Ferramentas para o Aprendizado de Linguagens Formais e Autômatos

José Luiz Villela Marcondes Mioni[1], Cinthyan Renata Sachs C. de Barbosa[1]

[1]Programa de Pós-Graduação em Ciência da Computação –
Universidade Estadual de Londrina (UEL)
Caixa Postal 10.001 – 86.057-970 – Londrina – PR – Brazil

{jose.luiz.villela, cinthyan}@uel.br

Abstract. *This work collects a number of different tools that can be applied to the use, study and application of concepts in the subject of Formal Languages and Automata, in addition it draws a brief comparison between their different characteristics functionalities and behaviors, which can be useful for this subject in the Computer Science course. As this subject is often taught in a theoretical way, tools which can help the teaching and learning process are very well appreciated.*

Keywords. *Formal Languages and Automata. Educational Tools.*

Resumo. *Este trabalho coleta diferentes ferramentas que podem ser aplicadas no uso, estudo e aplicação de conceitos na disciplina de Linguagens Formais e Autômatos, além de traçar um breve comparativo entre suas diferentes características, funcionalidades e comportamentos, o que pode ser muito útil na referida disciplina do curso de Ciência da Computação. Muitas vezes essa é dada de maneira apenas teórica, assim, ferramentas para auxiliar no processo de ensino e aprendizagem são bem-vindas.*

Palavras-chave. *Linguagens Formais e Autômatos. Ferramentas Educativas.*

1. Introdução

A aplicação e compreensão do conteúdo de Linguagens Formais e Autômatos (LFA) em disciplinas e cursos de Ciência da Computação, apesar de fundamental na fixação de conceitos e habilidades importantes para o profissional em formação, pode se provar uma atividade desafiadora e abstrata quando abordada por iniciantes.

É comum que os materiais e as atividades didáticas de LFA adotem um enfoque essencialmente algébrico, o que exige dos alunos não apenas uma boa formação matemática, mas também, e principalmente, uma grande capacidade de raciocínio lógico e abstrato. Por esse motivo, os estudantes de tais textos precisam lidar com um nível de complexidade adicional, além daquele que é inerente aos assuntos da área, para conseguir bons resultados na disciplina [Ramos 2009].

Embora fundamental na formação superior em computação, formalismo de linguagens e compiladores são matérias cuja compreensão dos formalismos envolvidos e das fases de compilação não é simples. Essa característica justifica a proposição de ferramentas de apoio com fins didático [Alckmin e Mello 2010]. Isso permite ao aluno

vivenciar, em ambientes de simulação, os principais conceitos vistos na teoria. Ao contemplar o estudante com uma estratégia didática complementar e alternativa em relação àquela exclusivamente algébrica, criam-se as condições para que, por meio de perspectivas diferentes, ele obtenha um nível de compreensão mais completo em relação à matéria. Nesse contexto, o uso de ferramentas que aumentem a interação e a resposta visual do conteúdo ao interagir com o usuário pode fazer da experiência um caminho mais intuitivo, lúdico e simples para diversos alunos.

Com esse objetivo em pauta, este trabalho coleta diferentes ferramentas que podem ser aplicadas no uso, estudo e aplicação de conceitos de Linguagens Formais e Autômatos, mais especificamente: autômatos finitos determinísticos (AFDs), autômatos finitos não determinísticos (AFNDs), conversões de AFNDs para AFDs, autômatos de pilha (AP) e representações de transições na forma escrita e gráfica.

Este trabalho é organizado como segue: a seção 2 aborda conceitos e definições de elementos de LFA, como autômatos finitos determinísticos e não determinísticos, bem como autômatos de pilha; a seção 3 lista algumas das principais ferramentas disponíveis atualmente e explica suas funções, comportamento e interface; as seções 4 e 5 traçam, respectivamente, uma análise comparativa entre as ferramentas e expõem as conclusões do trabalho.

2. Autômato Finitos

Nesta seção introduz-se a classe básica de dispositivos de computação chamados de autômatos finitos [Rosa 2010]. Os autômatos finitos envolvem estados e transições entre esses em resposta às entradas. Eles são úteis para a construção de vários diferentes tipos de software, incluindo o componente de análise léxica de um compilador e sistemas para verificar a exatidão dos circuitos ou protocolos. Os autômatos finitos constituem um modelo útil para muitos elementos de hardware e software [Hopcroft, Motwani and Ullman 2001].

De maneira simplificada, um autômato é uma representação gráfica ou algébrica de uma determinada linguagem chamada regular, sendo composto normalmente de um conjunto de estados (entre esses, ao menos um estado inicial e um estado final que pode ser o mesmo ou não) que tem o propósito de memorizar a porção relevante da história do sistema [Hopcroft, Motwani and Ullman 2001]. Cabe ressaltar que ele não memoriza a história inteira e sim apenas o que é importante naquele momento de reconhecimento da palavra (cadeias). As linguagens regulares também podem ser expressas por expressões regulares, como pode ser vista em Jargas (2012).

Existem diferentes tipos de autômatos. Este trabalho aborda ferramentas focadas na criação, manipulação e entendimento de autômatos finitos determinísticos (há apenas um caminho para uma transição de estados a partir de um único símbolo lido), não determinísticos (há múltiplos caminhos para uma transição envolvendo um determinado símbolo) e Autômatos de Pilha (onde o critério de aceitação envolve não só a palavra analisada, mas também a esvaziamento de sua pilha) [Hopcroft, Motwani and Ullman 2001].

Formalmente, um Autômato Finito (AF) M é uma 5-tupla [Menezes 1998] $M = (Q, \Sigma, \delta, q0, F)$ tal que: • Q conjunto finito não vazio de estados; • Σ alfabeto de

símbolos de entrada (corresponde a um conjunto finito não vazio dos símbolos de entrada ou átomos indivisíveis que compõem a cadeia de entrada submetida ao autômato para aceitação); • δ função programa ou função de transição δ: Q X (Σ U ε) → Q', a qual é uma função parcial (essa função mapeia o produto cartesiano Q X (Σ U ε) em Q', ou seja, fornece cada par (estado, símbolo de entrada) um novo estado para onde o autômato deverá mover-se); • q0 estado inicial tal que seja elemento de Q (é o estado para o qual o reconhecedor deve ser levado antes de iniciar suas atividades); • F conjunto de estados finais tal que F está contido em Q. Estados em que o autômato deve terminar o reconhecimento das cadeias de entrada que pertencem à linguagem que o autômato define.

Já um Autômato de Pilha (AP) é essencialmente, segundo Hopcroft, Motwani and 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".

3. Ferramentas para o aprendizado de Linguagens Formais e Autômatos

Palavras-chave de busca pela web como "ferramentas para aprenzidado de Linguagens Formais e Autômatos", "ferramentas para Linguagens Formais e Autômatos", "*tools used to learn Formal Languages and Automata*", "*Formal Language tools online*", "*Formal Languages and Automata tools*" foram aplicadas. Em seguida, critérios de exclusão como usabilidade, datas de publicações, relevância em algoritmos de busca e potencial didático foram aplicados.

A partir da revisão sistemática de literatura realizada durante a execução deste trabalho utilizando os critérios mencionados acima, um conjunto de aplicações para o aprendizado de LFA foi obtido, porém identificou-se que apesar da existência e catálogo de ferramentas que auxiliem a manipulação de elementos existentes nas disciplinas como autômatos finitos e de pilha de forma lúdica [Rodger *et al.* 1997], poucas ferramentas têm sido usadas nesses cursos [Brito Junior e Aguiar 2019].

Apesar do relativamente extenso número de aplicações abordando o tema, este trabalho apresenta uma breve descrição das ferramentas que tiveram maior incidência durante o processo de pesquisa, assim como maior disponibilidade de acesso e processo de instalação ou execução com complexidade adequada às habilidades de um iniciante de graduação no curso de Ciência da Computação. As principais funcionalidades e aplicações de algumas dessas ferramentas são listadas nas subseções abaixo e podem ser inseridas na disciplina de Linguagens Formais e Autômatos do curso de Ciência da Computação, seja por meio de software instalável, web ou aplicativos [Brito Junior e Aguiar 2019].

Muitas dessas ferramentas podem ser úteis também na disciplina de Compiladores para desenvolver analisadores léxicos cujo objetivo é separar a sequência de caracteres do texto de um programa-fonte em itens léxicos ou lexemas que são sequências de caracteres com um significado coletivo [Barbosa, Bonidia e Coelho Neto 2019], [Barbosa, Faria e Campano Junior 2021].

3.1. JFLAP

JFLAP (2018) é um pacote de ferramentas gráficas que pode ser usado como um auxílio no aprendizado dos conceitos básicos de Linguagens Formais e Teoria dos Autômatos.

Disponibilizado por meio de uma aplicação construída em JAVA, JFLAP realiza diversas ações dentro do conteúdo de LFA. JFLAP permite a criação de AFND, AFD, Autômatos de Pilha e Máquinas de Turing (MT).

Diferentes tipos de autômatos podem ser gerados pelo JFLAP (Figura 1), com a criação e manipulação de estados por meio do mouse para criar estados, transições e deletar elementos. JFLAP possui o teste iterativo de cada estado e conversão de AFND para AFD. Uma utilização do JFLAP para MTs pode ser encontrada em Campano Junior *et al.* (2019).

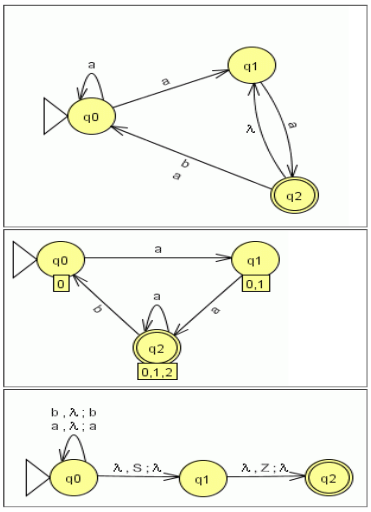

Figura 1. Diferentes tipos de autômatos gerados peo JFLAP

3.2. Automaton Simulator

Automaton Simulator [Doty 2020] é uma ferramenta visual *online* de código aberto que permite a criação de AFND, AFD e AP. As transições podem ser elaboradas ao clicar nas setas, assim como há também a criação de estados.

A interface gráfica permite testar cadeias no autômato resultado, assim como o acesso ao histórico de testes realizados durante a sessão de usuário. Os estados podem ser editados e o usuário pode determinar quais são os finais, o inicial e seus nomes. A Figura 2 apresenta um AFD criado no Automaton Simulator.

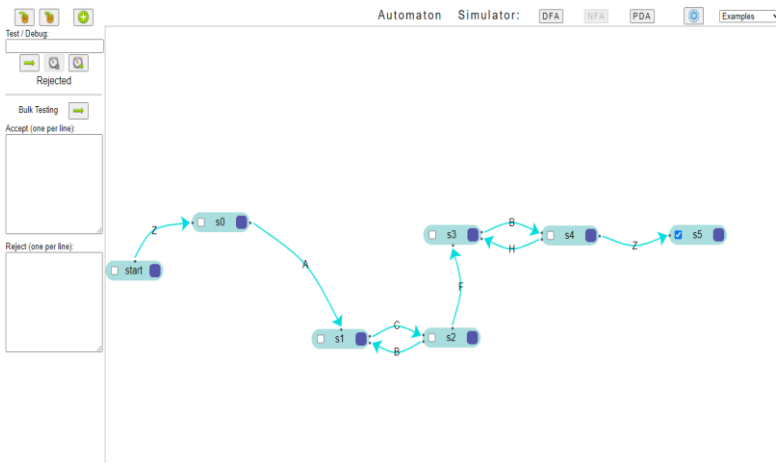

Figura 2. AFD criado com o Automaton Simulator [Doty 2020]

3.3. UC Davis Automaton Simulator

UC Davis Automaton Simulator [Dickerson 2021] é uma ferramenta *online* que permite diversas operações e reconhecimentos por meio de código similar ao do JavaScript. Com a referida ferramenta é possível gerar a representação de transições e estados de um AFD e AFND, além de expressões regulares, gramáticas livres de contexto e MTs.

O processo é um pouco diferente das ferramentas mais comuns. A inserção dos estados em formato de código cria a representação gráfica apenas dos estados e, na sequência, é possível usar um campo de inserção de dados para testar strings no AFD/AFND em tempo real. Dessa forma, o bloco de código abaixo gera a representação visual apresentada na Figura 3.

```
states = {q, q0, q00, q000}
input_alphabet = {0,1}
start_state =  q
accept_states =  {q,q0,q00}
delta =
  q,1   -> q;
  q0,1  -> q;
  q00,1 -> q;
  q000,1 -> q;
  q,0   -> q0;
  q0,0  -> q00;
  q00,0 -> q000;
  q000,0 -> q000;
```

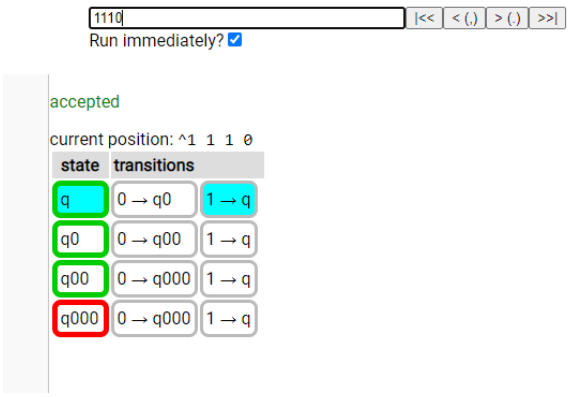

Figura 3. Representação de estados e transições a partir de código no UC Davis Automaton Simulator usada para testar a cadeia 1110

3.4. Autosim

Autosim, conforme Burch (2006), permite o desenho e simulação de uma variedade de máquinas teóricas, incluindo autômatos finitos determinísticos, autômatos finitos não determinísticos, autômatos de pilha determinísticos e máquinas de Turing. O programa deve ser executado em qualquer plataforma com Java 1.3 e está disponível sob a Licença Pública GNU.

A criação de estados é por meio de interface gráfica (Figura 4) e seu comportamento é similar aos outros tipos de ferramentas visuais apresentadas neste trabalho.

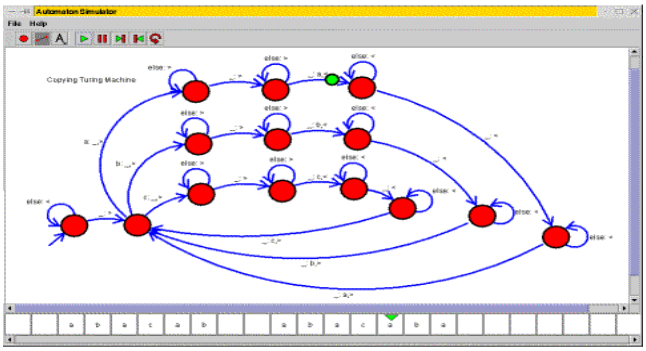

Figura 4. Captura de tela do programa Autosim

3.5. Finite State Machine Designer

O Finite State Machine Designer [Wallace 2010], conhecido como FSMD, é uma ferramenta *online* que permite a criação e edição de autômatos finitos determinísticos e não determinísticos com uma interface gráfica intuitiva, de maneira similar a outras ferramentas gráficas (cliques duplos adicionam estados e um clique com arrasto conecta estados, etc.). Isso permite a criação de um AF como o da Figura 5.

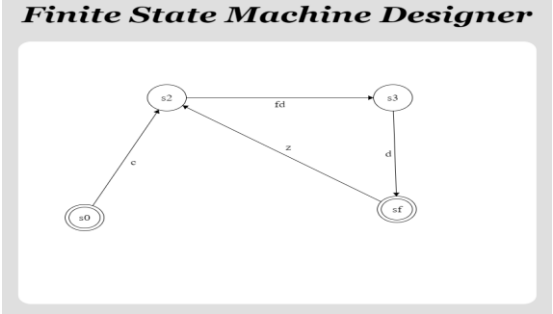

Figura 5. AFD projetado utilizando a FSMD

Uma funcionalidade única dentre os softwares abordados neste trabalho é a possibilidade do FSMD de exportar o autômato criado no formato de imagem PNG ou em um arquivo Gráfico de Vetor Escalável (SVG – *Scalable Vector Graphics*) e até mesmo no formato do editor LaTeX. Dessa maneira, o AFD exemplificado na Figura 5 pode ser exibido graficamente em um arquivo de texto no LaTeX. Há um fragmento do código necessário abaixo:

```
\documentclass[12pt]{article}
\usepackage{tikz}
\begin{document}
\begin{center}
\begin{tikzpicture}[scale=0.2]
\tikzstyle{every node}+=[inner sep=0pt]
\draw [black] (10.2,-40.2) circle (3);
\draw (57.8,-12.8) node {$s3$};
\draw [black] (22.9,-12.8) circle (3);
```

3.6. FSM Simulator

Já ao observar o FSM Simulator [Bruch 2006], os autômatos são construídos usando código. Estados e outras informações particulares como símbolos iniciais são precedidos do símbolo # que possibilita a criação de blocos de informações que determinam o conjunto de estados, alfabeto, transições entre outros. A aplicação em seguida interpreta as informações inseridas e cria uma representação visual como na Figura 6.

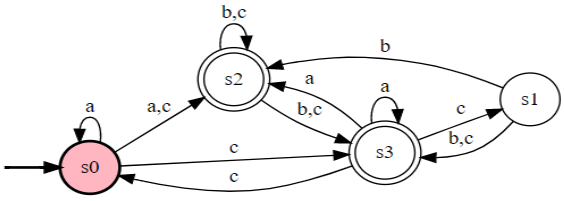

Figura 6. Visualização de um AFND criado com FSM

3.7. jFAST

Segundo White [White 2006], jFAST é uma ferramenta produzida em Java que permite ao usuário criar, editar e simular autômatos finitos e máquinas de estado. Ele fornece um método simples e intuitivo de interagir com vários tipos de máquinas de estado finito (FSM – *finite state machines*), incluindo AFD, AFND, APs, máquinas de estado (aqui chamada de SM – *states machine*) e máquinas de Turing (TM- *Turing Machine*). O jFAST também fornece ao usuário a capacidade de verificar sua FSM em diferentes entradas e fornece mensagens claras e significativas quando erros são encontrados.

Assim como em algumas ferramentas encontradas, o jFAST fornece uma interface gráfica funcional que permite a inserção e conexão de estados usando o ponteiro do mouse (demonstrado na Figura 7).

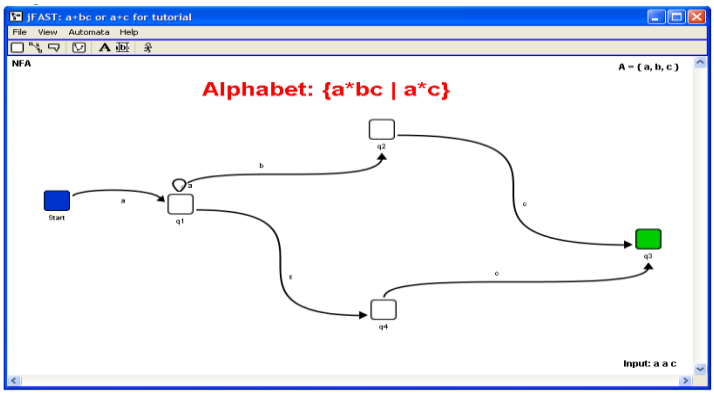

Figura 7.Captura de tela durante a execução do jFAST

4 Análise Comparativa das Ferramentas

Dentre as ferramentas observadas, os principais critérios avaliados foram sua flexibilidade de execução e capacidade de aplicação.

O quesito de flexibilidade foi verificado pela possibilidade de uma ferramenta em ser executada a partir de um navegador Web, sem a necessidade de instalação local e

pela presença de uma Interface Gráfica de Usuário (IGU), capaz de aumentar sua usabilidade e facilitar o aprendizado. Já o quesito de capacidade de aplicação foi medido por meio dos diferentes formatos e transições possíveis de serem gerados em cada ferramenta, o que amplia sua aplicação ao longo da execução da disciplina.

As possibilidades disponibilizadas por cada uma das ferramentas apontadas neste trabalho são abordadas na Tabela 1.

Tabela 1. Comparativo de possibilidades entre as ferramentas

Aplicação	AFD	AFND	AP	GUI	AFND- AFD	Web / Instalável
JFLAP	X	X	X	X	X	Instalável
Automaton Simulator	X	X	X	X		Web
UC Davis Automaton Simulator	X	X		X		Web
Autosim	X	X	X	X		Instalável
Finite State Machine Designer	X	X		X		Web
FSM Simulator	X	X	X			Web
jFAST	X	X	X	X		Instalável

Pode-se observar que a maioria das ferramentas analisadas atende AFD e AFND. Entretanto, apesar de abordar o mesmo tópico, o método para obter o resultado pode variar de acordo com cada ferramenta, seja por meio da manipulação de uma interface visual ou usando habilidades análogas à programação para a manipulação do autômato.

Nota-se a possibilidade de abordar o estudo do tópico por meio do correlacionamento de disciplinas de programação, uma vez que algumas das ferramentas para o aprendizado de Linguagens Formais e Autômatos usam linhas de código ou mesmo a resolução de exercícios e o estudo em dispositivos móveis ao usarmos as ferramentas já hospedadas na web.

5. Conclusões

Diversas ferramentas podem ser usadas na criação, validação e testes de distintos assuntos e pontos dentro da disciplina de LFA, as quais podem ser encontradas em várias tecnologias, rodando via navegador ou em ambiente desktop. Isso permite com que alunos e professores de diferentes cenários possam se beneficiar do auxílio gerado por esse tipo de ferramenta durante o ensino e o aprendizado dos conceitos abordados durante disciplinas que envolvam Linguagens Formais e Autômatos.

Tendo em vista que as disciplinas relacionadas aos tópicos abordados neste trabalho normalmente são de difíceis absorções, representando em alguns cursos os primeiros contatos mais intensos dos alunos de graduação com formalismos aplicados diretamente aos conceitos computacionais e que tais disciplinas são a raiz de diversos conteúdos abordados ao longo do curso de graduação em Ciência da Computação, como Compiladores e Teoria da Computação, a aplicação de ferramentas que possibilitem melhor fixação faz-se essencial para uma melhor experiência na jornada desses discentes.

O uso de ludicidade, interfaces e mecanismos que aumentem a interatividade ao mesmo tempo em que diminuem a dificuldade de assimilação de tópicos complexos, como é o caso dos conteúdos de Linguagens Formais e Autômatos, permite melhor compreensão e um foco ampliado no real conhecimento e nas diversas aplicações desse conteúdo de maneira simples e prática.

Ao passo que a formatação versátil em ferramentas que permitam o uso de código para representar os autômatos de maneira gráfica sem os entraves de ferramentas de edição ou inserções manuais de símbolos facilita o processo de produção escrita e estudo da disciplina, permitindo ao autor a dedicação de seu tempo ao conteúdo e não necessariamente ao manejo de um editor de texto ou imagens.

Como trabalhos futuros pretende-se testar tais ferramentas na disciplina de LFA nos cursos de graduação e do mestrado em Ciência da Computação da Universidade Estadual de Londrina, seguindo critérios descritos por Brito Junior e Aguiar (2018). Essa etapa deve ser comparada também com os trabalhos de Rodger, Lim and Reading (2007) e Paul (2015).

A despeito dos testes ainda não terem sido realizados, este trabalho apresenta uma grande contribuição por reunir ferramentas para serem usadas de modo criterioso, a fim de não preterir o pensamento algébrico do aluno não prejudicando o alcance dos objetivos da matéria de Linguagens Formais e Autômatos.

Cabe ressaltar que para testar tais ferramentas é necessário partir não só do Termo de Consentimento Livre e Esclarecido dos alunos, mas também passar por um Comitê de Ética da universidade em questão para termos a licença para que isso seja feito. Por isso, almejamos, inicialmente, disseminar esses conhecimentos teóricos das ferramentas levantadas reunindo em um só artigo em um evento da Sociedade Brasileira de Computação, para que, cumprindo-se os requisitos acima, possa ser aplicada a prática com os alunos colhendo suas impressões, experiências, etc. para uma publicação futura.

Referências

Alkmin, G. P. e Mello, B.A. (2010). Ferramenta de apoio às fases iniciais do ensino de Linguagens Formais e Autômatos e Compiladores. *Anais do XXII Simpósio Brasileiro de Informática na Educação*. Sociedade Brasileira de Computação, João Pessoa. p.1-4.

Barbosa C. R. S. C. Bonidia R. P. e Coelho Neto, J. (2019). Flex, JFlex e GALS: Ferramentas de Apoio ao Ensino de Compiladores. *Anais do XXVII Workshop sobre Educação em Computação*. Sociedade Brasileira de Computação, Belém. p.176-187.

Barbosa C. R. S. C., Faria, C. R. e Campano Junior, M. M. (2021). Análise de Ferramentas de Compiladores em Ambientes Virtualizados. In *Revista Brasileira de Informática na Educação*, v.29. p.1262-1290. RBIE.

Brito Junior, O. e Aguiar, Y. P. C. A. (2018). Taxonomia de critérios para avaliação de software educativo – TaCASE. *Anais do XXIX Simpósio Brasileiro de Informática na Educação*. Sociedade Brasileira de Computação, Fortaleza. p.298-307.

Burch, C.(2006). “Autosim”, http://www.cburch.com/proj/autosim/, June.

Campano Junior, M. M., Barbosa, C. R. S. C. Faria, C. R. e Felinto A. S. (2019). Um merge entre Máquina de Turing e Operações Matemáticas em Binário no Ensino de Linguagens Formais e Autômatos. Vol. 15. *Anais do XXIV Congresso Internacional de Informática Educativa*. J. Sanchéz (Ed.), Santiago do Chile. p.78-83.

Dickerson, K. (2021). "Automaton Simulator", https://automatonsimulator.com/, June.

Doty, D. (2020). "Automaton Simulator", https://web.cs.ucdavis.edu/~doty/automata/, June.

Hopcroft J. E., Motwani R. and Ullman, J. D. (2001). *Introduction to automata theory, languages, and computation*. Addison Wesley, 2rd edition, https://www-2.dc.uba.ar/staff/becher/Hopcroft-Motwani-Ullman-2001.pdf, June.

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

JFLAP. (2018). "JFLAP 7.0 Tutorial". https://www.jflap.org/tutorial/, July.

Menezes, P. B. (1998). *Linguagens Formais e Autômatos*. Porto Alegre: Instituto de Informática da UFRGS e Editora Sagra Luzatto, 2ª edição.

Paul, J. (2015). Using JFLAP to engage students and improve learning of computer science theory: tutorial presentation. In *Journal of Computing Sciences in Colleges*, v.26, pages 145-148.

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

Ramos, M. V. (2009). Ensino de linguagens formais e autômatos em cursos superiores de computação. In *Revista de Computação e Tecnologia da PUC-SP*. v.1, n.1, pages 22-34.

Rodger, S. H, Bilska, A. O., Leider, K. H., Procopiuc, M., Procopiuc, O., Salemme, J. R. and Edwin, T. (1997). A collection of tools for making automata theory and formal languages come alive. In *Association for Computing Machinery.* v.29, n.1, pages 15-19.

Rodger, S. H., Lim, J. and Reading, S. (2007). Increasing interaction and support in the formal languages and automata theory course. In *ACM SIGCSE Bulletin*, v.39, n.3, pages 58-62.

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

Wallace, E. (2010). "Finite State Machine Designer", https://www.cs.unc.edu/~otternes/comp455/fsm_designer/, July.

White, T. (2006). "JFAST – a Java Automata Simulator", http://jfast-fsm-sim.sourceforge.net/, May.

Zuzak, I. (2017). "FSM Simulator", https://ivanzuzak.info/noam/webapps/fsm_simulator/, June.