não perder o que foi processado.

Referências

ASSIS, J. R. (2007). Sislr - sistema de linguagens regulares. *UNIPAC, Barbacena - MG*.

CAMPANI, C. A. P. (2009). Disciplina de linguagens formais. *Disponível em http://minerva.ufpel.edu.br/ campani/lingformal.pdf. Data de acesso: 11/07/2011*.

DIVERIO, T. A. and MENEZES, P. B. (2000). *Teoria da Computação: maquinas universais e computabilidade*. Porto Alegre: Sagra Luzzato, 2º edition.

DOGNINI, J. M. (2003). Eduling - software educacional para linguagens regulares. *Univali, Itajaí - SC*.

JARGAS, A. M. (2009). *Expressões Regulares - Uma abordagem divertida*. São Paulo: Novatec, 3º edition.

LEWIS, H. and PAPADIMITRIOU, C. (2001). *Elementos de Teoria da Computação*. Porto Alegre: Bookman, 2º edition.

MENEZES, P. B. (2005). *Linguagens Formais e Autômatos*. Porto Alegre: Sagra Luzzato, 5º edition.

PEREIRA, L. M., POSTAL, A., and Pereira, C. J. (2009). Um ambiente simulador para máquinas de estados finitos. *CCET, Cascavel - PR*.

RODGER, S. H. (2009). Jflap - java formal languages and automata package. *Disponível em http://www.cs.duke.edu/csed/jflap. Data de acesso: 15/07/2011*.

SANDY, J. M. d. S. (2010). Conversor e validador de sequências de autômatos. *UNIPAC, Barbacena - MG*.

SZYMANSKI, C. and GARCINDO, L. A. S. (2004). Auger - ferramenta no apoio ao ensino de autômatos finitos. *UNISUL, Santa Catarina - SC*.

VIEIRA, Luiz Felipe Menezes; VIEIRA, M. A. M. V. N. J. (2002). Language emulador, uma ferramenta de auxílio no ensino de teoria da computação. *UFMG, Belo Horizonte - MG*.