Criação de uma aplicação para simulação de autômatos finitos

Filipe Andrade Peres[1], Lucas Elias de Andrade Cruvinel[2], Ramon Soares Mendes de Meneses Leite[3]

[1]Instituto de Biotecnologia – Departamento de Ciência da Computação
Universidade Federal de Catalão (UFCat)
CEP: 75704-020 - Av. Dr. Lamartine Pinto de Avelar, 1120 - Catalão - GO - Brazil

{filipeperes,lucascruvinel,ramonsoares}@discente.ufcat.edu.br

Abstract. *Computer-aided software simulation tools have spread at all levels of education. They are recognized as tools for the effective study of complex and abstract systems. By traditional learning of automaton theory, students cannot easily visualize theoretical constructions, so in this case, simulation systems represent an excellent combination of theoretical and practical experience. This article presents a tool developed with the objective of using an itative technique to simplify and visually present to students, abstract concepts and mathematical models of automaton theory, allowing from chains of inputs provided by the user to perform syntactic tests.*

Resumo. *Ferramentas de simulação de software assistidas por computador se espalharam em todos os níveis de ensino. Eles são reconhecidos como ferramentas para o estudo eficaz de sistemas complexos e abstratos. Pela aprendizagem tradicional da teoria dos autômatos, os alunos não conseguem visualizar facilmente construções teóricas, portanto, neste caso, os sistemas de simulação representam uma excelente combinação de experiência teórica e prática. Este artigo apresenta uma ferramenta desenvolvida com o objetivo de utilizar uma técnica iterativa para simplificar e apresentar visualmente aos alunos, conceitos abstratos e modelos matemáticos da teoria dos autômatos, permitindo a partir de cadeias de entradas fornecidas pelo usuário, realizar testes sintáticos.*

1. Introdução

A teoria da linguagem formal se baseia em modelos abstratos de sistemas de computador chamados ”autômatos”. Ela foi desenvolvida na década de 1950 buscando desenvolver teorias referentes às linguagens naturais. Realizado esse estudo, foi constatado que essa hipótese fornecia fundamentos teóricos primordiais para o estudo das linguagens artificiais (linguagens regulares, livres de contexto), essencialmente quando se trata das linguagens oriundas da computação [Menezes 2005].

Os autômatos finitos, consiste de uma lógica matemática em que as operações são realizadas em cada estado dos eventos, e estes eventos podem ser de um único estado ou vários. Por ter ações com estados de dois ou mais eventos, os autômatos finitos tendem a mudar de determinístico para não determinístico.

Os autômatos são, na verdade, abstrações matemáticas cujo estudo se baseia principalmente na imaginação de conceitos teóricos. A razão para isso é a complexidade

da realização de aulas práticas ou a falta de ferramentas educacionais adequadas. Estudantes da área de Ciência da Computação encontram o conceito de autômato por meio de currículos de várias disciplinas diferentes. Por exemplo, a Teoria das Linguagens Formais e Autômatas e Teoria da Computação.

Tendo em vista a importância do estudo da Teoria da Computação, para facilitar o entendimento acerca destes conceitos formais, que além de ser abstrato, possui um forte viés matemático, acredita-se que seja extremamente oportuno fornecer uma solução via software que possibilite uma melhor compreensão sobre autômatos finitos. A utilização de ferramentas computacionais torna o aprendizado mais fácil, trazendo entretenimento se o conteúdo for devidamente apresentado. De acordo com [Oliveira 2004], caso as ferramentas computacionais sejam usadas como apoio ao processo de ensino, a construção crítica do conhecimento será ampliada. Elas podem favorecer, no ambiente de sala de aula e para além dele, o processo da experimentação.

Este artigo descreve um sistema de software para visualização e simulação de máquinas de estados finitos, possibilitando criar e validar autômatos, isso através de uma interface gráfica que facilite a usabilidade.

O artigo está sistematizado em seis seções. A segunda seção apresenta a justificativa pelo qual o trabalho foi realizado. O terceiro capítulo descreve os objetivos principais do trabalho, provendo o ponto principal que se busca atingir através da implementação dessa ferramenta. A metodologia utilizada para desenvolvimento do trabalho do sistema de simulação é apresentada no quarto capítulo. Para demonstrar o funcionamento, os experimentos e resultados são descritos na quinta seção. Na sexta seção, é feita uma conclusão e são apresentados os planos para futuros aprimoramentos da ferramenta.

2. Justificativa

A disciplina de Teoria da Computação é de suma importância na formação dos graduados em Ciência da Computação, pois fornece fundamentos teóricos da disciplina que permite um melhor entendimento da computação e suas origens históricas e matemáticas, podendo explorar seus problemas e possibilidades.

A utilização de softwares de ensino pode ser útil para o ensino e aprendizagem de uma disciplina e auxiliar na busca pela excelência e melhoria da qualidade educacional, e partindo desse ponto, este trabalho descreve a concepção e desenvolvimento de uma ferramenta criada como uma opção de auxílio ao aprendizado de autômatos finitos na Teoria da Computação. A hipótese de partida é que a qualidade do ensino na disciplina poderia ser melhor aplicada utilizando ferramentas de software, dado que nesta disciplina os exercícios são de suma importância para a assimilação de conhecimentos teóricos e estes podem ser resolvidos seguindo uma série de etapas perfeitamente definidas.

É fato que a assimilação e aprendizagem dos conhecimentos e conceitos da matéria de forma sólida requerem a realização de um grande número de exercícios por parte do aluno.Entretanto, na maioria dos casos, essa possibilidade é inviável de ser realizada de forma intensiva nas aulas, em razão do conteúdo extenso e ao tempo limitado para ministrar as aulas.

Por conseguinte, torna-se cada vez mais necessário o estabelecimento de medidas que visem minimizar esse problema. Inicialmente, um plano que poderia ser aplicado

seria disponibilizar o material do assunto ao aluno antes da abordagem do assunto em aula, podendo os mesmo terem tempo de dedicar à resolução de exercícios. Entretanto, por mais que esse procedimento tenha uma perspectiva positiva do ponto de vista da aprendizagem, a desvantagem em realizar este processo está no fato de que os conceitos a serem compreendidos muitas vezes são abstratos, o que pode dificultar o processo de aprendizado sem o auxílio e explicação do docente.

Como alternativa à proposta anterior, este trabalho sugere a concepção e desenvolvimento de uma aplicação que concede aos alunos a alternativa de sugerir exercícios relacionados ao conteúdo de autômatos finitos e gerar soluções para os mesmos, isso dentro do escopo do que a aplicação foi proposta a realizar. A partir disso, é possível verificar os resultados, assim determinando se coincidem com a resolução do aluno. Isso pode gerar motivação no estudo da matéria, abrindo um leque de possibilidades para criação de exercícios próprios, além de potencializar suas habilidades criativas e analíticas.

A finalidade do desenvolvimento da aplicação que será descrita nesse é incorporar características que buscam fornecer ambientes fáceis de usar, utilizando do estímulo gráfico e visual. Dessa forma, o motivo da criação é auxiliar o aprendizado do conteúdo relacionado a Linguagens Formais e Autômatos.

3. Objetivo

A pretensão na criação desta aplicação é atender dois tipos de usuários em especifico: alunos e professores. Esses dois tipos tem necessidade e prioridades comumente diferentes, logo, deve ser levado em consideração que para realização satisfatória do projeto, tal deve atender ambos.

A principal funcionalidade a ser oferecida na ferramenta, levando em conta o ponto de vista do professor e do aluno, é a possibilidade de aceitar exercícios e calcular a solução dos mesmos, demonstrando de forma clara toda etapa realizada pelo algoritmo, concedendo em detalhes como a solução foi atingida. Levando em conta que a teoria dos autômatos outorga análise teóricas fundamentado em abstrações de computador, permitindo ter o vislumbre dos limites computacionais do modelo através de provas matemáticas, o projeto tem como propósito apresentar um algoritmo capaz de identificar autômatos finitos determinísticos e autômatos finitos não-determinísticos.

Um ponto chave da aplicação, além de apenas reconhecer os dois tipos de autômatos, se fundamenta também no princípio de relação de equivalência entres os autômatos finitos determinísticos e autômatos finitos não-determinísticos, no qual ocorre em virtude de identificarem a mesma classe de linguagem. A prova que serve como base para essa afirmação é fornecida pelo teorema que diz que para todo autômato finito não-determinístico, há um autômato finito determinístico equivalente.

Logo, levando em conta os aspectos mencionados acima, o objetivo principal deste trabalho é fornecer a capacidade de construir e testar autômatos em um ambiente intuitivo. É fornecido um algoritmo para validar o funcionamento de autômatos finitos determinísticos (DFA) e não-determinísticos (NFA) , no qual partindo da disposição da equivalência entre ambos, a aplicação dispõem de uma funcionalidade adicional, a transformação de um autômatos finito não-determinístico para um autômatos finitos determinístico.

4. Metodologia

Uma máquina de estados finitos é um modelo abstrato contendo um número finito de estados de algo. É utilizado para representar e controlar o fluxo de execução de quaisquer comandos. Uma máquina de estados finitos é um modelo de computação baseado em uma máquina de estados hipotéticos. Apenas um estado pode estar ativo por vez. Portanto, para realizar qualquer ação, a máquina deve alterar seu estado. São comumente usadas para organizar e representar o fluxo de execução de algo.

Autômatos finitos ou máquina de estados finitos, consiste em um conjunto finito de estados internos e um conjunto de regras de controle, essas regras são usadas para controlar para qual estado deve ser girado após a leitura do símbolo de entrada no estado atual [Menezes 2005].

Autômato finito é um modelo matemático que pode ser usado para descrever o processo de reconhecimento de cadeias de símbolos de entrada. É constituído por um alfabeto, um conjunto de um conjunto de estados finitos, uma função de transição, um estado inicial e um conjunto de estados finais. Seu funcionamento é baseado em uma função de transição , que recebe de um estado inicial uma string de caracteres pertencentes ao alfabeto (a entrada), e que lê essa string conforme o autômato se move de um estado para outro, para finalmente parar em um final ou estado de aceitação , que representa a saída [Sipser 2005].

Nesta máquina, seu estado está sempre em um determinado estado do estado finito, o estado atual do sistema resume as informações históricas, que são indispensáveis para o estado do sistema que pode ser determinado por entradas subsequentes. Simplificando, significa que o próximo estado pode ser determinado com base no estado atual do sistema e no próximo símbolo de entrada. Por exemplo, o mecanismo de controle do elevador é um exemplo de máquina automática finita. Os requisitos de serviço do cliente (ou seja, o andar a ser alcançado) são as informações de entrada do dispositivo, o número de andares e a direção de movimento do elevador indicam o estado do dispositivo. Este mecanismo não lembra todos os requisitos de serviço anteriores, mas lembra apenas em qual andar, a direção do movimento (para cima ou para baixo) e os requisitos de serviço que não foram atendidos.

Os computadores podem ser considerados um sistema de estados finitos. Embora o número de estados possíveis seja grande, ele ainda é limitado. A teoria dos autômatos finitos é uma ferramenta eficaz para projetar esses sistemas.

Os autômatos finitos têm uma relação muito próxima com a gramática formal e as expressões formais. Suas capacidades descritivas são as mesmas. Portanto, os autômatos finitos são uma ferramenta muito útil para identificar formas formais. Eles podem ser determinísticos (DFA) ou não-determinísticos (NFA), no qual este último significa que a transição de um estado pode ter vários destinos. Ambos são aptos a reconhecer conjuntos regulares [Menezes 2005].

4.1. Definição de um Autômato Finito Determinístico (AFD)

Um Autômato Finito Determinístico (AFD) é definido como uma quíntupla (Σ, S, δ, S0, F), onde [Vieira 2004]:

- Σ é um alfabeto de símbolos de entrada;

• S é o conjunto finito e não vazio de estados;

• δ é a função de transição, da forma δ: S $\times$ $\Sigma \rightarrow$ S;

• S0 é o estado inicial, S0 $\in$ S;

• F é o conjunto de estados finais, F $\subseteq$ S.

O funcionamento consiste na admissão de uma cadeia de caracteres de entrada, no qual o autômato vai estar em seu estado inicial, e a proporção que cada símbolo na cadeia de caracteres é processada, o estado é modificado em concordância com a função de transição. Assim que o último dos símbolos na cadeia de caracteres foi processado, o autômato para em seu estado final do processo. Se caso esse estado final for um estado de aceitação, então a cadeia de caracteres faz parte da linguagem reconhecida pelo autômato, do contrário, não pertence ao idioma.

4.2. Definição de um Autômato Finito Não-Determinístico (AFND)

Um AFND pode ser representado em diagrama por um gráfico direcionado rotulado, denominado gráfico de transições, em que os nós são os estados e as arestas rotuladas representam a função de transição. Este gráfico parece um diagrama de transição, mas o mesmo caractere pode rotular duas ou mais transições de um estado.

Este modelo se assemelha com um AFD, porém a divergência está no fato que uma AFND em pelo menos um estado q $\in$ Q, tal que para um símbolo a $\in \Sigma$ do alfabeto, há mais de uma transição δ (q, a) possível. A passo que um AFD realiza o processamento em linha, um AFND faz isso por meio de uma árvore de possibilidades, no qual se um dos ramos chegar ao estado final, será definido a partir disso o caminho de processamento [Menezes 2005].

A diferença desse modelo da 5-tupla em relação a um AFD está na função de transição. Enquanto em um AFD a função se caracteriza por δ: S $\times$ $\Sigma \rightarrow$ S, em uma AFND é definida como δ: S $\times$ $\Sigma \rightarrow$ p(S).

4.3. Equivalências entre autômatos finitos

Dois autômatos finitos são considerados equivalentes, se ambos reconhecerem a mesma linguagem regular. Existe a possibilidade de transformar um AFND (QN, Σ, q0, δN, FN) em um AFD equivalente (QD, Σ, q0, δD, FD), deixando o alfabeto o alfabeto Σ e o estado inicial q0 originais. Para converter, inicialmente é feita uma transição para um AFD intermediário com estados e transições redundantes, que devido ao fato de não serem acessíveis desde o estado inicial, são descartados para se atingir o AFD final. Para determinar o AFD intermediário, é preciso seguir os seguintes passos:

Inicialmente, o conjunto de estados é redefinido QN = {q0, q1, ..., qm} original, tal como uma que consiste de todos os subconjuntos de QN. Os novos estados finais vão consistir naqueles estados que contêm qualquer um dos estados finais originais. Em sequência, o conjunto original de transições é rearranjado, por transições do tipo ΣD (S, a), no qual a $\in \Sigma$, e S é a união de todos os estados q de QN para os quais a transição δ N (q, a). No final, são retirados os estados inacessíveis ou inalcançáveis (junto com suas transições de saída), ou seja, aqueles estados que não podem ser acessados a partir do estado inicial. Após essa depuração, o AFD final é alcançado.

4.4. Criação da aplicação

Para implementação da simulação dos autômatos finitos foi utilizado a linguagem de programação Python. O sistema foi projetado usando a biblioteca ”Tkinter”, nativa do Python para desenvolver uma interface gráfica para o usuário e a ”igraph”para a criação de grafos com o intuito de mostrar os autômatos em um formato mais amigável.

A estrutura do software para que a simulação ocorra de forma consistente, é necessário fornecer à interface de entrada / saída do usuário, componentes para inserir expressões regulares. Isso é feito para máquinas de estados finitos determinísticos e não-determinísticos.

Para a execução de um autômato finito determinístico, é preciso primeiro fazer a validação do autômato na execução do algoritmo, então é feita uma análise dos valores de entrada para ver se está tudo nos conformes. Tais análises consistem em:

• Verificar se o conjunto de estados é vazio;

• Verificar se os estados finais estão no conjunto de estados atingíveis;

• Verificar se cada estado inicial de cada transição está no conjunto de estados atingíveis;

• Verificar se os caminhos passados nas regras de transição estão no alfabeto do autômato;

• Verificar se cada estado alvo de cada transição está no conjunto de estados atingíveis.

Se os dados do autômato estiverem inválidos por algum desses motivos, é retornado “False”, caso contrário, é retornado “True” e a execução continua.

Após validar o autômato, vamos lê-lo, então começamos pelo estado inicial e iteramos sobre cada caractere da palavra. Dentro desse laço, temos outro que itera sobre todas as regras de transição para verificar para qual estado ele irá passar. Caso ele não alcance nenhuma transição, então ele não consegue chegar a um estado de aceitação e a palavra não é aceita e é considerada inválida, porém se chegar ao final da execução com todos os caracteres tendo uma transição e o último estado sendo um estado final, a palavra é aceita e é considerada válida.

Para executar a funcionalidade de simulação de autômatos finitos não-determinísticos, primeiramente é preciso convertê-lo para um AFD e, após isso, lê-lo normalmente após fazer a validação do autômato. Para a conversão, transformamos todas as regras de transição em um dicionário aninhado, onde o mais externo contém as chaves referentes ao estado inicial daquela transição, os dicionários internos contêm os caminhos como chave e os estados alvo como valores.

Após essa conversão para dicionário, podemos converter o AFN para AFD. A conversão se dá utilizando o dicionário como uma tabela de transição e fazendo as operações passo a passo através dela. Esse processo será demonstrado em detalhes na seção 5.

Depois da conversão, o dicionário é convertido novamente para listas aninhadas (formato padrão para a execução), os estados são substituídos pelos estados novos e as regras de transição são substituídas pelas novas regras de transição, assim, já é possível

executar o AFD.

5. Experimentos e Resultados

Na interface gráfica, a partir dos campos reservados de entrada, foram fornecidos exemplos para definir o diagrama de transições. Esse processo foi realizado fornecendo para os dados de entrada a 5-tupla para simulação dos autômatos finitos.

Caso essa cadeia de entrada seja fornecida para simulação de um autômato finito determinístico, o algoritmo apenas determina se a cadeia de entrada aceita ou não, percorrendo passo a passo as regras de transição.

Então inicialemnte é fornecido os dados de entrada para a aplicação, consistindo da 5-tupla como demonstrado na Figura 1.

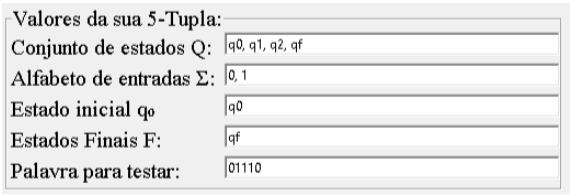

Figura 1. Dados do AFD e palavra a ser analisada

Para a visualização gráfica do algoritmo, é inserida as regras de transição como expresso na Figura 2.

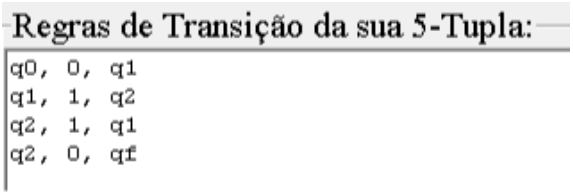

Figura 2. Regras de transição do AFD

As próximas imagens denotam detalhadamente as etapas de transição realizada pela aplicação, facilitando a visualização da construção teórica do autômato finito determinístico.

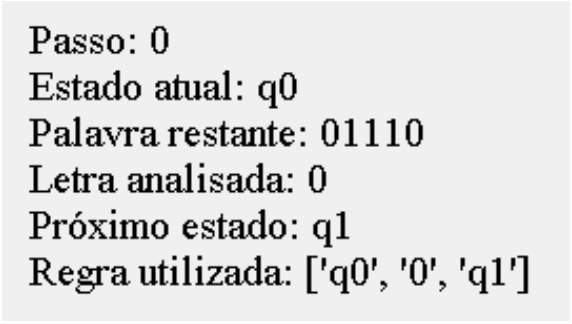

Figura 3. Informações do passo a passo da leitura da palavra

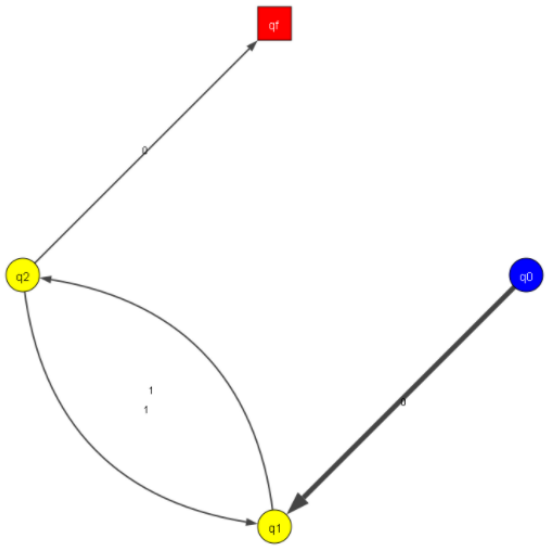

Figura 4. Grafo gerado para o Passo 0

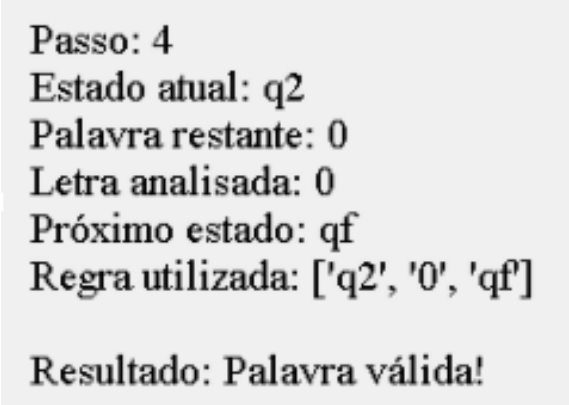

Figura 5. Passo final mostrando que a palavra é válida

Um autômato finito não-determinístico é equivalente a autômato finito determinístico, logo pode haver uma conversão do mesmo para um autômato finito determinístico, essa etapa pode ser constatada na Figuras 6, 7, 8 e 9.

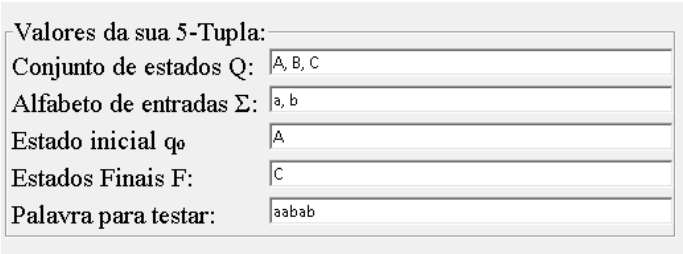

Figura 6. Dados do AFN e palavra a ser analisada

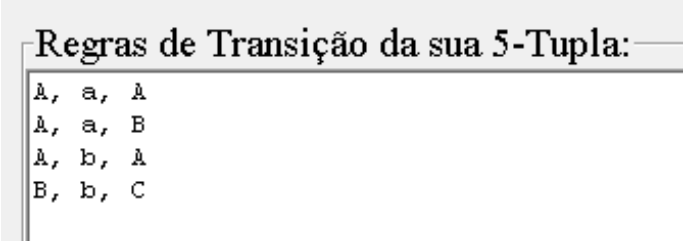

Figura 7. Regras de transição do AFN

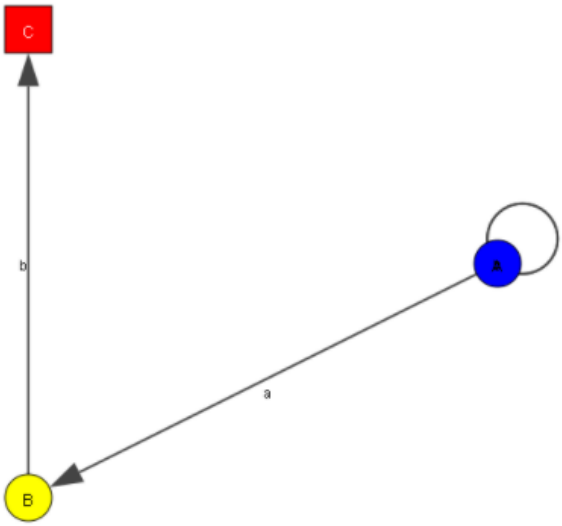

Figura 8. Grafo gerado para o Passo 0, ainda em AFN

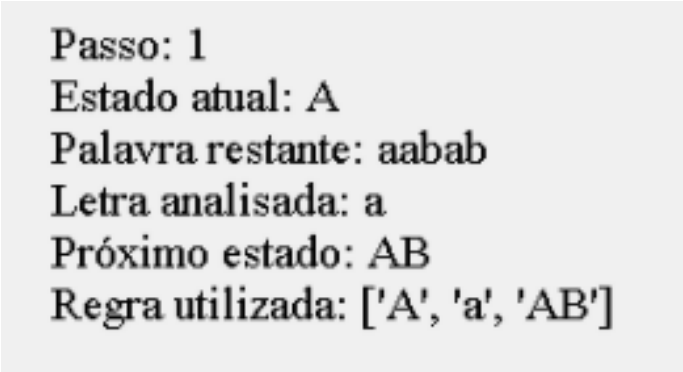

Figura 9. Passo 1 após a conversão para AFD

Como resultado principal deste projeto, tem-se que a aplicação desenvolvida aceita o reconhecimento de palavras da classe das Linguagens Regulares para simular os autômatos finitos detrminísticos e não-determiníticos. Quando a simulação da operação do autômato é iniciada, o sistema para todos os símbolos da cadeia de entrada mostra o procedimento de análise passo a passo, onde o estado atual do autômato é representado. Foi constatado que a aplicação entregou de forma consistente os resultados, dados os exemplos que foram fornecidos como entrada.

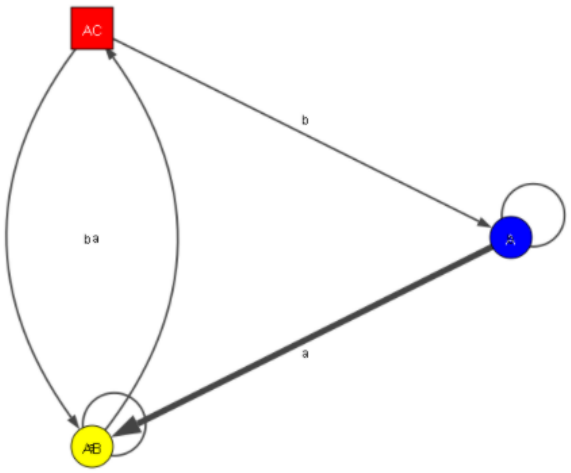

Figura 10. Grafo referente ao Passo 1

```
Passo: 5
Estado atual: AB
Palavra restante: b
Letra analisada: b
Próximo estado: AC
Regra utilizada: ['AB', 'b', 'AC']

Resultado: Palavra válida!
```

Figura 11. Passo final mostrando que a palavra é válida

6. Conclusões

Neste artigo, é apresentado um sistema de simulação para o aprendizado de autômatos finitos. Esta ferramenta foi desenvolvida com o objetivo de auxiliar os alunos no processo de aprendizagem e compreensão da teoria dos autômatos, permitindo a visualização de construções teóricas através de uma interface gráfica interativa.

Após a construção do autômato, é possível executar a simulação da operação para uma cadeia de caracteres de entrada arbitrária. O usuário recebe feedback sobre cada etapa da simulação na forma gráfica e textual. Além de simular o funcionamento de autômatos finitos, as funções adicionais do sistema são: conversão de expressões regulares em AFD, bem como em AFND, havendo a possibilidade de conversão de AFND baseada no princípio de equivalência entre ambos.

A ferramenta desenvolvida auxilia no processo de aprendizagem, pois simula a execução de autômatos finitos, dessa forma permite de maneira consistente experimentação prática das máquinas abstratas abordadas na Teoria da Computação. Possibilita também a visualização de construções teóricas através de uma interface gráfica.

O sistema de simulação ainda está em fase de desenvolvimento e pode ser significativamente melhorado. As próximas etapas em termos de melhoria do sistema de simulação estão no sentido de expandir sua funcionalidade com tipos adicionais de autômatos. Além disso, está planejando expandir a funcionalidade do sistema para incluir áreas adicionais da teoria da linguagem formal.

Referências

Menezes, P. B. (2005). *Linguagens Formais e Autômatos*. UFRGS, 5th edition.

Oliveira, G. P. (2004). Construção coletiva do conhecimento através de uma experiência de incentivo à autonomia dos estudantes no aprendizado de matemática discreta. *In Anais do Encontro Paulista de Educação Matemática, São Paulo*.

Sipser, M. (2005). *Uma Introdução à Teoria dos Computação*. CENGAGE.

Vieira, J. N. (2004). *Linguagens e Máquinas: Uma Introdução aos Fundamentos da Computação*. Departamento de Ciências Computação, UFMG.