ISSN 1519-0625

SIMULADOR DE AUTÔMATOS FINITOS DETERMINÍSTICOS - AUTOMATOGRAPH[1]

DETERMINISTIC FINITE AUTOMATA SIMULATOR - AUTOMATOGRAPH

Eduardo Bacchi Kienetz[2] e Ana Paula Canal[3]

RESUMO

O objetivo, neste trabalho, foi o desenvolvimento da ferramenta AutomatoGraph usada para representação de Autômatos Finitos Determinísticos - AFDs, um dos formalismos reconhecedores das Linguagens Regulares. Dado o caráter teórico da disciplina de Linguagens Formais e Autômatos, os alunos têm apresentado dificuldade em abstrair os conceitos para compreenderem o mecanismo de funcionamento dos formalismos. Uma forma de auxiliar nesse processo é utilizar um software que simule a execução dos autômatos. O AutomatoGraph foi desenvolvido na linguagem de programação Java e permite a construção e simulação do funcionamento de AFDs, bem como a geração da respectiva gramática regular.

Palavras-chave: linguagem formal, linguagem regular, gramática regular.

ABSTRACT

The goal of this project was the development of the AutomatoGraph tool, used to represent Deterministic Finite Automatas - DFAs, one of the formalisms recognizer of Regular Languages. Given the theoretical characteristic of the Formal Languages and Automatas subject, students have been showing difficulties in abstracting the concepts to comprehend the functional mechanism of the formalisms. One form of helping in this process is using a software program that simulates the execution of the automatas. AutomatoGraph was developed in the Java programming language and permits the construction and simulation of DFAs execution, as well as the generation of the respective Regular Grammar.

Keywords: *formal language, regular language, regular grammar.*

[1] Trabalho de Iniciação Científica - PROBIC.
[2] Acadêmico do Curso de Ciência da Computação - UNIFRA.
[3] Orientadora – UNIFRA.

INTRODUÇÃO

O estudo das Linguagens Formais desenvolveu-se significativamente e com diversos enfoques. Para a Ciência da Computação, algumas aplicações destacam-se como análise léxica e sintática de linguagens de programação, desenhos de *hardware* (circuitos digitais) e aplicações com linguagens naturais (MENEZES, 2000).

A disciplina de Linguagens Formais e Autômatos estuda as várias classes de linguagens formais, conforme a Hierarquia de Chomsky e os formalismos relacionados ao reconhecimento/geração/denotação dessas linguagens, como autômatos, gramáticas e expressões regulares, respectivamente. Dado o caráter teórico dessa disciplina, os alunos têm apresentado dificuldades em abstrair os conceitos, para compreenderem o mecanismo de funcionamento dos vários formalismos. Uma forma de auxiliar nesse processo de aprendizagem é através do uso de um *software* que simule a execução dos autômatos. Isso permite ao aluno construir o autômato que reconheça uma dada linguagem, escolhida por ele mesmo, e testar o funcionamento com palavras de entrada, as quais serão validadas pelo autômato simulado.

O objetivo, neste trabalho, foi o desenvolvimento do *software* AutomatoGraph para simular formalismos das Linguagens Regulares. Ele permite ao aluno construir um Autômato Finito Determinístico e validar o seu funcionamento, testando palavras da referida linguagem. Esse *software* trabalha com os formalismos reconhecedor (Autômato Finito Determinístico) e gerador (Gramática Regular) da classe das Linguagens Regulares.

Neste texto, inicialmente, é realizada uma explanação sobre as Linguagens Formais e suas classes de linguagens, conforme a Hierarquia de Chomsky. Após, descreve-se o desenvolvimento da ferramenta, caracterizando as seguintes etapas, consoante a Engenharia de *Software*: Levantamento de Requisitos, na qual é apresentada a definição formal de Autômato Finito Determinístico e Gramática Regular, bem como os requisitos de interface do sistema: Análise, Projeto e Implementação, caracterizando a estrutura do *software* em relação às classes de objetos utilizadas e implementação do algoritmo. A etapa de testes faz parte do processo de desenvolvimento de *software* e é tratada em uma seção a parte, a seção dos resultados, pois foi onde os alunos experimentaram a ferramenta e fizeram suas contribuições. Por fim, são apresentadas conclusões e possibilidades de trabalhos futuros.

LINGUAGENS FORMAIS

O estudo das Linguagens Formais iniciou na década de 50 e esteve relacionado ao estudo das Linguagens Naturais. Atualmente, suas principais aplicações são para análise léxica e sintática de linguagens, protocolos de comunicação que possuem número finito de estados, modelos de sistemas biológicos, verificação de ocorrências em corpo de texto, desenho de *hardware*, dentre outras (HOPCROFT et al., 2002).

À teoria das Linguagens Formais, os conceitos de análise léxica, sintática e semântica são importantes. Léxico é relativo a dicionário, assim a análise léxica abrange formalismos para a identificação da pertinência ou não de uma palavra/símbolo ao dicionário da linguagem. A análise sintática trata da verificação gramatical da linguagem, para a Ciência da Computação e das linguagens de programação. Já a análise semântica agrega o significado ou valor dos símbolos ao contexto dado (MENEZES, 2000).

Os formalismos usados para o tratamento sintático das linguagens podem ser distinguidos em três tipos: operacional, axiomático e denotacional. O formalismo operacional trata do reconhecimento das linguagens. Neste, encontram-se os autômatos. O formalismo axiomático, também chamado gerador, define regras associadas ao comportamento da linguagem para a geração de palavras, ou seja, são as gramáticas. Por fim, o formalismo denotacional possibilita representar uma linguagem, através das Expressões Regulares.

Chomsky definiu a Hierarquia das Linguagens (HOPCROFT et al., 2002), ilustrada na figura 1, também conhecida como Hierarquia de Chosmky. As linguagens são classificadas de acordo com sua complexidade e cada classe corresponde a uma classe de máquinas (formalismo), com maior ou menor e poder computacional. Nesse sentido, quanto mais interna for uma classe, menor é sua complexidade e é menor o poder computacional associado aos seus formalismos. Essa classificação é de suma importância para o tratamento de problemas de computabilidade (DIVERIO; MENEZES, 2000).

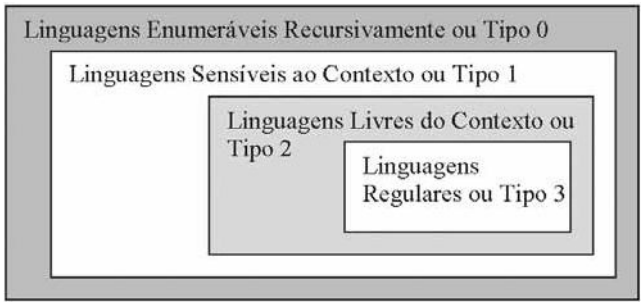

Figura 1. Hierarquia de Chomsky (MENEZES, 2000).

A Classe das Linguagens Regulares, objeto deste estudo, compreende os formalismos mais simplificados, porém não menos úteis, pois no estudo dos compiladores, esta classe é usada na análise léxica (PRICE; TOSCANI, 2001). De acordo com a Hierarquia de Chomsky, a Classe das Linguagens Regulares é a das linguagens mais simples, sendo possível a construção de algoritmos de reconhecimento ou de geração de pouca complexidade, grande eficiência e fácil implementação (MENEZES, 2000).

> São linguagens muito simples onde, por exemplo, não é possível fazer qualquer tipo de balanceamento de tamanho não-predefinido. A principal característica desta classe é que o tempo de reconhecimento de uma palavra é diretamente proporcional ao comprimento da entrada (DIVERIO; MENEZES, 2000, p. 127).

Como formalismos reconhecedores das Linguagens Regulares, tem-se o Autômato Finito Determinístico - AFD; Autômato Finito Não-Determinístico - AFND e o Autômato Finito com Movimentos Vazios - AFε. São todos baseados em um sistema de estados finitos para o reconhecimento de palavras da linguagem e equivalentes em relação à classe da linguagem que reconhecem (Linguagens Regulares). A diferença entre eles está no grau de generalização dos modelos de máquina que cada um propicia. O formalismo axiomático desta classe compreende a Gramática Regular e o formalismo denotacional, a Expressão Regular.

Neste trabalho, o foco está na Classe das Linguagens Regulares, especificamente, nos formalismos Autômato Finito Determinístico – AFD e Gramática Regular. Na próxima seção, aspectos do desenvolvimento do AutomatoGraph são apresentados.

A FERRAMENTA AUTOMATOGRAPH

O desenvolvimento da ferramenta AutomatoGraph seguiu as etapas definidas pela engenharia de *software* (PRESSMAN, 2006): levantamento de requisitos, análise, projeto, implementação e testes do sistema. A etapa de testes será abordada na Seção Resultados, pois trata dos resultados obtidos com o *software* através da validação com os alunos.

Para o Levantamento de Requisitos do sistema, foram realizados estudos sobre a Classe das Linguagens Regulares e dos formalismos reconhecedor (Autômato Finito Determinístico) e gerador (Gramática Regular). Para tanto, são definidos esses formalismos.

Segundo Menezes (2000), um Sistema de Estados Finitos é um modelo matemático de sistema com entradas e saídas discretas, que assume um número finito e predefinido de estados. Um Autômato Finito Determinístico (AFD) é um sistema de estados finitos e é uma máquina composta por três partes:

- fita de entrada: a palavra a ser processada está armazenada e será lida pelo autômato. É composta por células (uma para cada símbolo);
- Unidade de Controle é uma unidade de leitura. Inicialmente posicionada na célula mais à esquerda da fita. Acessa a uma célula de cada vez e movimenta-se para a direita;
- Programa ou Função de Transição ou Função Programa é a função que determina o comportamento do autômato, conforme o símbolo lido da fita e o estado em que se encontra.

Formalmente, um AFD é definido por uma 5-upla $M = (\Sigma, Q, \delta, q0, F)$, em que: Σ é o alfabeto da linguagem; Q é o conjunto dos possíveis estados do autômato; δ é a Função Programa ou Função Transição $\delta: \Sigma \times Q \rightarrow Q$; q0 é o estado inicial e F é o conjunto de Estados Finais (MENEZES, 2000).

Na figura 2, há a representação em forma de grafo, da função programa do Autômato $M1 = (\{a, b, c\}, \{q0, q1, q2, q3\}, \delta, q0, \{q3\})$, que reconhece a Linguagem Regular, denotada pela Expressão Regular (1)

*(ac + ba + aba + aca) (a+b)** (1)

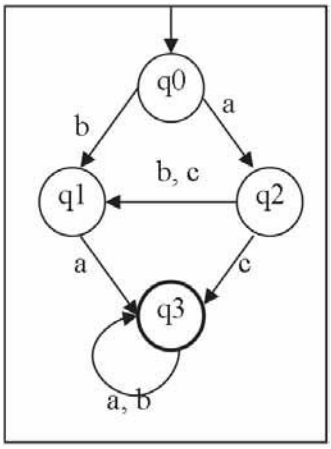

Figura 2. Autômato Finito Determinístico M_1.

Uma Gramática Regular é definida, formalmente, como uma quádrupla ordenada G = (V, T, P, S), em que: V é o conjunto de símbolos

variáveis ou não-terminais; T é o conjunto de símbolos terminais, disjunto de V; P é o conjunto finito de pares, denominados Regras de Produção de tal forma que a primeira componente é palavra de $(V\ T)^+$ e a segunda é palavra de $(V \cup T)^*$; S é elemento de V, denominado variável inicial (MENEZES, 2000).

Uma regra de produção (α, β) de uma Gramática é representada por: $\alpha \rightarrow \beta$ e define as condições de geração das palavras da linguagem. A aplicação sucessiva de regras de produção permite derivar as palavras da linguagem.

Nas etapas de análise e projeto, foi definida a estrutura do sistema. Foram modeladas as classes do sistema, definidas as interfaces gráficas e desenvolvidos os algoritmos para simulação do processamento de uma palavra pela função programa do autômato e para geração da Gramática Regular a partir do Autômato inserido pelo usuário.

O *software* é composto por cinco Classes: AutomatoGraph, *Reader*, *Transition*, *Automaton* e *State*. Na figura 3, está o diagrama de classes com as dependências entre elas.

Com relação às suas funcionalidades, o *software* possibilita o reconhecimento de palavras pertencentes a uma dada linguagem. Ao executá-lo, o usuário informa o número de estados do Autômato Finito Determinístico, especificando estado inicial e estados finais, alfabeto e transições entre os estados. A partir disso, é possível fazer o reconhecimento de palavras com o intuito de saber se as mesmas são reconhecidas pela linguagem para a qual o autômato foi configurado.

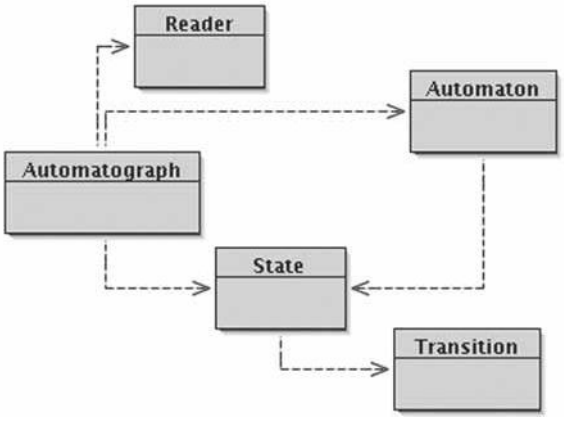

Figura 3. Diagrama de classes simplificado.

O método de reconhecimento utilizado pelo simulador percorre a palavra-teste (palavra inserida pelo usuário, após a configuração do autômato), seguindo as transições estabelecidas pela máquina de estados. Ao final do processamento, é informado se a palavra foi aceita, pertence à linguagem, foi rejeitada, ou não pertence à linguagem.

O reconhecimento acontece através da execução da função programa do autômato, sempre no sentido estado-inicial → estado-final, para verificar se a palavra-teste pertence ou não à linguagem. Durante esse processamento, o AutomatoGraph informa a transição que está sendo verificada. Em caso de aceite, a última transição informada aponta para o estado final do autômato. Para a geração da gramática regular, foi desenvolvido um algoritmo para gerá-la a partir do autômato inserido pelo usuário.

Na figura 4, encontra-se a interface principal do *software*. Na caixa à esquerda, é mostrada a seqüência de passos percorridos pelo autômato para efetuar o reconhecimento da palavra de entrada. Na caixa à direita, é apresentada a Gramática Regular, gerada a partir do Autômato Finito Determinístico, inserido pelo usuário. Os resultados mostrados nessas interfaces referem-se ao Autômato Finito Determinístico do exemplo ilustrado na figura 2.

A implementação do sistema foi realizada com a linguagem de programação orientada a objetos Java (SUN, 2006). Em um primeiro momento foi implementado o Autômato Finito Determinístico para o reconhecimento de palavras de uma linguagem, posteriormente, agregou-se ao *software* a geração da Gramática Regular. Para a geração de documentação e empacotamento do *software,* foi utilizado o editor BlueJ (BLUEJ, 2006).

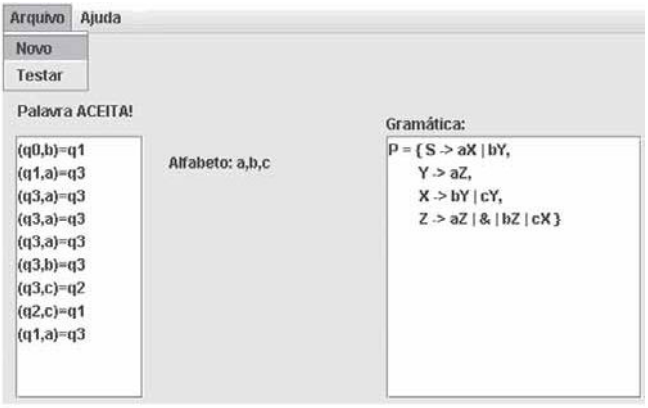

Figura 4. Interface Principal da Versão 1 do AutomatoGraph

RESULTADOS

Como resultado principal deste projeto, tem-se a ferramenta AutomatoGraph que suporta o reconhecimento de palavras da classe das Linguagens Regulares, através do Autômato Finito Determinístico e permite gerar a Gramática Regular da respectiva Linguagem.

O AutomatoGraph foi testado por alguns alunos da turma da Disciplina de Linguagens Formais e Autômatos. Considerações foram feitas em relação à Gramática Regular que, inicialmente, não era gerada se o alfabeto da linguagem fosse diferente de letras. Pequenas alterações quanto à Interface Usuário-Máquina também foram sugeridas, como por exemplo: informar ao usuário o tipo de Autômato que deve ser inserido (para esta versão do *software*, o Autômato Finito Determinístico) e deixar claro quais caracteres podem ser inseridos para compor o alfabeto da linguagem. Todas as sugestões, tanto da Gramática quanto da Interface, foram efetuadas e constam na versão 1 do AutomatoGraph.

Como trabalhos relacionados ao AutomatoGraph, salientam-se o VAS – *Visual Automato Simulator* (BOVET, 2004), que faz a simulação de autômatos finitos e da máquina de *Turing* e o jFAST (WHITE; WAY, 2006), que simula autômatos finitos, enfatizando a visualização deste formalismo. O AutomatoGraph diferencia-se por, além de permitir a simulação do autômato, gerar a Gramática Regular correspondente ao autômato representado.

CONCLUSÕES E TRABALHOS FUTUROS

O estudo das Linguagens Formais é fundamental para os alunos da computação, pois envolve formalismos de reconhecimento e geração de linguagens que são usados na construção de interpretadores/compiladores de linguagens de programação.

Considera-se que este *software* é uma ferramenta que auxilia no processo de aprendizagem, pois simula a execução de Autômato Finitos Determinísticos e constrói a Gramática Regular, formalismos reconhecedor e gerador, respectivamente, associados à classe das Linguagens Regulares. Isso permite a experimentação prática das máquinas abstratas abordadas na Disciplina de Linguagens Formais e Autômatos.

A ferramenta desenvolvida possibilita a realização de um número maior de testes sobre a máquina (autômato) construída. Pretende-se continuar a utilização desta ferramenta com turmas da Disciplina de Linguagens Formais e Autômatos, a fim de auxiliar no aprendizado dos formalismos relacionados às Linguagens Regulares e permitir o seu aprimoramento. Também é prevista, a construção de outra versão do *software,* que inclui o processamento do Autômato Finito Não-Determinístico.

REFERÊNCIAS BIBLIOGRÁFICAS

BLUEJ. **BlueJ:** the interactive Java environment. Disponibilidade em: http://www.bluej.org . Acesso em Julho de 2006.

BOVET, J. **Visual Automata Simulator:** a tool for simulating automata and turing machines. Universidade de San Francisco. 2004. Disponibilidade em http://www.cs.usfca.edu/~jbovet/vas.html. Acesso em Maio de 2006..

DIVERIO, T. A.; MENEZES, P. B. **Teoria da computação**: máquinas universais e computabilidade. Porto Alegre: Sagra Luzzatto, 2000.

HOPCROFT, John E.; ULLMAN, Jeffrey D.; MOTWANI, Rajeev. **Introdução à teoria de autômatos, linguagens e computação**. Rio de Janeiro: Campus, 2002.

MENEZES, Paulo Blauth. **Linguagens formais e autômatos**. Porto Alegre: Sagra Luzzatto, 2000.

PRESSMAN, Roger S. **Engenharia de *software***. Rio de Janeiro: McGraw Hill, 2006.

PRICE, A. M. A.; TOSCANI, S. S**. Implementação de linguagens de programação**. Porto Alegre: Sagra Luzzatto, 2001.

SUN MICROSYSTEMS. **Java 2.** Plataform SE v1.4.2, Standard Edition, API Specification. Disponibilidade em: http://java.sun.com/j2se/1.4.2/docs/api/ . Acesso Maio de 2006.

WHITE, Timothy M.; WAY, Thomas P. jFAST: a java finite automata simulator. **Proccedings of 37th SIGCSE,** New York, Issue 1, v. 38, p. 384-388, março, 2006.