

# Language Emulator, uma ferramenta de auxílio no ensino de Teoria da Computação

Luiz Filipe Menezes Vieira<sup>1</sup>, Marcos Augusto Menezes Vieira<sup>1</sup>, Newton José Vieira<sup>1</sup>

<sup>1</sup> Departamento de Ciência da Computação - Universidade Federal de Minas Gerais (UFMG) - Av. Antônio Carlos, 6627 - 31270-010, Belo Horizonte, MG, Brasil

{lfvieira, mmvieira, nvieira} @dcc.ufmg.br

**Abstract.** *During a master degree class, we developed a tool to help students to understand the concepts of Automata Theory. It was denominated Language Emulator and has been used by undergraduate students to help them to learn the ideas of Automata Theory. Language emulator is a software developed in Java that allows the manipulation of regular expressions, regular grammar, deterministic finite automata, non-deterministic finite automata, non-deterministic finite automata with lambda transitions, Moore and Mealy machines. Using this software tool was of great acceptance among students, leading to over 95% of approval in a recent opinion research.*

**Resumo.** *Durante o curso de pós-graduação, foi desenvolvida uma ferramenta para ajudar os estudantes a compreender os conceitos da teoria da computação. Ela foi denominada Language Emulator e tem sido usada por estudantes da graduação com o intuito de ajudar-lhes a aprender as idéias da teoria dos autômatos. O Language Emulator é um software desenvolvido em Java na UFMG que permite a manipulação de expressões regulares, gramáticas regulares, autômatos finitos determinísticos, autômatos finitos não-determinísticos, autômatos finitos não-determinísticos com transições lambda, máquinas de Moore e de Mealy. O uso da ferramenta foi de grande aceitação entre os estudantes, chegando a 95% de aprovação em uma pesquisa recentemente realizada com os alunos.*

**Keywords:** *Teoria da computação, ferramenta, ensino.*

## 1. Introdução

O curso de teoria da computação é tradicionalmente ensinado sem nenhuma ajuda de ferramentas de software, limitando-se ao uso de tarefas que não necessitam do uso de um computador. Todos os exercícios envolvem o uso do lápis e papel e estes podem ser um inconveniente neste curso. Uma possível solução seria ter alguns exercícios que envolvessem o uso de ferramentas de software. Como mostrado por [1], há duas razões principais porque muitos estudantes têm mais dificuldade com cursos da área de teoria da computação do que cursos de outras áreas. A primeira é que disciplinas da teoria requerem um conhecimento maior em matemática do que a maioria das outras disciplinas de Ciência da Computação. A segunda razão é o fato dos estudantes não recebem uma correção imediata ao trabalharem em problemas usando lápis e papel.

Estes efeitos podem ser reduzidos pelo uso de ferramentas de software, tais como o Language Emulator que permite que os estudantes interajam com a teoria da computação e com autômatos. Neste artigo, nós descrevemos os trabalhos relacionados na seção 2, a funcionalidade do Language Emulator na seção 3, a apresentação gráfica na seção 4. A seção 5 descreve como esta ferramenta pode ser usada em uma sala de aula e o estudo de caso da ferramenta em uma sala da graduação. E, finalmente, na seção 7 as conclusões e trabalhos futuros.

## **2. Trabalhos relacionados**

Na Ciência da Computação, para muitos assuntos existem ferramentas para ajudar os estudantes na aprendizagem. Muitas ferramentas, incluindo [2], foram desenvolvidas para criar animações dos algoritmos e estas podem ser usadas no ensino.

JFLAP[1] (Java Formal Languages and Automata Package) é uma excelente ferramenta que pode ser usada na disciplina de fundamentos da teoria da computação, mais especificamente, na parte de teoria dos autômatos. Infelizmente, não é internacionalizada, aceita somente o inglês, e não suporta a transição entre os vários tipos de autômatos que trabalha.

Deus Ex Machine [3] foi desenvolvido por Nick Savoio e compreende simulações de sete modelos da computação. Foi desenvolvido para acompanhar o livro “Models of Computation and Formal Languages” de R. Gregory Taylor. Atualmente só há versões para a plataforma Windows.

Turing’s World [4] é uma ferramenta gráfica que permite especificar máquinas de Turing desenhando o diagrama de estados e as transições. Ela permite visualizar graficamente as operações na máquina de Turing.

Webworks Laboratory [5] apresenta um conjunto de projetos na área de Fundamentos da Teoria da Computação. Se destaca o Animating the Theory of Computing, uma ferramenta desenvolvida para permitir que os estudantes aprendessem através de animações de automâtos finitos.

Ganimal [6] é um projeto de software para aprendizado de compiladores que gera animações de máquinas abstratas. A máquina pode ser escolhida de acordo com o tipo de linguagem: imperativa, funcional ou a pilha.

The Java Computability Toolkit [7] é um ambiente gráfico para criar e simular automâtos finitos determinísticos e não-determinísticos e máquinas de Turing. Ele foi desenvolvido em Java, e permite minimizar o AFD e converter um AFN em um AFD.

## **3. A Ferramenta**

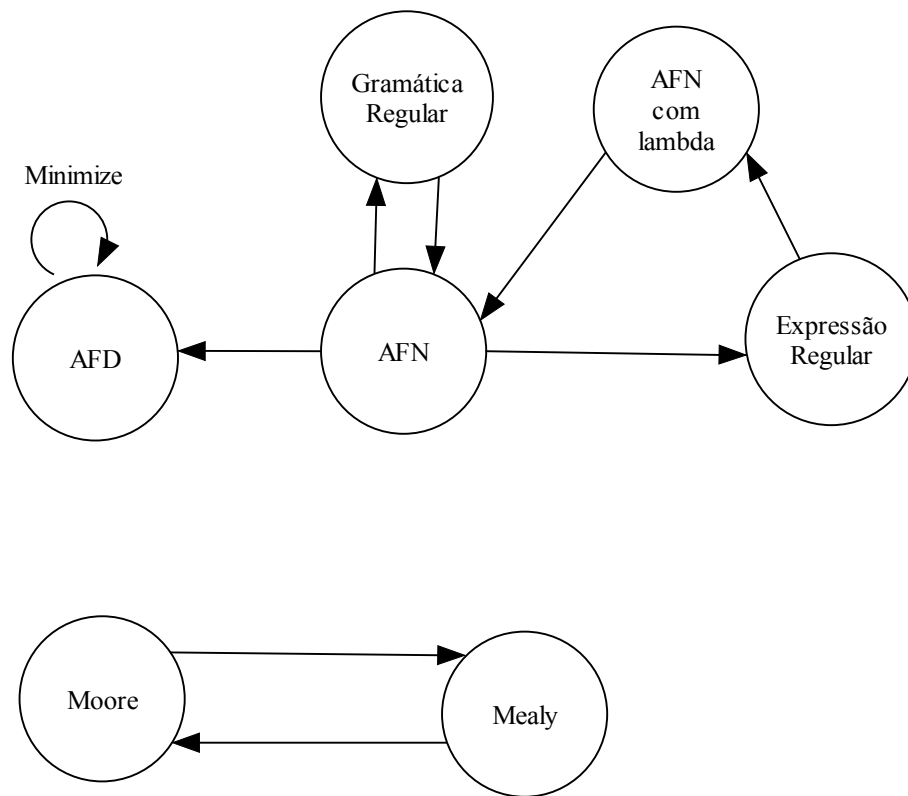
O Language Emulator foi construído para trabalhar com as formas mais usuais de descrição de linguagens regulares. Os sete formatos suportados são:

- Expressão Regular.
- Gramática Regular.
- Autômato finito determinístico (AFD).
- Autômato finito não determinístico (AFN).

- ❑ Autômato finitos não determinístico com transições lambda (AFN $\lambda$ ).
- ❑ Máquina de Moore.
- ❑ Máquina de Mealy.

E adicionalmente as seguintes funcionalidades:

- ❑ Minimizar autômatos finitos determinísticos.
- ❑ Transformar autômato finito não-determinístico em autômato finito determinístico.
- ❑ Transformar autômato finito não-determinístico com transições lambda em autômato finito não-determinístico.
- ❑ Transformar uma gramática regular em um autômato finito não-determinístico.
- ❑ Transformar autômato finito não-determinístico em uma gramática regular.
- ❑ Transformar uma expressão regular em um autômato finito não-determinístico com transição lambda.
- ❑ Transformar autômato finito não-determinístico em uma expressão regular.
- ❑ Derivar todas as palavras até um tamanho n que pertençam a uma gramática regular.
- ❑ Determinar se uma palavra pertence a uma gramática regular.
- ❑ Determinar se uma palavra pertence à linguagem especificada por um autômato finito determinístico.
- ❑ Determinar se uma palavra pertence à linguagem especificada por uma expressão 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.
- ❑ Determinar se uma palavra pertence à uma máquina Mealy ou de Moore.
- ❑ União, interseção e diferença de dois AFDs.
- ❑ Complemento de um AFD.
- ❑ União das diferenças entre dois AFDs.  $A_F = (A_1 - A_2) \cup (A_2 - A_1)$
- ❑ Salvar e Carregar AFDs.
- ❑ Exportar o AFDs para um arquivo no formato utilizado pela ferramenta GraphViz[8]. Esta ferramenta recebe uma descrição textual de um grafo e gera uma representação visual do mesmo.

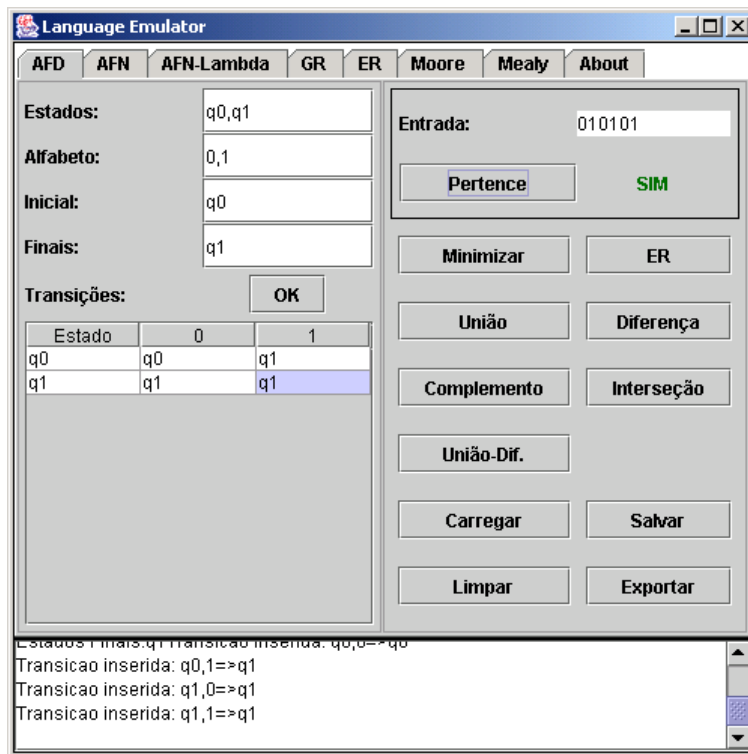


**Figura 1 Transformação entre formatos de linguagem regulares**

A ferramenta também mostra os passos intermediários do algoritmo, imprimindo os valores de cada etapa da execução.

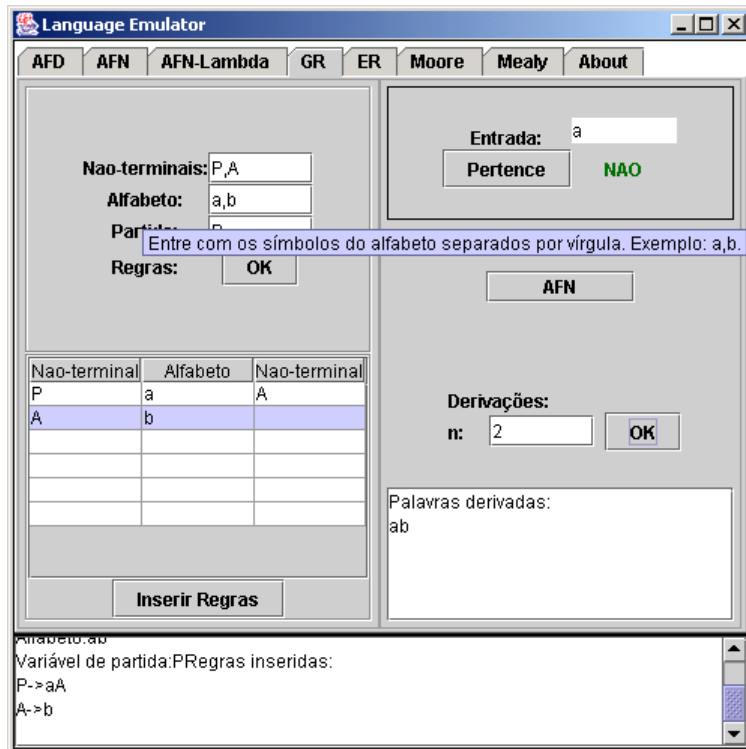
#### **4. Interface**

A linguagem de programação escolhida para criar a ferramenta de ajuda para a teoria dos autômatos foi Java, simplesmente porque funciona em toda plataforma que suporta a máquina virtual Java, o que é comum em um ambiente educacional. Ela foi projetada usando Java Swing que é o padrão de interface da plataforma Java. Ela foi desenvolvida inicialmente para o português, mas é possível internacionalizar a ferramenta, mudando para a língua desejada, como, por exemplo, inglês.



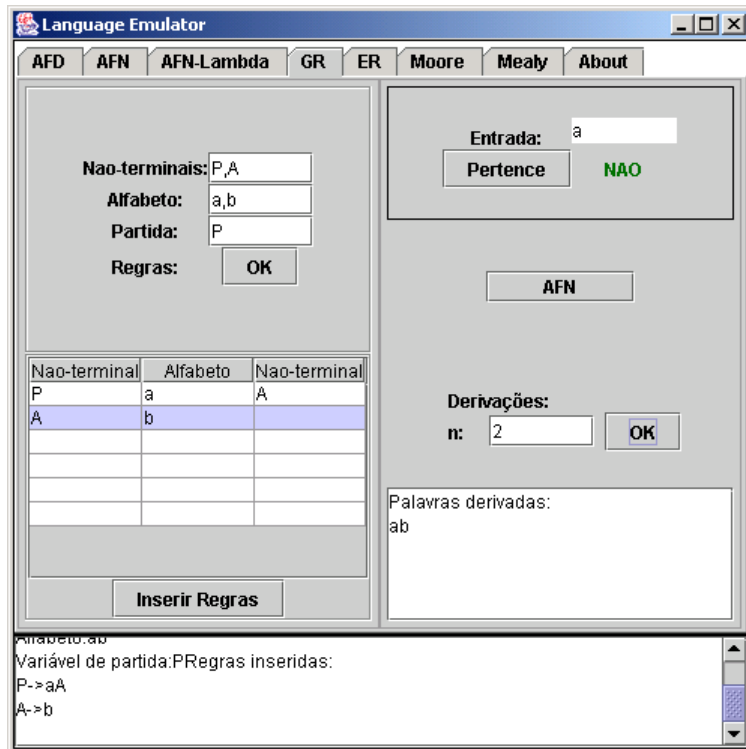
**Figura 2 Autômatos finitos determinísticos no Language Emulator.**

Há sete abas que o usuário pode escolher na relação que permite construir qualquer um dos sete formatos de linguagem regular suportadas no Language Emulator: autômatos finitos determinísticos, autômatos finitos não-determinísticos, autômatos finitos não-determinísticos com transições lambda, gramática regular, expressão regular, máquinas Moore e máquinas Mealy. Para cada formato existe uma interface apropriada. Esta pode ser visualizada quando o usuário seleciona a aba com o nome do formato. Ela permite acessar as funcionalidades associadas ao formato. Por exemplo, os autômatos finitos determinísticos podem ser minimizados ou gerar a expressão regular que reconhece a mesma linguagem que eles. Estas duas funcionalidades podem ser acessadas pelo usuário através dos botões Minimizar e ER presentes na tela dos autômatos finitos determinísticos, que é mostrada na Figura 2.



**Figura 3 ToolTips estão presentes no Language Emulator.**

A Figura 3 mostra uma tela do Language Emulator que mostra a presença de ToolTips. Eles estão presentes na ferramenta, o que torna mais intuitivo o uso desta.



**Figura 4 Uma Gramática Regular no Language Emulator.**

A Figura 4 mostra a construção de uma gramática regular. O usuário introduz primeiramente os não terminais, o alfabeto e então o não terminal inicial, de acordo com a definição matemática de uma gramática regular. Então acione o botão OK para iniciar a tabela onde podem ser inseridas as regras. Clicando sobre inserir regras, a gramática é construída. É possível ver se há uma palavra que pertença ou não à linguagem regular que é reconhecida pela gramática regular, ou construir um autômato finito não determinístico (AFN) que reconheça a mesma linguagem que a gramática.

Com este AFN é possível descobrir se uma palavra é reconhecida pelo autômato, modificá-lo, inserindo novos símbolos no alfabeto ou mudando os estados iniciais e/ou finais. Também é possível gerar uma expressão regular que represente a linguagem aceita pelo autômato e, inclusive, gerar a gramática ou um autômato finito determinístico (AFD).

No Language Emulator, as funcionalidades que permitem manipular um AFD estão descritas na seção sobre a ferramenta. É possível modificar o autômato em qualquer um dos seus componentes (estados, alfabeto, estado inicial, estados finais e transições). Este pode ser minimizado ou pode-se gerar a expressão regular equivalente.

O Language Emulator pode ser utilizado para verificar se uma palavra pertence ou não a linguagem equivalente a uma expressão regular e/ou gerar o  $AFN\lambda$  equivalente a uma expressão regular. Esta também pode ser editada.

Na ferramenta, uma representação no formato de um  $AFN\lambda$  apresenta quase as mesmas funcionalidades que uma no formato de um AFN. É possível descobrir se uma palavra é reconhecida pelo autômato, modificá-lo e pode ser gerado um AFN equivalente.

## **5. Auxílio no ensino**

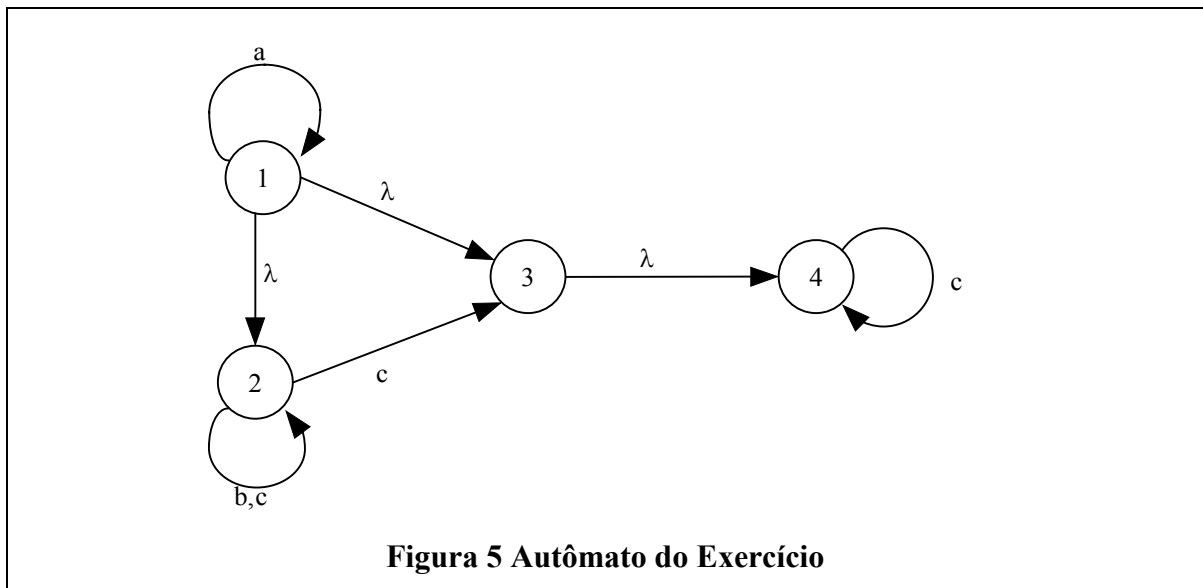
Junto com a teoria, alguns exercícios práticos podem ser atribuídos. A cada parte da teoria abordada, como autômatos finitos determinísticos, uma tarefa pode ser dada envolvendo o uso da ferramenta. Este modelo torna o assunto mais divertido para aprender.

### **Caso de Uso - UFMG**

Durante o primeiro semestre de 2002, na aula da disciplina Fundamentos da Teoria da Computação na UFMG (Universidade Federal de Minas Gerais) a ferramenta foi utilizada. Seguindo o modelo sugerido acima, após o ensino de  $AFN\lambda$ , AFN, AFD e ER foi proposto o seguinte exercício:

### **Exemplo de exercício**

Seja o  $AFN\lambda$ :



- (a) Faça no papel a tabela de transições.
- (b) Construa um AFN $\lambda$  na ferramenta.
- (c) Faça no papel o AFN equivalente.
- (d) Utilizando a ferramenta, converta o AFN $\lambda$  para um AFN.
- (e) Faça no papel um AFD equivalente.
- (f) Utilizando a ferramenta, converta o AFN para um AFD.
- (g) Faça no papel a minimização do AFD.
- (h) Utilizando a ferramenta, minimize o AFD.
- (i) Construa uma expressão regular que denote a linguagem reconhecida.
- (j) Utilizando a ferramenta, converta o AFD para uma ER.

Com a aplicação do exercício para ser resolvido em casa, os alunos puderam corrigir os próprios exercícios e aprender aonde estavam errando.

### Auxílio ao professor

A ferramenta além de ser utilizada pelos alunos, também pode ser utilizada pelo professor para corrigir exercícios. Seja  $L_1$  a linguagem reconhecida pelo autômato  $A_1$  descrito pelo aluno e seja  $L_2$  a linguagem reconhecida pelo autômato correto  $A_2$ .

A ferramenta permite calcular o autômato resultante da união, interseção ou diferença entre dois autômatos finitos determinísticos. Além disso, foi incorporado a ferramenta a operação de união das diferenças de dois AFDs. Esta operação é útil, conforme mostraremos a seguir.

Utilizando a própria ferramenta, selecionando o botão União das Diferenças na interface dos AFDs, podemos calcular:

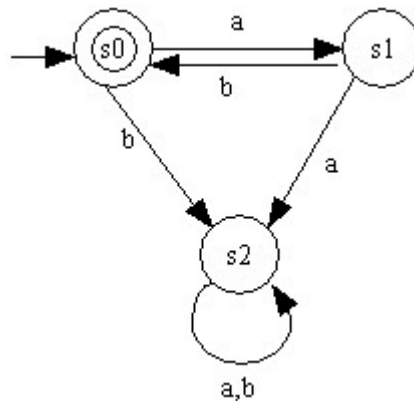
$$A_3 = (A_1 - A_2) \cup (A_2 - A_1)$$

A linguagem reconhecida pelo autômato  $A_3$  será vazia se o autômato estiver correto. E gerando uma palavra de  $A_3$  é possível encontrar um exemplo de palavra que torna a solução incorreta.

Exemplo:

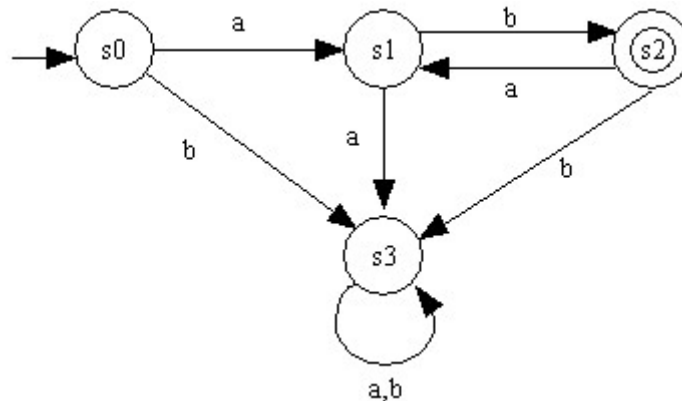
Seja a linguagem  $L_2 = \{ab\}^*$ .

O autômato correto  $A_2$  seria:



**Figura 6 AFD que reconhece  $L_2$**

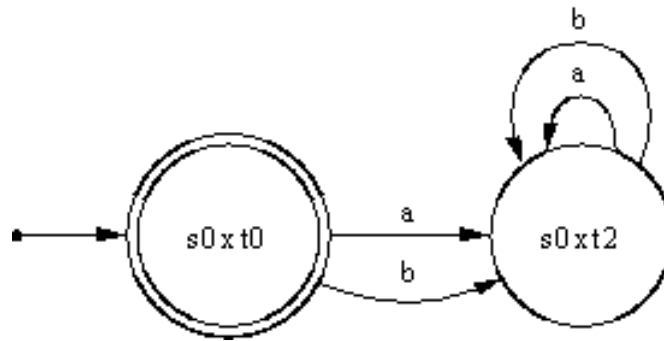
Suponha que o autômato  $A_1$  do aluno seja:



**Figura 7 AFD desenvolvido por um aluno.**

O autômato  $A_1$  reconhece  $ab, abab, ababab, \dots, (ab)^n$ . Porém ele não está correto. Uma maneira rápida de mostrar isto é encontrar um contra-exemplo. No nosso caso, o autômato  $A_1$  não reconhece  $\lambda$  e portanto, não reconhece a linguagem  $L_2$ .

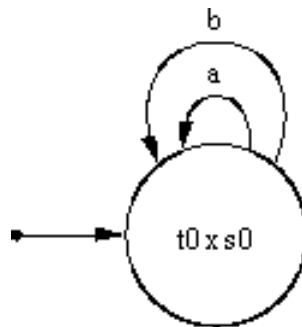
Utilizando a função de diferença entre dois AFDs da ferramenta, calculamos  $A_2 - A_1$  para encontrarmos as palavras que não são corretamente reconhecidas pelo autômato  $A_1$ .



**Figura 8** Autômato resultante de  $A_2-A_1$ .

O autômato da Figura 8 reconhece apenas  $\lambda$ .

E também calculamos  $A_1-A_2$  para encontrar as palavras que são incorretamente reconhecidas pelo autômato  $A_1$



**Figura 9** Autômato resultante de  $A_1-A_2$ .

O autômato da Figura 9 reconhece o conjunto vazio. Observe que não há estados finais.

Figura 8 e Figura 9 foram geradas com o GraphViz[8] após utilizar a função exportar do Language Emulator.

Outra maneira de se verificar a corretude do autômato do aluno seria gerar a expressão regular equivalente ao autômato e verificar se ela representa a mesma linguagem que o autômato correto. No exemplo, o Language Emulator retornou  $ER_1=(ab(ab)^*)$  para o autômato do aluno. Observa-se que  $\lambda$  não pertence a linguagem equivalente a  $ER_1$ .

### **Avaliação da ferramenta**

Como era a primeira vez que o Language Emulator estava sendo utilizado em uma sala de aula, os estudantes da disciplina participaram também no processo de avaliação da ferramenta. Quase 95% dos estudantes responderam que o uso do Language Emulator ajuda no processo de aprender a teoria dos autômatos. Eles deram também algumas sugestões que podem ser feitas como trabalho futuro para melhorar a ferramenta e sua capacidade de ajudar no ensino. Uma das vantagens citadas pelos estudantes é o fato que eles podem fazer os exercícios com lápis e papel e depois usar o Language Emulator para verificar se o fizeram corretamente.

## 6. Conclusão e Trabalhos Futuros

O Language Emulator tem sido utilizado no processo de ensinar teoria dos autômatos. É possível o estudante interagir com os formatos de linguagens regulares e receber um feedback imediato. O uso da ferramenta em um curso de computação teve um nível elevado de aprovação, concordando com a finalidade de tal ferramenta. Os estudantes gostaram muito do fato de poderem verificar se seu trabalho feito a lápis estivesse correto.

A ferramenta está em um estágio inicial e pode ser melhorada. Um trabalho futuro seria permitir o desenho dos estados e transições, tornando a entrada de dados gráfica.

## Agradecimentos

Este trabalho teve apoio do CNPq.

## Referências

- [1] Rodger, S.H. Using JFLAP to interact with Theorems in Automata Theory. SIGCCE (1999), 336-340.
- [2] P. Gloor, AACE – Algorithm Animation for Computer Science Education. Workshop on Visual Languages, p. 25-31 ,1992.
- [3] Deus Ex Machine, web site: <http://www.ics.uci.edu/~savoIU/dem/>
- [4] Turing's World, web site: <http://www-csli.stanford.edu/hp/Logic-software.html>
- [5] Webworks Laboratory ,web site: <http://www.cs.montana.edu/webworks>
- [6] Ganimal, web site: <http://rw4.cs.uni-sb.de/~ganimal/>
- [7] Robinson, M.B., Hamshar, J.A., Novillo, J.E., Duchowski, A.T. *A Java-based Tool for Reasoning About Models of Computation Through Simulating Finite Automata and Turing Machines*. 30th Annual ACM SIGCSE Symposium (Special Interest Group on Computer Science Education), New Orleans, Louisiana, March 24-28, 1999.
- [8] E. Koutsofios and S. C. North. Drawing Graphs with dot. AT&T Bell Laboratories, Murray Hill NJ, U.S.A., October 1993.