Avaliação do JFLAP para ensino de Autômatos

Lucas Pereira Cassanho[1], João Felipe Pavret Michels[1], Cinthyan Renata Sachs Camerlengo de Barbosa[1]

[1]Programa de Pós-Graduação em Ciência da Computação – Universidade Estadual de Londrina (UEL)
Caixa Postal 10011 – 86057-970 – Londrina – PR – Brazil

lucascassanho@gmail.com,{pavret.michels,cinthyan}@uel.br

Abstract. *Educational software has emerged as an auxiliary tool in the teaching and learning process of Formal Languages and Automata (FLA) in Computer Science courses. However, few studies have been published analyzing the tool JFLAP in this context. This paper addresses the classroom experience with JFLAP, specifically focusing on recognizer machines content. We found that, although JFLAP interface is somewhat unfriendly, the tool proves to be effective for teaching recognizer machines, significantly contributing to the understanding and application of FLA theoretical concepts.*

Resumo. *Softwares educativos têm surgido como uma ferramenta auxiliar no processo de ensino e aprendizagem dos conteúdos da disciplina Linguagens Formais e Autômatos (LFA) no curso de Ciência de Computação. Porém, poucos trabalhos têm sido publicados analisando a ferramenta JFLAP nesse processo. Este trabalho aborda a experiência em sala de aula com o JFLAP especificamente com os conteúdos de máquinas reconhecedoras. Verificamos que embora a interface do JFLAP seja pouco amigável, a ferramenta se revela eficaz para o ensino de máquinas reconhecedoras, contribuindo de forma significativa para a compreensão e aplicação dos conceitos teóricos de LFA.*

1. Introdução

O estudo de Linguagens Formais e Autômatos (LFA) é essencial na Ciência da Computação, oferecendo aos estudantes um entendimento aprofundado dos princípios teóricos que fundamentam a área. Contudo, a natureza abstrata e matemática desses tópicos pode representar um desafio significativo para muitos alunos. Para enfrentar esse desafio, várias ferramentas educacionais foram desenvolvidas, incluindo o JFLAP (*Java Formal Languages and Automata Package*), uma ferramenta amplamente utilizada para visualização e experimentação com autômatos e gramáticas.

A disciplina concentra-se em modelos matemáticos que viabilizam a especificação, criação e identificação de linguagens, juntamente com a análise de suas propriedades e características [Sudkamp 2005]. Apesar de sua importância em diversos campos, como Compiladores e Inteligência Artificial, nota-se uma taxa significativa de reprovação, atingindo a marca de 51% nos cursos de graduação [Terra 2005].

Assim, dentre os vários métodos e metodologias utilizados no processo de ensino e aprendizagem da disciplina, utilizamos dos softwares educativos que representam uma alternativa aos métodos tradicionais. Eles possibilitam que os alunos sejam protagonistas do aprendizado, experimentando e testando seus conhecimentos e ideias.

DOI: 10.5753/sbie.2024.242754

O JFLAP permite que os estudantes explorem de forma interativa esses conceitos, aprimorando a compreensão por meio de simulações e exercícios práticos. Segundo Paiva, Souza e Terra (2023) é a ferramenta mais completa com suporte à manipulação de máquinas e gramáticas, salvar e carregar arquivos no formato XML, realizar múltiplos testes, simular respostas passo a passo e salvar imagem na tela. Porém, apesar de sua popularidade, há uma necessidade de avaliar empiricamente o estado atual do uso do JFLAP e a sua influência no aprendizado.

Neste estudo foram analisados os trabalhos de 28 alunos de graduação em Ciência da Computação na Universidade Estadual de Londrina (UEL) que utilizaram o JFLAP na disciplina de LFA (ministrada no terceiro semestre do curso de Ciência da Computação) com o objetivo de avaliar seu uso, identificando os principais desafios enfrentados pelos alunos nos conteúdos de máquinas reconhecedoras e verificar a eficácia do software como material complementar em sala de aula.

A Seção 2 apresenta a fundamentação teórica necessária para a compreensão dos tópicos abordados em LFA. A Seção 3 define os materiais e métodos utilizados durante a realização desta pesquisa. A Seção 4 mostra os resultados obtidos. Por fim, a Seção 5 apresenta as conclusões.

2. Fundamentação Teórica

Nesta seção serão explorados os fundamentos teóricos essenciais sobre Autômatos, além de revisarmos a literatura relevante sobre o tema.

Bibliografias básicas e complementares de Linguagens Formais e Autômatos foram apresentadas pela docente da referida disciplina, como é o caso de Hopcroft, Motwani e Ullman (2002), Menezes (2010) e Simon (1981). Esses estudos são cruciais para entender as vantagens e limitações de usar software no processo de ensino e aprendizagem. Terra (2016) discute metodologias de ensino de Linguagens Formais e Autômatos, destacando as taxas de reprovação e os desafios enfrentados pelos alunos, enquanto Sudkamp (2005) fornece uma introdução abrangente aos conceitos de LFA com exemplos teóricos e práticos. Sipser (2005) também é uma referência fundamental, que detalha os conceitos essenciais que norteiam autômatos, gramáticas e máquinas de Turing. Lewis e Papadimitriou (2000) complementam essa base com uma abordagem dos elementos restantes da teoria da computação. As linguagens regulares também podem ser expressas por expressões regulares, como podem ser vistas em Jargas (2012).

Gechele e Venske (2007), Pereira e Terra (2018) e Mioni e Barbosa (2022) analisaram a eficácia de diferentes ferramentas educacionais no ensino da disciplina, com destaque para o JFLAP. Pirovani e Mataveli (2013) também já apontava que o JFLAP possibilita também a conversão entre vários modelos computacionais equivalentes. Esses trabalhos serviram de base para o presente trabalho, o qual usa Metodologia Ativa que reside na transição do modelo de ensino centrado no professor para um modelo centrado no aluno, onde o discente assume um papel ativo e participativo em seu próprio processo de aprendizagem. As Metodologias Ativas são fundamentadas em princípios construtivistas que valorizam a participação ativa dos estudantes na construção do conhecimento [Carvalho Neto *et al.* 2023].

A seguir serão abordados autômatos para a classe de linguagem regulares (no caso representados por autômatos finitos, máquinas de Moore e Mealy), linguagens livres de

contexto (descritas por autômatos de pilha) e linguagens sensíveis ao contexto e irrestritas (representadas pelas máquinas de Turing).

2.1. Autômatos Finitos Determinísticos

Os Autômatos Finitos Determinísticos (AFDs) são um tipo específico de autômatos, onde existe uma transição única definida para cada estado e símbolo de entrada. Possuem uma transição para cada estado levando em conta cada símbolo do alfabeto de entrada. Em outras palavras, dado um estado e um símbolo do alfabeto, há exatamente uma transição possível a ser seguida.

De acordo com Gribkoff (2013), esses são naturalmente capazes de representar de forma concisa qualquer sistema que mantenha uma definição interna de estado. Essa capacidade permite que o sistema seja representado utilizando a metodologia e a terminologia dos AFDs.

2.2. Autômatos Finitos Não Determinísticos

Conforme mencionado por Hopcroft, Motwani e Ullman (2002) autômatos finitos não determinísticos (AFNDs) permitem diversas transições para o mesmo par do estado-símbolo, além de uma maior flexibilidade, mas são potencialmente mais complexos.

Segundo Sipser (2005), um AFND é definido de maneira semelhante a um Autômato Finito Determinístico. A principal diferença entre eles está no fato de que, em um AFND, uma transição lendo um símbolo único pode produzir um conjunto de estados ao invés de um único estado, como ocorre em um AFD.

2.3. Máquina de Moore

Baseado em Hopcroft, Motwani e Ullman (2002), a Máquina de Moore (MO) é um tipo de AFD que possui saídas associadas aos estados. A saída de uma MO depende unicamente do estado atual e é independente das entradas. São úteis em aplicações onde a saída deve ser estável e previsível durante transições de estado.

2.4. Máquina de Mealy

As Máquinas de Mealy (ME) são semelhantes às máquinas de Moore exceto por terem saídas associadas às transições. Conforme mencionado por Hopcroft, Motwani e Ullman (2002), as saídas na ME são dependentes tanto do estado atual quanto da entrada, fazendo com que essas tenham uma reação mais rápida às entradas, visto que a saída pode mudar assim que a entrada é recebida [Hopcroft, Motwani e Ullman 2002].

2.5. Pilha

Um Autômato de Pilha (AP) é essencialmente, segundo Hopcroft, Motwani e Ullman (2001), um AFND-ε com a inclusão de uma pilha, que pode ser lida, aumentada e diminuída apenas no topo, exatamente como a estrutura de dados “pilha”. Essa é uma estrutura de dados onde é possível adicionar elementos e cujo acesso segue o modelo LIFO (*Last In*, *First Out*). Isso significa que, em qualquer momento, o único elemento acessível para remoção ou consulta é o último que foi adicionado à pilha, chamado de topo da pilha. Ao inserir um novo elemento na pilha, esse se torna o novo topo. Quando o topo é removido, o elemento que estava imediatamente abaixo se torna o novo topo ou a pilha fica vazia se não houver mais elementos.

2.6. Máquina de Turing

Máquinas de Turing (MT) são modelos computacionais mais eficientes do que autômatos finitos e autômatos com pilha [Sipser 2005]. Consistem em uma fita infinita que pode ser lida e escrita, juntamente com uma cabeça que se move ao longo dela. As MT têm a capacidade de simular qualquer algoritmo computacional e servem como base teórica para definir o que é computável. É um modelo abstrato de computação que foi proposto durante a discussão sobre automatização para solução de problemas. Foi inicialmente introduzido por Alan Turing (1912-1954) em 1936, em seu artigo intitulado '*On Computable Numbers, with an Application to the Entscheidungsproblem*'. Nesse trabalho, Turing demonstrou que para um problema matemático X com resposta binária (sim ou não) não é possível garantir a existência de um algoritmo Y que sempre pare e determine qual a conclusão correta.

Lewis e Papadimitriou (2000) ressalta que as Máquinas de Turing não são simplesmente mais uma classe de autômatos para ser substituídos mais tarde por um tipo ainda mais poderoso. O trabalho de Turing é particularmente significativo por ter sido o primeiro a identificar programas escritos para uma "máquina computacional", como noções intuitivas de efetivamente computável. A intenção do modelo de Turing, denominada *Máquina de Turing*, foi simular, tanto quanto possível, as atitudes humanas relacionadas à computação [Diverio e Menezes 2000]. Em Rosa (2010) são introduzidas as Máquinas de Turing e, então, apresentados os principais resultados de Turing, a universalidade dessas máquinas e a insolubilidade do problema de parada (*halting*). Na sua forma geral, o último resultado diz que nenhum algoritmo pode corretamente decidir se um programa de computador arbitrário para em uma entidade arbitrária.

Trabalhos específicos sobre Máquinas de Turing usando o JFLAP podem ser encontrados em Campano Junior *et al.* (2022) e Campano Junior *et al* (2019). Este presente trabalho se diferencia das referidas pesquisas, uma vez que trabalham outras formas de reconhecedores, além de Máquina de Turing.

3. Materiais e Métodos

Nesta seção serão apresentados os métodos utilizados para avaliar o JFLAP como ferramenta educacional no ensino específico dos conteúdos de autômatos dentro da disciplina obrigatória de Linguagens Formais e Autômatos.

Esse estudo visa analisar como os estudantes utilizaram essa ferramenta para ampliar seus conhecimentos na disciplina mencionada. É importante destacar que este trabalho foi realizado em parceria com a docente responsável pela disciplina, que ao final de cada conteúdo de autômatos sugeriu exercícios ou exemplos a serem testados ou reproduzidos por meio do JFLAP. Essas atividades foram consideradas avaliativas para compor a nota do segundo bimestre (no primeiro bimestre fizeram atividades similar com foco em gramáticas [Michels *et al.* 2024]) da turma de graduação em Ciência da Computação, totalizando 13 atividades avaliadas. A turma tinha 44 alunos matriculados, porém apenas 28 optaram por realizar essas atividades.

A docente da disciplina instruiu os alunos que, ao término de cada tópico, haveria uma atividade ou exemplo para ser reproduzido ou testado na ferramenta JFLAP. Por exemplo, ao final da aula sobre Autômato Finito Determinístico foi proposta uma atividade de fixação para os alunos, a qual seria também realizada no JFLAP. Terminando a disciplina de LFA, os alunos deveriam entregar um relatório contendo todas as

atividades realizadas no JFLAP, incluindo um comentário geral sobre a experiência de utilização da ferramenta para seu aprendizado. Importante mencionar que a docente não relatou especificamente se o JFLAP seria capaz ou não de reproduzir ou analisar as atividades, deixando os alunos livres para explorar as limitações da ferramenta.

Para a avaliação final quanto ao uso de JFLAP para máquinas reconhecedoras foram analisados todos os relatórios dos alunos e realizado um agrupamento das informações para a constituição das notas, considerando as atividades entregues e as impressões dos alunos sobre as resoluções dos exercícios.

4. Resultados

Nesta seção serão apresentados os resultados das atividades realizadas pelos alunos e sua análise.

4.1. Atividade de Ensino

A atividade de ensino consistiu em exemplos que os alunos deveriam reproduzir de forma independente. Embora a proposta fosse realizar a tarefa individualmente e sem consultas externas, alguns alunos optaram por consultar tutoriais ou utilizar a internet. Depois iremos discutir os resultados obtidos pelos alunos.

4.1.1 Autômato Finito Determinístico

Para fixação do conteúdo sobre Autômato Finito Determinístico foi proposta a seguinte atividade para que o aluno verificasse se a referida máquina reconhecia a linguagem L_1 = {w/w possui aa ou bb como subpalavra}. O AFD é $M_1 = (\{q_0, q_1, q_2, q_f\}, \{a, b\}, \delta_1, q_0, \{q_f\})$, onde δ_1 reconhece a linguagem L_1, definido como:

$\delta_1(q_0, a) = q_1$; $\delta_1(q_0, b) = q_2$ $\quad\quad$ $\delta_1(q_2, a) = q_1$; $\delta_1(q_f, a) = q_f$

$\delta_1(q_1, a) = q_f$; $\delta_1(q_1, b) = q_2$ $\quad\quad$ $\delta_1(q_2, b) = q_f$; $\delta_1(q_f, b) = q_f$

Nesse autômato δ_1 é a função de transição que define como o autômato se comporta para cada símbolo do alfabeto {a,b}, garantindo que reconheça a linguagem L_1. Na Figura 1 é apresentada uma solução do exercício com o JFLAP, onde foi testado se o autômato aceita ou não a linguagem. Temos a entrada definida como "*input*" e o resultado após a execução do programa descrito em "*result*". Observa-se que se a cadeia foi aceita em *result* tem "*Accept*", caso contrário "*Reject*".

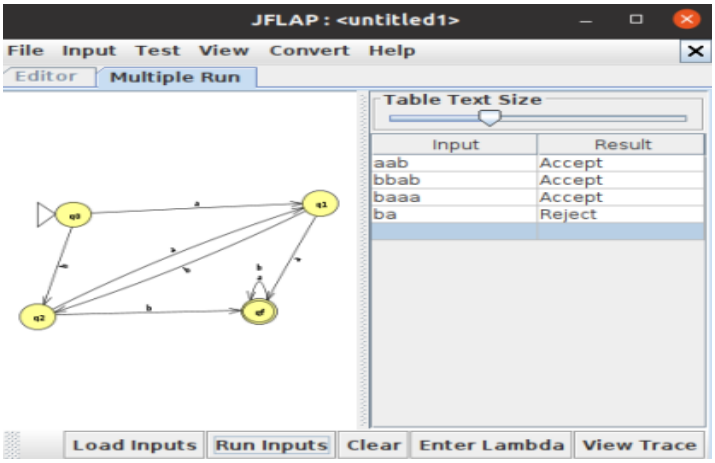

Figura 1. JFLAP - Autômato Finito Determinístico

4.1.2 Autômato Finito Não-Determinístico

Para entender o conteúdo de Autômato Finito Não-Determinístico (AFND), foi considerada também a linguagem L_2 = {w/w possui aa ou bb como subpalavra}. Foi definido o AFND como $M_2 = (\{q_0, q_1, q_2, q_f\}, \{a, b\}, \delta_2, q_0, \{q_f\})$, onde δ_2 reconhece a linguagem L_2. Assim, as funções de transição δ_2 são especificadas como segue:

$\delta_2(q_0, a) = \{q_0,q_1\}$; $\delta_1(q_2, a) = \{\ \}$ $\delta_2(q_1, a) = \{q_f\}$; $\delta_1(q_f, a) = \{q_f\}$
$\delta_2(q_0, b) = \{q_0,q_2\}$; $\delta_1(q_2, b) = \{q_f\}$ $\delta_2(q_1, b) = \{\ \}$; $\delta_1(q_f, b) = \{q_f\}$

O exercício proposto incentivava que o aluno verificasse se o autômato reconhecia a linguagem ou não pelo AFND, utilizando a ferramenta JFLAP. Na Figura 2 é apresentada a simulação da aceitação ou não da linguagem pelo autômato. Observa-se que quando a linguagem é aceita, temos o "input" "*accept*" e, caso contrário, "*reject*". Isso ilustra como a ferramenta auxilia o entendimento do aluno na resolução desse exercício.

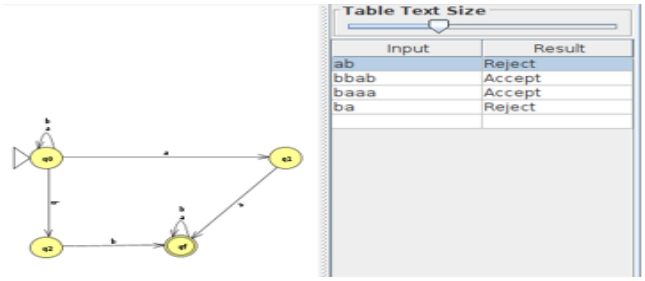

Figura 2. JFLAP - Autômato Finito Não-Determinístico

4.1.3 Gramática e sua relação com autômatos

Outra atividade proposta envolve gramáticas e sua relação com autômatos. Para esse experimento foi dado o exemplo de um autômato que aceita as cadeias do exemplo, definido pelo AF = $(\{S, B, Q_f\}, \{0, 1\}, \delta, S, Q_f)$:

$\delta(S,0) = \{B\}$; $\delta(S, 1) = \{\ \}$; $\delta(B,0) = \{B, Q_f\}$; $\delta(B, 1) = \{S\}$.

Sabendo que a GR é definida como $S \rightarrow 0B$; $B \rightarrow 0$; $B \rightarrow 0B$; $B \rightarrow 1S$.

Para facilitar a compreensão foram rastreadas em aula pela docente cadeias que devem ser aceitas nas simulações dos alunos. É o caso, por exemplo, da cadeia 001000100, exibida em: {(S, 001000100) |-- (B, 01000100) |-- (B, 1000100) |-- (S, 000100) |-- (B, 00100) |-- (B, 0100) |-- (B, 100) |-- (S, 00) |-- (B, 0) |-- (Q_f, λ)}. A cadeia foi aceita, pois chegou em Q_f tendo λ no final do rastreio consumindo todos os dígitos.

Na Figura 3 são apresentados todos os passos para a resolução da atividade proposta. Primeiramente, insere-se o autômato em '*Finite Automaton*' Após isso, solicita-se a sua conversão seguindo os passos '*Convert*' e '*Convert to Grammar*'. Em seguida é obtida uma gramática, cuja linguagem é a mesma representada pelo AF. Observa-se que ao final da atividade temos o resultado positivo com '*Accept*' para a cadeia dada.

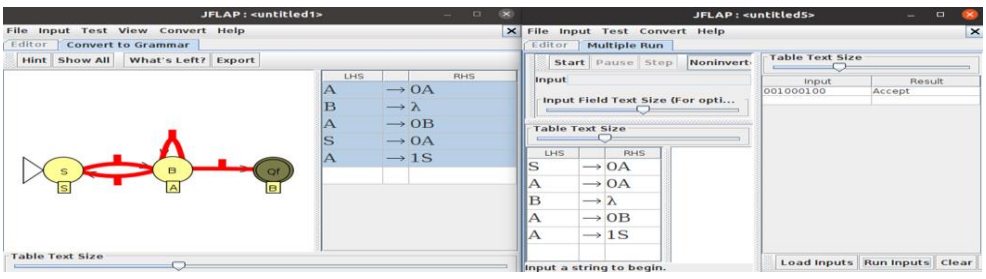

Figura 3. JFLAP - Conversão de Autômatos em Gramática Linear à Direita

4.1.4 Máquina de Moore

Como continuação das atividades de ensino, temos o tópico de Máquinas de Moore, onde a máquina MO é definida como (Σ, Q, δ, q_0, Δ, δ_S) onde: Σ é alfabeto de símbolos de entrada; Q é conjunto de estados possíveis do autômato, o qual é finito; δ é função programa ou função de transição: δ: Q x $\Sigma \rightarrow$ Q a qual é função parcial; q_0 estado inicial tal que q_0 é elemento de Q: F é conjunto de estados finais tal que F está contido em Q; Δ é alfabeto de símbolos de saída; e q_S é função de saída tal que δ_S: Q $\rightarrow \Delta^*$ a qual é uma função total.

Os alunos testaram o exemplo de Menezes (2010), onde MO= ({a, β}, Q, δ_{MO}, <q, ε>, F, {a, β}, δ_S) tal que Q = F = {q, p} × {ε, a, β} ilustrada na Figura 4.

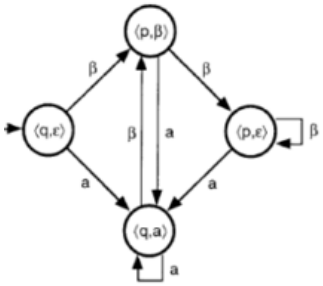

Figura 4. Máquina de Moore (Menezes 2010)

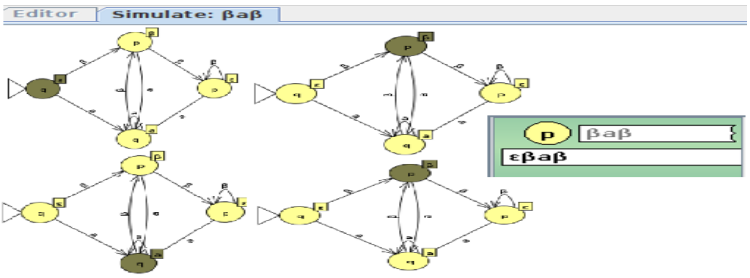

Figura 5. JFLAP - Máquina de Moore utilizando JFLAP.

Para reproduzir a Máquina de Moore utilizando a entrada da cadeia "*βaβ*" no JFLAP, temos os seguintes passos: selecionar a opção "*Moore Machine*" para criar uma nova MO e definir os componentes dessa conforme a especificação: Estados (Q): {q, p}; Alfabeto de entrada: {β, a}; Função de transição (δ_{MO}):
δ_{MO}(q, β) = p (significa que ao ler 'β' no estado q, transita para o estado p)
δ_{MO}(q, a) = q (significa que ao ler 'a' no estado q, permanece no estado q)
δ_{MO}(p, β) = p (significa que ao ler 'β' no estado p, permanece no estado p)
δ_{MO}(p, a) = q (significa que ao ler 'a' no estado p, transita para o estado q)
Saída de estados (Γ): Para q: β; Para p: a; Estado inicial: q (isto é, <q, β>); Estados finais (F): {q}. Após configurar todos esses elementos no JFLAP, é possível simular a Máquina de Moore com a entrada "βaβ" para verificar seu funcionamento.

Na Figura 5 temos a reprodução da Máquina de Moore pelo software JFLAP. Iniciamos a entrada com '*βaβ*', que resulta em 'ε '. Na próxima etapa, temos ' ε*β*', seguido por '*εβa*' e, finalmente, ' ε*βaβ*'. Observa-se que cada etapa é ilustrada no JFLAP e suas mudanças de estados são mostradas de forma intuitiva, facilitando o entendimento.

4.1.5 Máquina de Mealy

Com complemento da atividade de Máquina de Moore, temos a Máquina de Mealy, definida como ME= (Σ, Q, δ, q_0, F, Δ), onde [Menezes 2010]: Σ é alfabeto de símbolos de entrada, Q é conjunto de estados possíveis do autômato o qual é finito; δ é função programa ou função de transição: δ: Q x $\Sigma \rightarrow$ Q x Δ^* a qual é uma função parcial; q_0 é estado inicial do autômato tal que q_0 é elemento de Q; F é conjunto de estados finais tal que F está contido em Q; Δ é alfabeto de símbolos de saída.

Os alunos testaram o exemplo de Menezes (2010) onde a Máquina de Mealy ME=({a,β}, {q,p}, δ_{ME}, q, <q,p>, {a, β})} que compacta os brancos de um texto onde a representa um símbolo qualquer do texto e β o símbolo branco, como ilustrado na Figura 6. Na Figura 7 temos o resultado final da simulação da ME pelo software JFLAP. Para chegar a esse resultado foi utilizada a entrada da cadeia '*βββaa*' e foi feita a simulação em '*Mealy Machine*'. Ao entrarmos com essa cadeia, temos as seguintes sequências de estados: *β -> βε -> βεε -> βεεa -> βεεaa*. Ao final, temos o resultado da leitura da máquina. É possível verificar que a ferramenta mostra passo a passo os estados da máquina e ao fim apresenta o resultado de aceitação ou não.

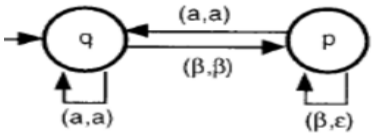

Figura 6. Máquina de Mealy (Menezes 2010)

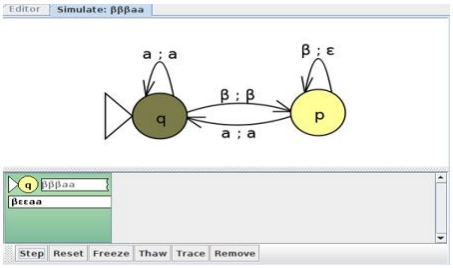

Figura 7. JFLAP - Máquina de Mealy

4.1.6 Autômato de Pilha

Também foram dados em aula na disciplina de LFA dois métodos de Autômatos de Pilha e vamos exibir um deles que é por meio do alcance do estado final [Menezes 2010]. Considerando a seguinte linguagem L_3 ={ww^R/w pertence a {a,b}*} onde δ_3 :

$\delta_3(q_0, a, \varepsilon) = \{(q_0, a)\}$ $\delta_3(q_0, b, \varepsilon) = \{(q_0, b)\}$ $\delta_3(q_0, \varepsilon, \varepsilon) = \{(q_1, \varepsilon)\}$
$\delta_3(q_1, a, a) = \{(q_1, \varepsilon)\}$ $\delta_3(q_1, ?, ?) = \{(q_f, \varepsilon)\}$ $\delta_3(q_1, b, b) = \{(q_1, \varepsilon)\}$

O autômato M_3 ilustrado na Figura 8 tal que ACEITA (M_3)=L_3. Note que M_3 é não-determinístico devido ao movimento vazio de q_0 para q_1. Além disso, o alfabeto auxiliar é igual ao de entrada. Em q_0 é empilhado o reverso do prefixo. A cada símbolo empilhado, ocorre um movimento não-determinístico para q_1 o qual verifica se o sufixo da palavra é igual ao conteúdo da pilha. Na Figura 9 temos a simulação do Autômato de Pilha, onde é possível verificar quais cadeias foram aceitas ou rejeitadas pelo autômato, considerando as cores roxa para rejeição e verde para aceitação. Na simulação, foi usada a cadeia de entrada "*aabbaa*" e criado o autômato com o comando "*Pushdown Automaton*".

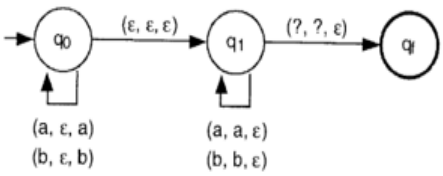

Figura 8. JFLAP - Autômato de Pilha (Menezes 2010)

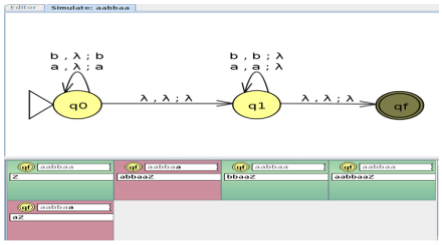

Figura 9. JFLAP- Autômato de Pilha

4.1.7 Máquina de Turing

O exercício final da atividade da disciplina consiste da simulação de uma Máquina de Turing M=(Σ, Q, δ, q_0, F, V, β, ©), onde Σ é alfabeto de símbolos de entrada; Q é conjunto de estados possíveis da máquina o qual é finito; δ é função programa ou função de transição: δ: Q x (Σ ∪ V ∪ {B, ©)}→ Q x (Σ ∪ V ∪ {B, ©))x{E,D} a qual é uma função

parcial; q_0 é estado inicial da máquina tal que q_0 é elemento de Q; F é conjunto de estados finais tal que F está contido em Q; V é alfabeto auxiliar (pode ser vazio); β é símbolo especial branco; © é símbolo ou marcador de início de fita.

Para isso consideramos a seguinte linguagem $L_4 = \{a^n b^n \mid n > 0\}\}$. Nesse exemplo também é citado que a Máquina de Turing MT (Figura 10) possui duplo Balanceamento, sendo MT = ({a, b}, {q_o, q_1,q_2 q_4}, δ_1,q_0,{q_4},{A,B} β,}.

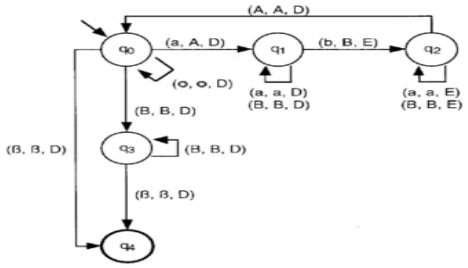

Figura 10. JFLAP - Máquina de Turing (Menezes 2010)

Dois testes de cadeia aaabb (Figura 11) aaabbb (Figura 12) foram realizados para verificar se a máquina aceita ou não a cadeia. Lembrando que deve estar de acordo com a linguagem $a^n b^n$. Nas Figuras 11 e 12 temos a simulação da Máquina de Turing, onde é possível verificar quais cadeias foram aceitas ou rejeitadas. Conforme esperado, a cadeia "*aaabb*" é rejeitada pela MT, enquanto a cadeia "*aaabbb*" é aceita. É importante ressaltar que essa última atividade foi ensinada aos alunos na última aula da disciplina. Ainda assim, apenas 2 alunos não entregaram a simulação da MT, enquanto 26 alunos completaram normalmente essa atividade.

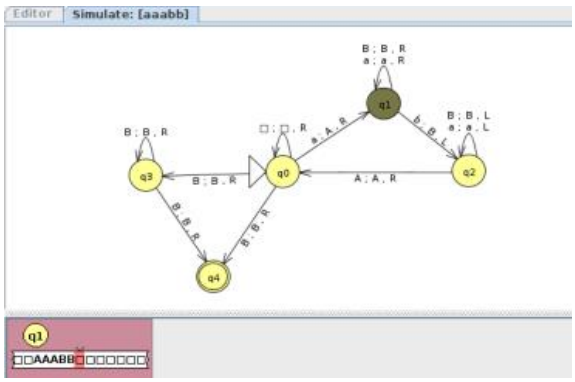

Figura 11. JFLAP - Máquina de Turing testando cadeia "aaabb".

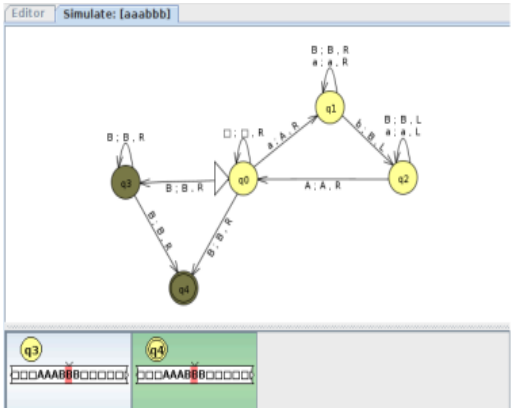

Figura 12. JFLAP - Máquina de Turing testando cadeia "aaabbb".

A Tabela 1 apresenta uma fácil visualização da avaliação realizada, apresentando o número de alunos que avaliaram o uso do JFLAP para o ensino de máquinas reconhecedoras e seus pontos positivos e negativos destacados pelos alunos. Foram inseridos na tabela os pontos que possuíam mais de uma citação na soma dos resultados.

Como **Pontos Positivos** três alunos mencionaram que o JFLAP permite visualizar em tempo real o comportamento dos autômatos, o que facilita a compreensão dos conceitos teóricos. Dois alunos destacaram que o JFLAP ajuda a identificar padrões de aceitação e rejeição de palavras. Dois alunos apontaram que o software permite realizar simulações passo a passo de máquinas reconhecedoras, o que facilita o entendimento dos

conceitos complexos. Por fim, dois alunos elogiaram a capacidade do JFLAP de converter autômatos AFND em AFD.

Já como **Pontos Negativos** cinco alunos consideraram a interface do JFLAP ultrapassada e apontaram dificuldade na navegação e uso eficiente do software. Quatro alunos mencionaram a falta de recursos de depuração para correção de erros. Quatro alunos relataram que a documentação disponível é insuficiente e dificulta a compreensão dos recursos e a forma de utilizá-los. Dois alunos apontaram que o JFLAP tem uma limitação, pois não permite transformar Autômato Finito (AF) em Expressão Regular (ER). Cabe ressaltar que a docente deu três métodos (Regras de Arden, Método Indutivo e Equações Simultâneas) de transformação de AF para ER em sala de aula que infelizmente não teve como testar no JFLAP devido à limitação da ferramenta.

Tabela 1. Avaliação do Software JFLAP Fonte: Os autores.

Categoria	Feedback	Número de alunos
Pontos Positivos		
	Visualização em tempo real	3
	Identificação de padrões	2
	Simulações passo a passo	2
	Conversão de autômatos	2
Pontos Negativos		
	Interface Ultrapassada	5
	Documentação Insuficiente	4
	Falta de recursos de depuração	4
	Limitação na transformação de AF para ER	2

5. Conclusões

Avaliações empíricas realizadas em situações práticas que envolveram o uso do JFLAP em atividades de construção de modelos formais por alunos da disciplina de Linguagens Formais e Autômatos no curso de Bacharelado em Ciência da Computação da UEL indicaram que a ferramenta foi eficaz para o ensino de máquinas reconhecedoras, contribuindo significativamente para a compreensão e prática dos conceitos teóricos de LFA. No entanto, melhorias na interface e na documentação são necessárias para otimizar ainda mais a experiência do usuário. Utilizando os resultados obtidos aqui, planejamos implementar essas melhorias, visando aprimorar o ensino da disciplina e proporcionar uma experiência de aprendizado ainda mais efetiva para os alunos.

Para pesquisas futuras pretende-se incorporar outras turmas do curso para realizar uma análise mais abrangente e diversificada. Isso permitirá expandir a amostra de dados. Ao comparar vários grupos de alunos, pode-se realizar uma avaliação pedagógica mais ampla, permitindo a comparação de diversas expectativas e resultados entre as turmas. Reconhecemos que alunos tendem a divagar quando têm liberdade para expressar suas opiniões, dificultando a obtenção de insights claros e práticos. Portanto, pretende-se tornar novas pesquisas mais estruturadas e focadas, limitando o escopo das perguntas facilitando a análise dos resultados. Assim, espera-se que esses sejam mais precisos e úteis e que possam ser usados para melhorar ainda mais o ensino e aprendizagem de LFA.

Referências

Campano Junior, M. M., Barbosa, C. R. S. C. de, Felinto, A. S. Aylon, L. B. R. (2022). Multiplicando números binários com Máquinas de Turing: Interdisciplinaridade no Ensino de Computação. In: *Anais do XXXIII Simpósio Brasileiro de Informática na Educação.* SBC, Manaus. p. 1313-1323.

Campano Junior, M. M., Faria, C. R., Barbosa, C. R. S. C. de, Felinto, A. S. (2019). Um merge entre Máquinas de Turing e Operações Matemáticas em Binário no Ensino de Linguagens Formais e Autômatos. In: *Anais do XXIV Congresso Internacional de Informática Educativa*. Jaime Sánchez (Ed.), Arequipa, Peru. p. 78-83.

Carvalho Neto, R., Victor, V. F., Cavalcante, R. P., Castilho, W. S. e Senna, M. L. G. S. (2023). Metodologias Ativas: Teoria da Aprendizagem. In: *Humanidades e Inovação*, v.10, n.9, páginas 141-153.

Diverio, T. A. e Menezes, P. B. (2000). *Teoria da Computação: máquinas universais e computabilidade.* Porto Alegre: Instituto de Informática da UFRGS – Sagra Luzzato, 2ª edição.

Gechele, L. M. G. e Venske, S. M. H. S. (2007). Ferramentas de Ensino de Linguagens Formais: um comparativo. In: *Anais do I Encontro de Iniciação Científica do PROIC*. Unicentro, Guarapuava, p. 110-119.

Gribkoff, E. (2013). Applications of deterministic finite automata. In: *ECS Lectures Notes 120.* University of California, pages 1-9.

Hopcroft, J. E., Ullman, J. D. e Motwani, R. (2002). *Introdução à Teoria de Autômatos, Linguagens e Computação.* Tradução de Vandenberg D. de Souza. Rio de Janeiro: Elsevier.

JFLAP (2023). https://www.jflap.org/, April.

Jargas, A. M. (2012). *Expressões Regulares: uma abordagem divertida*. São Paulo: Novatec.

Lewis, H. R. e Papadimitriou, C. H. (2000). *Elementos de Teoria da Computação*. Tradução: Edson Furmankiewicz. Porto Alegre: Bookman, 2ª edição.

Menezes, P. B. (2010). *Linguagens Formais e Autômatos.* Vol. 3. Porto Alegre: Bookman, 6ª edição.

Menezes, P. B. e Diverio, T. A. (2000). *Teoria da Computação: Máquinas Universais e Computabilidade*. Porto Alegre: Instituto de Informática da UFRGS – Sagra Luzzato, 2ª edição.

Michels, J. F. P., Cassanho, L. P., Burigo, B. R. e Barbosa, C. R. S. C. de. (2024). Avaliação do JFLAP como Ferramenta de Ensino de Gramáticas na Disciplina de Linguagens Formais e Autômatos. In: *Anais do XXXV Simpósio Brasileiro de Informática na Educação.* SBC, Rio de Janeiro.

Mioni, J. L.V. M e Barbosa, C. R. S. C. de. (2022). Ferramentas para o Aprendizado de Linguagens Formais e Autômatos. In: *Anais Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital.* SBC, Natal. p. 969-978.

Paiva, P., Souza, M. e Terra, R. (2023). Ferramentas de apoio para a disciplina de Linguagens Formais e Autômatos: uma proposta de uso. In: *Anais do XXXIV Simpósio Brasileiro de Informática na Educação.* SBC, Passo Fundo. p.1698-1709.

Pereira, C. H. and Terra, R. (2018). A mobile app for teaching formal language and automata. In: *Computer Applications in Engineering Education.* v.26, n.5, pages 1742-1752.

Pirovani, J. P. C. e Mataveli, G. V. (2013). Estudo e adaptação de software para o ensino de Linguagens Formais e Autômatos. In: *Revista Brasileira de Informática na Educação*. v. 21, n.3, páginas 2373-2382.

Rosa, J. L G. (2010). *Linguagens Formais e Autômatos*. Rio de Janeiro: LTC.

Simon, I. (1981). *Linguagens Formais e Autômatos.* São Paulo: Escola de Computação.

Sipser, M. (2005). *Introduction to the Theory of Computation*. Boston: Thomson Course Technology. 2ª edição internacional.

Sudkamp, T. A. (2005). *Languages and Machines: An Introduction to the Theory of Computer Science*. Boston: Pearson.

Terra, R. (2016). *Dados da disciplina de Linguagens Formais e Autômatos*. Technical Report. Universidade Federal de Lavras. p. 1–547.