Ambiente Computacional Aplicado: Recursos para Ajuda Educacional

Mário Peixoto, Eliana Almeida

Instituto de Computação
Universidade Federal de Alagoas
mario.peixoto@gmail.com, eliana.almeida@pesquisador.cpnq.br

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

Resumo

O objetivo deste projeto é estender o AmbAP (ver de Almeida, 2002) a um ambiente integrado de ferramentas para auxílio no estudo de computação, e não apenas de programação, permitindo, através da construção de plugins (módulos adicionais), introduzir novas funcionalidades, como, por exemplo, o suporte a teoria da computação através da representação dos modelos de computação (desenvolvido nesta etapa do projeto), especificamente da Máquina de Turing.

Introdução

O módulo trabalhado aqui consiste de um simulador de Máquina de Turing. As ferramentas disponibilizadas atualmente para a criação/simulação/execução de máquinas de Turing, são complexas na sua execução, com interfaces não-amigáveis, o que acaba por confudir os alunos aprendizes de teoria da computação. Uma nova abordagem de criação/simulação/execução de máquinas de Turing, mais intuitiva, que facilite a compreensão dos conceitos associados a este modelo de computação, seria de grande valia para o ensino de computabilidade. Este projeto permitiu a criação do módulo, a ser integrado ao ACARAjE, que desmistifica os conceitos teóricos associados a Máquina de Turing, através da visualização/simulação desta de uma forma prática (inclusive explicitando o estado da fita após cada instrução).

Estudo da máquina de turing

Primeiramente, estudou-se a máquina de turing como um modelo formal de procedimento efetivo, algoritmo ou função computável, onde se enfatizou o uso desta como máquina executora de instruções e reconhecedora de linguagens enumeráveis recursivamente e sensíveis ao contexto. Essa leitura teve como ponto de partida os livros "Elements of the Theory of Computation" (ver Lewis & Papadimitriou, 1997) e "Introduction to the Theory of Computation" (ver Sisper, 2005).

Ferramentas existentes

"Turing Machines implemented in JavaScript" (ver *Turing Machines implemented in JavaScript*, n.d.):

- implementação simplória;
- difícil uso;
- necessário conhecimento do modelo formal.

JFLAP (ver JFLAP, n.d.):

- implementação mais sofisticada;
- uso mais simples, porém não é simples o suficiente;
- pouco extensível.

Ferramentas e modelos de desenvolvimento utilizados

Ferramentas:

- Java Technology (ver Sun, n.d.);
- Java Graph Visualization and Layout (ver *Java Graph Visualization and Layout*, n.d.);
- Eclipse Platform (ver IBM, n.d.).

Modelos de desenvolvimento:

- eXtreme Programming (ver *eXtreme Programming*, n.d.);
- Test Driven Development (ver *Test-Driven Development*, n.d.).

Modelagem do módulo

Nesta etapa foi feita a modelagem do módulo simulador de máquina de Turing, utilizando-se da linguagem UML já estudada anteriormente, chegamos ao modelo (resumido) mostrado na figura 1 em que se vê claramente o uso do padrão de projeto de MVC, apresentado por Freeman & Freeman (2005), para o desacoplamento da interface do interpretador propriamente dito.

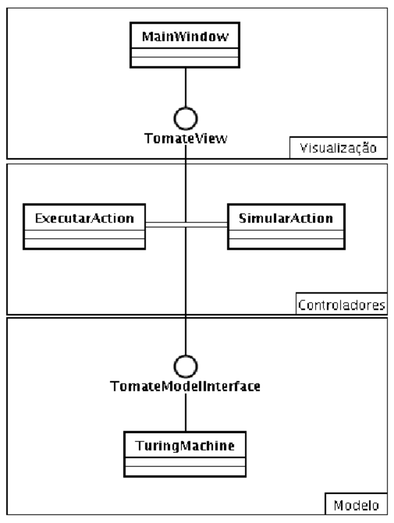

Figura 1: *Modelagem do sistema*

Modelo:

- parte do sistema que efetivamente executa as ações;
- arquitetura interna não visível ao usuário.

Visualização:

- parte do sistema que contém as classes relacionadas à interface de usuário;
- cada ação executada pelo usuário resulta na execução de uma *Action* do controlador.

Controladores:

- parte do sistema que contém as *Actions* que serão executadas quando o usuário solicitar;
-

A ferramenta

De forma a exemplificar a utilização da ferramenta `TOMaTE`, segue como exemplo a construção de uma máquina de Turing que reconhece uma palavra do tipo *aa*bb*aa**:

- construção da máquina de Turing utilizando a barra de ícones (ver figura 2;
- simulação da máquina de Turing construída com explicitação de estado atual e conteúdo fita (ver figura 3;
- resposta do sistema quando informado uma palavra válida (ver figura 4);
- resposta do sistema quando informado uma palavra inválida (ver figura 5).

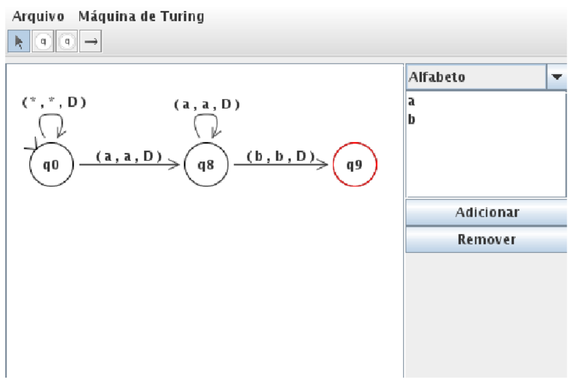

Figura 2: *TOMaTE – Construção*

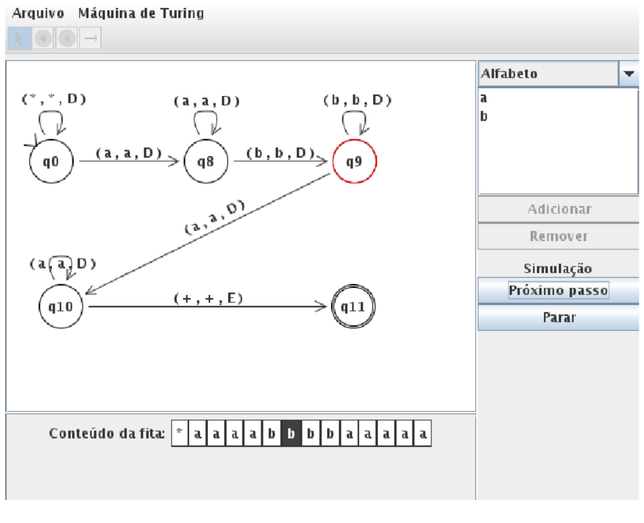

Figura 3: *TOMaTE – Simulação*

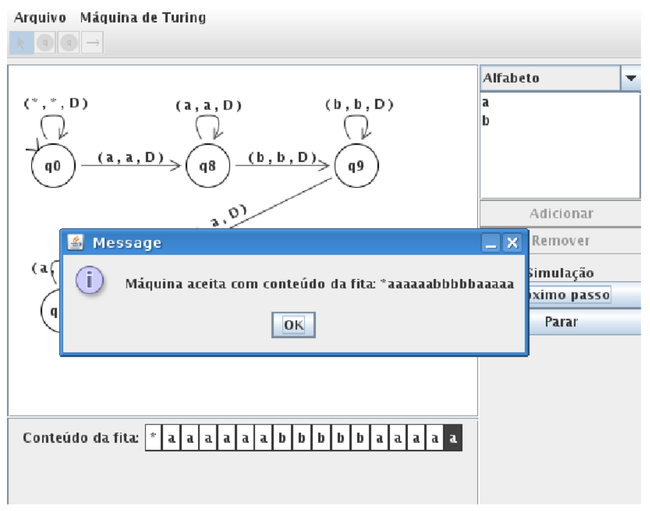

Figura 4: *TOMaTE – Máquina aceita*

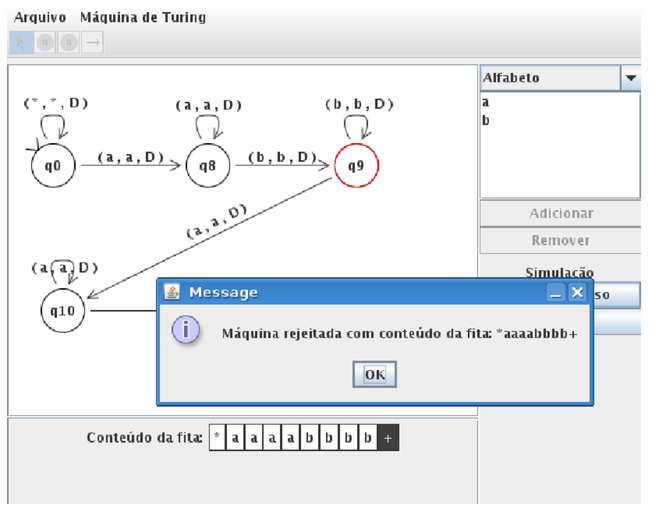

Figura 5: *TOMaTE – Máquina rejeitada*

Referências

de Almeida, E. S. (2002), Ambap: Um ambiente de aprendizado de programação, *in* 'Anais do XXII Congresso da Sociedade Brasileira de Computação - X WEI'.
eXtreme Programming

eXtreme Programming *(n.d.)*. *URL* `http://www.extremeprogramming.org`, *última consulta em janeiro de 2007*.

Freeman, E. & Freeman, E. (2005), Use a cabeça! Padrões de Projeto (Design Patterns)*, 1 ed., Alta Books*.

IBM (n.d.), 'Eclipse platform'. URL `http://www.eclipse.org`.
Java Graph Visualization and Layout

Java Graph Visualization and Layout (n.d.). URL `http://www.jgraph.com`, última consulta em agosto de 2007.

JFLAP (n.d.), 'An interactive formal languages automata package'. URL `http://www.jflap.org`, última consulta em janeiro 2007.

Lewis, H. R. & Papadimitriou, C. H. (1997), *Elements of the Theory of Computation*, 2 ed., Prentice Hall International.

Sisper, M. (2005), *Introduction to the Theory of Computation*, 2 ed., Course Technology.

Sun (n.d.), 'Java technology'. URL `http://java.sun.com`, última consulta em janeiro de 2007.
Test-Driven Development

Test-Driven Development *(n.d.). URL* `http://www.testdriven.com`, *última consulta em janeiro de 2007*.
Turing Machines implemented in JavaScript

Turing Machines implemented in JavaScript (n.d.). URL `http://www.turing.org.uk/turing/scrapbook/tmjava.html`, última consulta em janeiro de 2007.