Rafael Cardoso da Silva

Um sistema para auxiliar na aprendizagem da disciplina Linguagens Formais e Autômatos

Santo André - SP, Brasil

2016

Rafael Cardoso da Silva

Um sistema para auxiliar na aprendizagem da disciplina Linguagens Formais e Autômatos

Projeto de pesquisa em conformidade com as normas ABNT apresentado à Disciplina de Projeto Dirigido.

Universidade Federal do ABC – UFABC

Bacharelado em Ciências e Tecnologia

Projeto Dirigido

Santo André - SP, Brasil

2016

Sumário

Resumo

Resolver exercícios é fundamental para um aluno fixar os conceitos apresentados em aula. Por outro lado, ter seus exercícios corrigidos também é muito importante, para que ele possa avaliar o seu aprendizado. Na UFABC, a disciplina de Linguagens Formais e Autômatos contempla vários exercícios que admitem infinitas respostas, o que torna a correção deles praticamente impossível, principalmente quando as turmas são grandes. O objetivo deste projeto é a criação e implementação de um sistema para aplicação e correção automática de exercícios envolvendo autômatos finitos determinísticos. Através do estudo de métodos e algoritmos presentes na literatura, será possível implementar o teste de equivalência entre o autômato-resposta do aluno e o autômato-gabarito previamente armazenado no banco de dados. E ao final do projeto, o sistema será utilizado em caráter experimental numa turma da UFABC da disciplina de Linguagens Formais e Autômatos, afim de testar a sua qualidade. E ao final da disciplina, a nota que os alunos obterão ao revolver os exercícios do sistema ajudarão a compor o conceito final de cada um na disciplina.

Palavras-chave: autômato finito, equivalência de autômatos, minimização de autômato, programação para web.

1 Introdução

Resolução de exercícios é uma atividade fundamental para que um aluno possa praticar o que foi apresentado em aula. A correção dos exercícios também é importante para que o aluno possa avaliar seu aprendizado. Para turmas grandes, torna-se inviável corrigir rapidamente todos os exercícios propostos e devolvê-los em tempo hábil aos alunos. Na disciplina MC3106 – Linguagens Formais e Autômatos – vários exercícios da parte inicial da disciplina pedem como resposta um autômato finito determinístico. Tais exercícios são particularmente trabalhosos de se corrigir porque cada um deles admite infinitas respostas corretas e, por esse motivo, sua correção se torna exaustiva para o monitor ou docente responsável.

Com o intuito de agilizar essa tarefa, este projeto propõe o desenvolvimento de um sistema web que permita ao aluno inserir seu autômato-resposta por meio de uma interface intuitiva e fácil de ser utilizada e obter um *feedback* quase que imediatamente, um tempo drasticamente reduzido se comparado à correção manual. Acreditamos que um sistema como esse será de grande valia para o aprendizado dos alunos. Por outro lado, para o docente responsável, facilitará o processo de composição do conceito final na disciplina de Linguagens Formais e Autômatos.

2 O Principal Problema

Um *autômato finito determinístico* (também chamado de AFD ou máquina de estados) consiste de um conjunto de estados não vazio Q, um alfabeto Σ, um estado inicial $s \in Q$, um conjunto de estados de aceitação $F \subseteq Q$ e uma função de transição $\delta\colon Q \times \Sigma \to Q$. É comum estender a função de transição para uma função $\hat{\delta}\colon Q \times \Sigma^* \to Q$ definida indutivamente por

(i) $\hat{\delta}(q, \varepsilon) = q$ para todo estado $q \in Q$;

(ii) $\hat{\delta}(q, \sigma) = \delta(q, \sigma)$ para todo estado $q \in Q$ e símbolo $\sigma \in \Sigma$;

(iii) $\hat{\delta}(q, \sigma_1 \cdots \sigma_k) = \delta(\hat{\delta}(q, \sigma_1 \cdots \sigma_{k-1}), \sigma_k)$ para todo estado $q \in Q$ e símbolos $\sigma_1, \ldots, \sigma_k \in \Sigma$.

Todo autômato pode ser representado por um diagrama de estados. Considere o diagrama da Figura 1 abaixo.

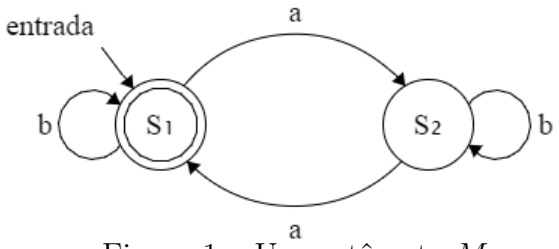

Figura 1 – Um autômato M_1.

No autômato da Figura 1, temos $Q = \{S_1, S_2\}$, $\Sigma = \{a, b\}$, $s = S_1$, $F = \{S_1\}$ e a função de transição é dada pela tabela a seguir.

δ	`a`	`b`
S_1	S_2	S_1
S_2	S_1	S_2

Tabela 1 – Tabela de transições do M_1.

Nesse exemplo, se o AFD M_1 é alimentado com a palavra **`aabab`**, ele irá começar no estado S_1 e irá percorrer os estados S_2, S_1, S_1, S_2, S_2 nesta ordem, e irá rejeitar a palavra dada (pois o último estado $S_2 \notin F$). Se, para uma outra palavra w, a simulação tivesse terminado em S_1 (que pertence a F) diríamos que o autômato aceita a palavra w. Mais formalmente, se $M_1 = (Q, \Sigma, \delta, s, F)$ é um autômato finito determinístico, dizemos que M_1 *aceita* w se $\hat{\delta}(s, w) \in F$ e que M_1 *rejeita* w caso contrário.

A *linguagem reconhecida* por um autômato $M_1 = (Q, \Sigma, \delta, s, F)$ é o conjunto de palavras que são aceitas por esse autômato, ou seja, é definida pelo conjunto $L(M_1) = \{w \in \Sigma^*\colon \hat{\delta}(s, w) \in F\}$. No exemplo da Figura 1 acima, a linguagem reconhecida por por M_1 é $L(M_1) = \{w \in \Sigma^* : w$ tem número par de símbolos `a`$\}$.

Vários tipos de exercícios pedem por uma resposta que é um autômato. Por exemplo:

(i) construir um autômato que reconheça a linguagem regular dada por um certo conjunto de palavras descrito matematicamente;

(ii) transformar um autômato finito não determinístico num determinístico equivalente;

(iii) criar um autômato que reconheça a mesma linguagem descrita por uma expressão regular dada;

(iv) construir o menor autômato finito determinístico que reconheça uma certa linguagem regular.

Cada um destes tipos de exercícios possui centenas de casos particulares que os alunos podem fazer para praticar os conceitos de autômatos abordados na disciplina. Tais exercícios podem, por exemplo, ser encontrados abundantemente no livro *Introdução à Teoria da Computação* de M. Sipser (SIPSER, 2012). Outras referências importantes da disciplina que possuem exercícios semelhantes são (HOPCROFT; MOTWANI; ULLMAN, 2006) e (LINZ, 2011). Em todos os casos acima, contudo, há uma infinidade de respostas corretas para cada exercício. Por exemplo, considere o diagrama abaixo.

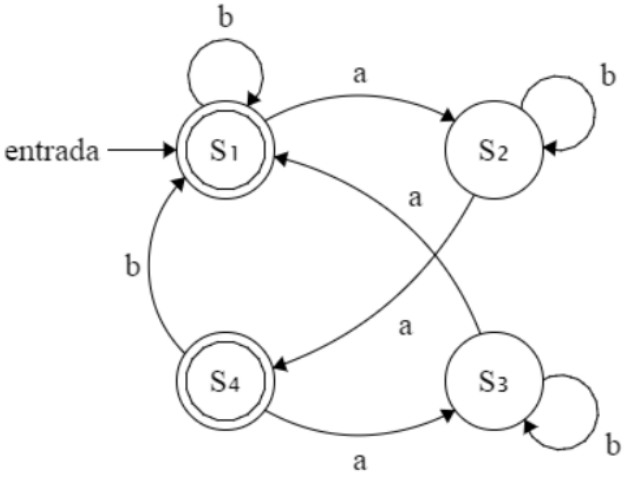

Figura 2 – Um autômato M_2.

Apesar de visivelmente maior e mais complexo que M_1, este autômato reconhece a mesma linguagem que o autômato M_1 da Figura 1. É possível demonstrar que, para cada linguagem regular, existem infinitos autômatos que a reconhecem. Como decidir então se o autômato-resposta dado por um aluno reconhece a linguagem pedida? No caso do sistema que iremos desenvolver isso se reduz ao seguinte problema:

Problema: Decidir se dois autômatos finitos determinísticos reconhecem a mesma linguagem.

2.1 Breve descrição do método

Determinar se dois autômatos finitos e determinísticos reconhecem linguagens diferentes é um problema bem resolvido. A idéia fundamental é tentar encontrar uma palavra w que é aceita por um autômato e rejeitada pelo outro. Se tal palavra puder ser encontrada, então as linguagens reconhecidas por esses autômatos são distintas. Caso tal palavra não exista, os autômatos reconhecem a mesma linguagem.

Mais formalmente, dizemos que dois estados q_1 e q_2 (no mesmo autômato ou não) são *equivalentes* se, para toda palavra $w \in \Sigma^*$ vale que $\hat{\delta}(q_1, w) \in F$ se e somente se $\hat{\delta}(q_2, w) \in F$. Dois autômatos serão, portanto, equivalentes se e somente se seus respectivos estados iniciais forem equivalentes no sentido que acabamos de definir.

Esse problema está intimamente relacionado com o problema de minimização de autômatos finitos determinísticos. Os primeiros algoritmos desenvolvidos para testar a equivalência de autômatos resolviam também o problema da minimização. Eles apareceram em (HUFFMAN, 1954) e (MOORE, 1956) e podem ser implementados sem muito esforço em tempo $O(n^2)$, onde n é o número total de estados. O algoritmo mais eficiente conhecido para minimização de autômatos (HOPCROFT, 1971) executa em tempo $O(n \log n)$. Posteriormente Hopcroft e Karp (HOPCROFT; KARP, 1971) desenvolveram um algoritmo linear para testar a equivalência de autômatos.

Neste projeto será implementado todos os algoritmos envolvidos na discussão acima como parte de um sistema maior que descrevemos na próxima seção.

2.2 Breve descrição do sistema

Como mencionamos anteriormente, a proposta deste projeto é a criação de um site que disponibilize uma lista de exercícios aos usuários – que serão os alunos da disciplina MC3106 – e que permita a correção automática desses exercícios. Esse site será semelhante aos sistemas de correção automática utilizados em eventos de maratona de programação, onde o competidor submete sua respostar para resolução de algum problema proposto, e o sistema analisa a resposta recebida e dá um retorno da submissão. Em nosso sistema, por meio de uma interface intuitiva, o usuário poderá ver a lista de exercícios e, para cada um deles, poderá consultar o seu estado: se ainda não foi resolvido, se há um rascunho de resposta salvo, se uma resposta já foi submetida, o número de submissões, e qual foi o conceito ou *feedback* devolvido ao aluno em cada submissão.

Ao clicar num exercício especifico, o aluno verá uma página com o título do exercício, o corpo do texto com a descrição do exercício que o ele deve resolver, e uma área dedicada para a confecção da resposta-autômato. Para a confecção desta resposta, usaremos como front-end um JavaScript desenvolvido e disponibilizado por Evan Wallace (WALLACE,

2010) sob a licença MIT (MIT, 1988) que, de maneira muito simples e intuitiva, permite a construção de diagramas de máquina de estados.

O aluno então resolve o exercício da maneira tradicional e concebe um autômato como possível resposta. Em seguida ele poderá, com o auxílio do mouse e do teclado, inserir o diagrama de seu autômato no espaço designado, alterando-o quantas vezes achar necessário. O aluno tem a opção de salvar seu trabalho para continuar mais tarde e também de submetê-lo como resposta. Quando ele decidir submeter, a estrutura do autômato construído pelo aluno nesta interface será validada (por um script que decide se o diagrama construído é de fato um AFD) e será enviada para o servidor.

No servidor, será executado o programa que será implementado pelo bolsista com o algoritmo de Hopcroft e Karp para testar a equivalência de autômatos finitos determinísticos. Se o autômato que o aluno construiu for equivalente ao AFD-gabarito armazenado no banco de dados, então o aluno terá resolvido o exercício corretamente. Se não forem equivalentes, podemos encontrar uma palavra que distingue os dois autômatos e enviar essa palavra ao aluno para que ele tenha um *feedback* e um certificado de que sua resposta está incorreta. Examinando essa palavra com mais cuidado, o aluno poderá então perceber a razão pela qual seu autômato não reconhece a linguagem pedida. Ele poderá elaborar uma outra resposta-autômato e fazer uma nova submissão, de acordo com a data final para a submissão de cada exercício determinado previamente pelo administrador.

Do ponto de vista do professor da disciplina, que será um administrador do sistema, pretendemos implementar as seguintes funcionalidades: criação e gerenciamento de turmas, o gerenciamento de listas compostas de exercícios, gerenciamento do banco de questões para compor as listas e, por fim, visualização e exportação da planilha final de notas.

Para a implementação deste sistema, serão necessários conhecimentos das disciplinas Programação para WEB e Linguagens Formais e Autômatos, que o aluno proponente deste projeto já cursou na UFABC. Se houver tempo, confeccionaremos ainda outras ferramentas para correção automática de outras classes de exercícios que a disciplina engloba e faremos a integração com o sistema.

3 Objetivos

Os Objetivos Gerais deste projeto são os seguintes:

- Incentivar o gosto do bolsista pela pesquisa aplicada.
- Desenvolver a maturidade do aluno em tópicos centrais de ciência da computação.
- Incentivar e aprimorar as habilidades de programação do candidato.
- Envolver o aluno em um projeto com fins de ensino e aprendizagem.

Os Objetivos Específicos são que o aluno implemente:

- o algoritmo de Moore para minimização e equivalência de AFDs;
- o algoritmo de Hopcroft para minimização de AFD;
- o algoritmo de Hopcroft e Karp para testar a equivalência de AFDs;

Todos os itens citados fazem parte da finalidade deste projeto de implementar o sistema descrito, algo que também envolve implementações de programas em JavaScript e C/C++ além de outras linguagens e tecnologias envolvidas na interface cliente/servidor como HTMl, PHP, MySQL, etc.

4 Metodologia

Esta Iniciação Cientifica será conduzida da seguinte forma. Num período inicial o aluno se preocupará em ler as seções relevantes das referências (SIPSER, 2012), (HOPCROFT; MOTWANI; ULLMAN, 2006) e (LINZ, 2011), a fim de criar familiaridade com o problema de equivalência, e abstrair os conceitos iniciais do algoritmo de minimização de autômatos finitos determinísticos. Será uma revisão de tópicos cobertos na disciplina de Linguagens Formais e Autômatos, que o aluno já cursou na UFABC.

Passada a fase inicial, o aluno aprofundará seus estudos nos artigos (HOPCROFT, 1971) e (HOPCROFT; KARP, 1971) que descrevem os algoritmos mais eficientes para minimização e equivalência de autômatos respectivamente. À medida que o aluno for conhecendo os melhores algoritmos para cada problema, ele irá implementar o sistema, aplicando os conceitos e os algoritmos estudados. Uma outra parte considerável do projeto será dedicada a elaboração e compilação de exercícios (populando um banco de dados e/ou possivelmente elaborando scripts que geram exercícios automaticamente) e aos testes do sistema para que ele possa ser usado na UFABC já com a primeira turma de MC3106 após o término deste projeto.

O orientador irá acompanhar seu progresso em reuniões semanais, quando o aluno esclarecerá suas dúvidas e o orientador poderá complementar a formação do bolsista explicando-lhe conceitos pertinentes que possivelmente não estejam cobertos nos livros e artigos indicados anteriormente, bem como apontar novos artigos relevantes sobre o assunto e indicar direções que devam ser seguidas na implementação do sistema.

5 Cronograma

O bolsista deverá dedicar a este projeto um mínimo de 20 horas por semana. O projeto irá se desenrolar conforme os itens abaixo:

1. Leitura da bibliografia (capítulos de livros e artigos citados).
2. Implementação dos algoritmos em C++.
3. Adequação do FSM Designer às necessidades deste projeto.
4. Elaboração do relatório parcial.
5. Desenvolvimento do sistema.

5.1. Levantamento de requisitos, análise e projeto do sistema.

5.2. Implementação do sistema.

6. Confecção dos exercícios no sistema.
7. Aplicação numa turma experimental.
8. Elaboração do relatório final.

O andamento do projeto se dará conforme a Tabela 2 abaixo (os números à esquerda indicam as tarefas enumeradas na seção anterior).

Tabela 2 – Cronograma

	ago	set	out	nov	dez	jan	fev	mar	abr	mai	jun	jul
1	x	x	x	x	x	x	x					
2	x	x	x	x								
3				x	x							
4					x	x	x					
5						x	x	x	x	x		
5.1						x						
5.2							x	x	x	x		
6									x	x		
7										x	x	x
8									x	x	x	x

6 Orçamento do Pesquisa

O Orçamento para a execução deste projeto pode ser vista pela tabela a abaixo.

Tabela 3 – Orçamento

CUSTEIO		
Serviços de Terceiro		
(5 meses)	Hospedagem do Site	R$ 74,50* (R$ 14,90 cada mês)
(1 ano)	Registro de Domínio	R$ 49,90*
TOTAL		R$ 124,40

* valores obtidos com base nos pacotes oferecidos no site LocaWeb[1].

[1] Disponível em <http://www.locaweb.com.br/hospedagem-de-sites/>

Referências

GAREY, M. R.; JOHNSON, D. S. *Computers and intractability*. [S.l.]: W. H. Freeman and Co., San Francisco, Calif., 1979. x+338 p. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences. ISBN 0-7167-1045-5. Nenhuma citação no texto.

HOPCROFT, J. An n log n algorithm for minimizing states in a finite automaton. In: *Theory of machines and computations (Proc. Internat. Sympos., Technion, Haifa, 1971)*. [S.l.]: Academic Press, New York, 1971. p. 189–196. Citado 2 vezes nas páginas 7 e 10.

HOPCROFT, J. E.; KARP, R. M. A linear time algorithm for testing equivalence of finite automata. *Technical report of Cornell University*, p. 71–114, 1971. Citado 2 vezes nas páginas 7 e 10.

HOPCROFT, J. E.; MOTWANI, R.; ULLMAN, J. D. Automata theory, languages, and computation. *International Edition*, v. 24, 2006. Citado 2 vezes nas páginas 6 e 10.

HUFFMAN, D. A. The synthesis of sequential switching circuits. I, II. *J. Franklin Inst.*, v. 257, p. 161–190, 275–303, 1954. ISSN 0016-0032. Citado na página 7.

LINZ, P. *An introduction to formal languages and automata*. [S.l.]: Jones & Bartlett Publishers, 2011. Citado 2 vezes nas páginas 6 e 10.

MIT. *The MIT Licence*. 1988. <http://opensource.org/licenses/MIT>. Citado na página 8.

MOORE, E. F. Gedanken-experiments on sequential machines. In: *Automata studies*. [S.l.]: Princeton University Press, Princeton, N. J., 1956, (Annals of mathematics studies, no. 34). p. 129–153. Citado na página 7.

SIPSER, M. *Introduction to the Theory of Computation*. [S.l.]: Cengage Learning, 2012. Citado 2 vezes nas páginas 6 e 10.

WALLACE, E. *Finite State Machine Designer*. 2010. <http://madebyevan.com/fsm/>. [Online; acessado 29 de agosto de 2016]. Citado na página 8.