UNASP

CENTRO UNIVERSITÁRIO ADVENTISTA DE SÃO PAULO

CAMPUS SÃO PAULO

CURSO DE CIÊNCIA DA COMPUTAÇÃO

Wellmmer Lucas de Oliveira Pinto

Simulador de Máquina de Turing: Uma Aplicação Web para o Ensino

Trabalho de conclusão de curso apresentado ao Centro Universitário Adventista para obtenção do título de Bacharel em Ciência da Computação.

São Paulo, SP, Novembro de 2016

Wellmmer Lucas de Oliveira Pinto

Simulador de Máquina de Turing: Uma Aplicação Web para o Ensino

Trabalho de conclusão de curso apresentado ao Centro Universitário Adventista para obtenção do título de Bacharel em Ciência da Computação

Área de Concentração:
Ciência da Computação

Orientador:
Prof. Ms. Laercio Martins Carpes

São Paulo, SP, Novembro de 2016

Dedico este trabalho ao homem mais sábio deste mundo na minha opinião, e a mulher com a maior fé que já vi em toda minha vida.... Meus queridos e amados pais.

RESUMO

O legado deixado por Alan Turing (1912–1954), com certeza, é um inegavelmente grandioso e tem contribuído com estudos e pesquisas até os dias de hoje. O artigo nomeado *"On Computable Numbers, with an Application to the Entscheidungsproblem*" contempla uma de suas maiores invenções e que hoje usamos e chamamos de computadores. Turing detalha neste artigo sua teoria sobre os números computáveis, máquinas computáveis e como o que ele chamou de "máquina universal", conhecida hoje por máquina de Turing, atenderia como resposta ao tão polêmico *"Problema de Decisão"*, ou do original alemão "*Entscheidungsproblem*", de David Hilbert (1862-1943). No entanto, a linguagem usada por Turing para explanar suas ideias soa muitas vezes de forma complicada e por assim dizer, muito específica à matemática. Segundo Leavitt (2007) Turing desbrava, em seu artigo, pântanos de símbolos não familiares, letras alemãs e gregas, e números binários, além de uma linguagem filosófica e matemática altamente técnica. Foi então, pensando em simplificar o estudo da máquina de Turing que este trabalho propõe uma aplicação web voltada para o ensino dos conceitos e funcionamento da máquina de Turing através de um um simulador.

Palavra Chave: Máquina de Turing; Simulador; Aplicação Web.

ABSTRACT

The legacy left by Alan Turing (1912-1954), certainly, is undeniably magnificent and has contributed to studies and research until these days. The article named "On Computable Numbers, with an Application to the Entscheidungsproblem" contemplates one of his greatest inventions and that today we use and call computers. Turing details in this paper his theory on computable numbers, computable machines and how what he called "universal machine", known today as the Turing machine, would answer in response to the controversial "Decision Problem" or the German original "Entscheidungsproblem" of David Hilbert (1862-1943). However, the language used by Turing to explain his ideas often sounds complicated and, so to speak, very specific to mathematics. According to Leavitt (2007) Turing raises, in his article, swamps of unfamiliar symbols, German and Greek letters, and binary numbers, in addition to a highly technical philosophical and mathematical language. Then, thinking about simplifying the study of the Turing machine that this project proposes a web application focused on teaching the concepts and functioning of the Turing machine through a simulator.

Keywords: Turing Machine; Simulator; Web Application.

AGRADECIMENTOS

Agradeço à Deus em primeiro lugar pois “tudo foi possível nAquele que sempre me fortaleceu”. Também dirijo meus agradecimentos aos meus amigos que me ajudaram de diversas formas para conclusão deste trabalho: Guilherme Iazzetta, Lucas Coelho, Gabriel Guimarães e André Buthner. Sem muitas delongas agradeço imensamente a minha namorada Layla Camargo que esteve ao meu lado inúmeras manhãs, tardes, noites e madrugadas, me ajudando a escrever, revisar e desenvolver este trabalho. E por fim, agradeço aos meu amados pais por todo apoio, investimento e confiança.

Sumário

1 INTRODUÇÃO

De acordo com o *American Heritage Dictionary* (2016), a palavra "algoritmo" é "um processo de solução de um problema passo a passo, especialmente um procedimento computacional recursivo estabelecido para a solução de um problema num número de passos finitos.". Embora hoje conhecemos muito bem a palavra "algoritmo" e seu significado, voltemos para o século XIII onde tudo isso estava no escuro da mente de alguns homens ousados, dos quais chamamos hoje de gênios.

No terceiro capítulo do livro *"O homem que sabia demais"*, de David Leavitt (2007), é esboçada uma introdução muito detalhada da história por trás do *Entscheidungsproblem*, ou melhor dizendo, o problema de decisão. Começando por Raimundus Lullus (1232-1316) e toda sua tese em cima do método geral de solução de problemas, intitulado por ele de *ars magna.* Mais tarde ampliado pelos estudos de Leibniz (1646-1716), resultando no estabelecimento de uma linguagem simbólica que efetivasse a solução do problema, a *characteristica universalis*, e na distinção de duas versões da *ars magna:* a *ars inveniendi*, encontrando as verdadeiras afirmações científicas, e a *ars iudicandi*, permitindo a decisão se uma afirmação científica é verdadeira ou não.

Quase um século depois, o matemático alemão David Hilbert (1862-1943) lança para a comunidade de matemáticos da época o *Entscheidungsproblem,* que caía na perspectiva do *ars iudicandi* de Leibniz, no qual, segundo Hilbert poderia se restringir a uma simples questão de "sim" ou "não" para o caso de haver um algoritmo que decida a validade de uma fórmula de primeira ordem. Isso gerou muita polêmica entre a comunidade de matemáticos, quando de um lado tinham matemáticos prós à uma descoberta para o problema de decisão, e do outro matemáticos contra, como o caso do matemático inglês G. H. Hardy (1877-1947), ao dizer que "isso seria uma infelicidade, pois se houvesse [o algoritmo], precisaríamos ter um conjunto mecânico de regras para a solução de todos os problemas matemáticos, e nossas atividades como matemáticos chegariam a um fim.".

No entanto, havia um jovem britânico que por seu constante isolamento, possivelmente não estava de nenhum dos lados em relação a solução do *Entscheidungsproblem,* porém encarava o problema de decisão uma simples questão que requisitava-se uma solução. Talvez pelo motivo claro de Turing não considerar o problema esperando um possível resultado negativo ou positivo, ele conseguiu enfrentá-lo de uma maneira totalmente inovadora, e foi

desse modo tão literal de pensar que Turing apresentou seus resultados, em 1936, no artigo publicado *"On Computable Numbers, with an Application to the Entscheidungsproblem"*, no qual ele define o conceito e as possíveis funcionalidades da "máquina universal", e desbanca o problema de decisão. (Leavitt, 2007)

1.1 Argumentação Justificativa

Em um breve artigo, escrito sob encomenda e destinado à alunos de Ciência da Computação, o renomado Prof. Dr. Eng. Valdemar W. Setzer, do Depto. de Ciência da Computação da USP, destaca a importância dos estudos e legado deixado por Turing ao enfatizar:

> *"Se o leitor está terminando um Bacharelado em Ciência da Computação, e nunca ouviu falar o nome Turing, é melhor começar tudo de novo em uma faculdade de um nível razoável. Isso se deve ao fato de Alan Mathison Turing (1912-1954) ter estabelecido alguns dos fundamentos mais importantes da Ciência da Computação, tanto do ponto de vista prático quanto teórico e, portanto, deve obrigatoriamente ser mencionado e suas descobertas estudadas em qualquer curso dessa área." (Setzer, V. W. 2003).*

Seguindo este raciocínio, embora o aprendizado dos conceitos da *"máquina universal"* de Turing, contidos em seu artigo citado acima, sejam imprescindíveis tanto para um ingressante quanto para um formando de Bacharelado em Ciência da Computação, como em qualquer outro curso na área da tecnologia, é viável por antemão ser assumido que a compreensão de tais não é algo tão trivial. O estudo sobre a *"máquina universal"* de Turing é algo que pode parecer simples superficialmente, mas é necessário um aprofundamento técnico matemático para compreender certas representações, equações e símbolos usados por Turing para explanar suas ideias.

De acordo com David Leavitt, autor do livro *"O homem que sabia demais"*:

> *"Como muitos trabalhos de Turing, 'Computable Numbers' é marcado por uma curiosa mistura de frases cheias de modéstia, especulação meio*

> *filosófica e matemática altamente técnica. O resultado, para um leitor comum, é desconcertante, pois invariavelmente essas passagens, cuja importância é fácil de aprender, passam suave e imediatamente em densos pântanos de símbolos não familiares, letras alemãs e gregas, e números binários." (Leavitt, 2007).*

Portanto, foi pensando carinhosamente no legado e todo trabalho deixado por Alan Turing, que este trabalho procura por objetivo imputar os conceitos e funcionalidades da máquina de Turing em uma aplicação web e, através de uma forma simples e didática, possa ajudar na aprendizagem dos fundamentos da computação deixados por sua descoberta. Assim, a disponibilização de todo o material desenvolvido tenha como intuito o uso como referencial teórico e prático em estudos e trabalhos futuros, demonstrações, aulas, etc.

2 REFERENCIAL TEÓRICO

2.1 Entscheidungsproblem: O Problema de Decisão

Segundo David Hilbert (1928), o *Entscheidungsproblem* é resolvido quando conhecemos um procedimento que permite, para qualquer expressão lógica dada, decidir sua validade ou satisfatibilidade. Boolos e Jeffrey (1974) completam afirmando que "uma expressão apenas é satisfazível se houver a possibilidade de uma interpretação ou modelo torná-la verdadeira".

Partindo dessa premissa, Hilbert propôs o desafio chamado *"Problema de Decisão"* ou *"Entscheidungsproblem"*, do original alemão. Neste desafio, um certo algoritmo, por assim dizer, recebe como entrada dados em linguagem formal e dados matemáticos, que outrora retornam ao usuário um sinal "verdadeiro ou falso" como resultado de validação. Após trabalhar tais dados, o algoritmo tem o objetivo de validar qualquer expressão lógica à ele atribuída, retornando então se tal expressão é ou não válida e, também, o seu nível satisfatório.

Ruy J.G.B. de Queiroz acrescenta dizendo:

> *"Tal algoritmo seria capaz de decidir, por exemplo, se enunciados tais como a conjectura de Goldbach ou a hipótese de Riemann, são verdadeiras, muito embora nenhuma prova ou refutação desses enunciados seja conhecida." (Ruy J.G.B. de Queiroz, 2012)*

2.2 A solução de Alan M. Turing

Na visão de David Leavitt (2007), o artigo é basicamente dividido em três partes: a primeira definindo o conceito dos *"números computáveis"* e da *"máquina computável"*; a segunda abordando o conceito e as funcionalidades de uma possível *"máquina universal"*; e finalmente a terceira aplicando toda a teoria abordada para provar a insolubilidade do *Entscheidungsproblem*. Colocando em nota que para este trabalho, não será relevante mostrar a conclusão da *"máquina universal"* de Turing ao provar que o *Entscheidungsproblem* é insolúvel.

2.2.1 O que são "números computáveis"?

Turing (1936) inicia seu artigo definindo *"números computáveis"* como "números reais cujas expressões decimais são calculáveis por meios finitos", porém ainda no final do primeiro capítulo, embora sem nenhuma ostentação ao dizer, os redefine afirmando ousadamente: *"De acordo com minha definição, um número é computável se seu decimal pode ser registrado por uma máquina." (Turing, 1936)*

Leavitt (2007) comenta que "a importância de tal afirmação não deve ser subestimada, uma vez que falar de uma hipotética *"máquina"* de computação, ainda mais e um artigo sobre matemática nos anos de 1930, era como violar as regras de uma ortodoxia assumidamente rígida, pois nenhuma dessa possíveis *"máquinas"* existia à época, apenas algumas muito cruas e obviamente não programáveis."

2.2.2 O que são "máquinas computáveis"?

Sem tentar justificar muito antes da hora, a razão de sua definição particular para os *"números computáveis"*, Turing (1936) estabelece que "de fato a mente (memória) humana é devidamente limitada" e que podemos "comparar um homem no processo de computação de um número real com uma máquina que é unicamente capaz de um número finito de condições". Turing chamou a esse número finito de condições de *"configuração-m"*.

De acordo com Turing (1936), a *"máquina"* é alimentada, por assim dizer, com uma "*fita"* percorrendo sua estrutura. A *"fita"* é dividida em seções, como *"células"*, cada uma capaz de receber um *"símbolo"*. A *"célula"* que, no momento, estiver na *"máquina"* é chamada de *"célula registrada",* quando que o *"símbolo"* da *"célula registrada" é* chamado

“símbolo registrado”. O possível comportamento da *“máquina”*, para qualquer situação, pode ser determinado pela *“configuração-m”* juntamente com o *“símbolo registrado”*, definindo a *“configuração da máquina”.* Leavitt (2007) resume os possíveis comportamentos da máquina comentando que:

> *“Dependendo de sua configuração a máquina vai escrever um símbolo numa célula em branco, apagar um símbolo já escrito lá, mover a fita um espaço para esquerda ou mover a fita um espaço para direita. O que determina como ela irá agir é uma ‘tabela de comportamento’ especificando a sequência das configurações-m de acordo com as quais a máquina pode executar seu algoritmo particular”.*

2.3 A Enigma

De acordo com WINTERBOTHAM (1978), Enigma é o nome pelo qual foi conhecida a máquina eletromecânica de criptografia com rotores, utilizada tanto para criptografar como para descriptografar códigos de guerra, usada em várias formas na Europa a partir dos anos 1920.

Tratava-se de uma máquina de médio porte, com medidas 65x45x35 cm e pesando cerca de 50 kg. Sua primeira versão foi patenteada em 1918, por Arthur Scherbius e exibidas nos congressos da União Postal Universal em 1923 e 1924. Esta primeira versão foi chamada de Enigma modelo A.

> *“Três outras versões comerciais lhe sucedem, e a Enigma-D torna-se o modelo mais divulgado após suscitar o interesse da marinha alemã em 1926. A marinha alemã interessou-se pela Enigma e comprou alguns exemplares, adaptando-as ao seu uso em 1926. Estas primeiras máquinas de uso militar denominavam-se Funkschlüssel C.*
>
> *Em 1928 o exército elaborou a sua própria versão - a Enigma G. A partir desse momento, o seu uso estende-se a toda a organização militar alemã e a uma grande parte*

> *da hierarquia nazi. A marinha chama a Enigma a máquina M." (*WINTERBOTHAM, *1978)*

Em meados 1938, Alan Turing foi convocado a se juntar ao GC&CS, em Bletchley Park, conhecido na época como o centro de decodificação de mensagens da inteligência britânica, para efetuar a criptoanálise da Enigma nazista. A máquina tinha por rotina mudar os padrões de códigos e criptografia diariamente, obrigando que o desafio de decifração fosse o mais rápido possível. Em 1940, Turing projetou a *"Bombe"*, uma máquina eletromecânica que ajudaria a decriptar as mensagens da Enigma.

A fama desta máquina alavancou-se após ter sido utilizada pelas forças militares alemãs, em 1930, entusiasmados pela facilidade de uso e supostos "códigos indecifráveis". Porventura, com a quebra desses conceitos e da suposta visão de "máquina indecifrável", a Enigma se desvalorizou e foi tida como a responsável pelo fim prematuro da Segunda Guerra Mundial.

> *"O código foi de fato quebrado em 1933 por matemáticos da Polônia (Marian Rejewski, Jerzy Różycki e Henryk Zygalski) com a ajuda de meios eletromecânicos, as bombas. [...] Versões aperfeiçoadas das "bombas" polonesas criadas pelos britânicos em Bletchley Park, sob a liderança do matemático Alan Turing, aceleraram o processo de decodificação das Enigmas usadas pela marinha alemã." (*WINTERBOTHAM, *1978)*

2.4 Autômatos Finitos

As máquinas de Turing são uma categoria de autômatos finitos caracterizada pela linguagem de recursivamente enumerável, do qual se aplica a resposta ao "Problema de Decisão" de David Hilbert. Izabela Melo, do Departamento de Sistemas & Computação da UFCG nos traz uma definição mais abstrata e "humana", do que vem a ser um autômato finito:

> *"Você já parou para pensar em como funcionam as portas que abrem e fecham automaticamente? E as lavadoras de louça/roupa, termômetros eletrônicos, relógios digitais, calculadoras e máquinas de venda automática? Todos esses dispositivos eletromecânicos*

> *têm uma controladora que nada mais é do que um autômato finito.*
>
> *Além desses dispositivos (controladores), os autômatos finitos também são importantes para reconhecer padrões em dados, como, por exemplo, na análise léxica de um compilador; projetar uma nova linguagem para uma aplicação específica (desenvolvendo seu compilador); processamento de voz; etc." (MELO, 2011)*

Os autômatos finitos são modelos simples computacionais, usados em máquinas com memórias extremamente limitadas, podendo ser exemplificada com uma porta automática de um supermercado: Geralmente, existem dois tapetes, um à frente e outro atrás da porta e neles estão alinhados os eixos do sensor de presença. Neste exemplo, o autômato funciona de forma bem simples: Quando alguém se enquadra em um dos tapetes a porta se abre e só se fecha quando a pessoa complete a passagem pelos dois tapetes, evitando assim, que a porta se feche atingindo o usuário.

Em uma linguagem mais computacional:

> "O controlador pode estar em dois estados (aberto ou fechado) e passa de um estado para outro dependendo do estímulo (entrada) que recebe. Nesse caso, tem-se 4 estímulos diferentes: frente (uma pessoa está no tapete da frente); retaguarda (uma pessoa está no tapete de dentro); ambos (há pessoas sobre os dois tapetes) e nenhum (não há ninguém sobre os tapetes). Para cada estímulo, o controlador vai responder de uma forma diferente, realizando as transições de um estado para outro. A representação das possíveis transições de acordo com cada estímulo, chamamos essa representação de diagrama de estados." (MELO, 2011)

2.5 E o “Lambda Calculus”?

Relacionado com o tema principal deste trabalho, temos o cálculo lambda, ou do original *“Lambda Calculus”*, de Alonzo Church, 1936. Em 1937, Alan Turing provou a equivalência entre a sua máquina e o cálculo criado por Church em termos de computabilidade.

De acordo com um artigo de Jorge Muniz Barreto, professor da Universidade Federal de Santa Catarina:

> *“O cálculo lambda é uma coleção de diversos sistemas formais baseados em uma notação para funções inventada por Alonzo Church em 1936 com o intuito de capturar os aspectos mais básicos da maneira pela qual operadores ou funções podem ser combinados para formar outros operadores. O cálculo lambda serve como uma ponte entre linguagens funcionais de alto nível e suas implementáveis de baixo nível. Raízes para a apresentação do cálculo lambda como uma linguagem intermediária: uma linguagem extremamente simples, consistindo de somente algumas poucas construções sintáticas e de uma semântica simples. Uma implementação do cálculo lambda necessita somente suportar algumas construções simples. A sua semântica simples nos permite analisar facilmente a correção de sua implementação. Trata-se de uma linguagem expressiva, a qual é suficientemente poderosa para expressar todos os programas funcionais e, por conseguinte, todas as funções computáveis. Isto significa que, se uma boa implementação do cálculo lambda é disponível, pode-se implementar qualquer linguagem funcional através da implementação de um compilador desta para o cálculo lambda.”*

3 PROJETO

3.1 Objetivo Geral

O principal objetivo deste trabalho é tornar simples e didático todo o ensino do funcionamento e conceitos da máquina de Turing, através de uma aplicação web. Para efeito de aprendizagem a aplicação terá animações e características visuais que irão detalhar passo a passo, como um tutorial, ou até mesmo um estudo dirigido, o que Turing quis descrever em seu artigo sobre a "máquina universal", que outrora, por via de livros e artigos, talvez seria, consideravelmente, de difícil absorção.

3.2 Metodologia

Antes de iniciar o desenvolvimento da aplicação, será feito um levantamento das possíveis funcionalidades e esboços do design e características visuais da aplicação. Todo o projeto estará seguindo o tradicional modelo de engenharia de software, ou mais conhecido como "Modelo Cascata", criado por Royce em 1970, justo selecionado pela facilidade de implementação e organização do cronograma de atividades. (BOEHM, 1987) Então, após toda a parte de análise do projeto, serão desenvolvidos e implementados cada conceito e funcionalidade da máquina de Turing, tendo como aplicação final um simulador de máquina de Turing que auxiliará no ensino e na aprendizagem dos conceitos de Turing para a máquina.

Devido à uma melhor portabilidade do projeto em geral, falando mais apropriadamente do código fonte da aplicação, será utilizado o editor de código Visual Studio Code, recentemente lançado pela Microsoft e de alto índice de aceitação pela comunidade de desenvolvedores. O Visual Studio Code é de categoria *open-source,* disponibilizado para os principais sistemas operacionais Windows, Linux e Mac, e vem com um "leque" considerável de *plugins*, extensões e recursos para diversas linguagens de programação.

Para o desenvolvimento do projeto web foi utilizado como linguagem nativa o Javascript com plugins de jQuery para tratamento de elementos HTML, jQueryUI para tratamento gráfico do CSS3, HTML5 para elaboração das páginas web e pacote de temas e estilos do Bootstrap. Todas essas linguagens e ferramentas de linguagem de programação foram utilizadas em sua versão mais atualizada e estável, sendo importante notar que também são de categoria *open-source* como todo o projeto.

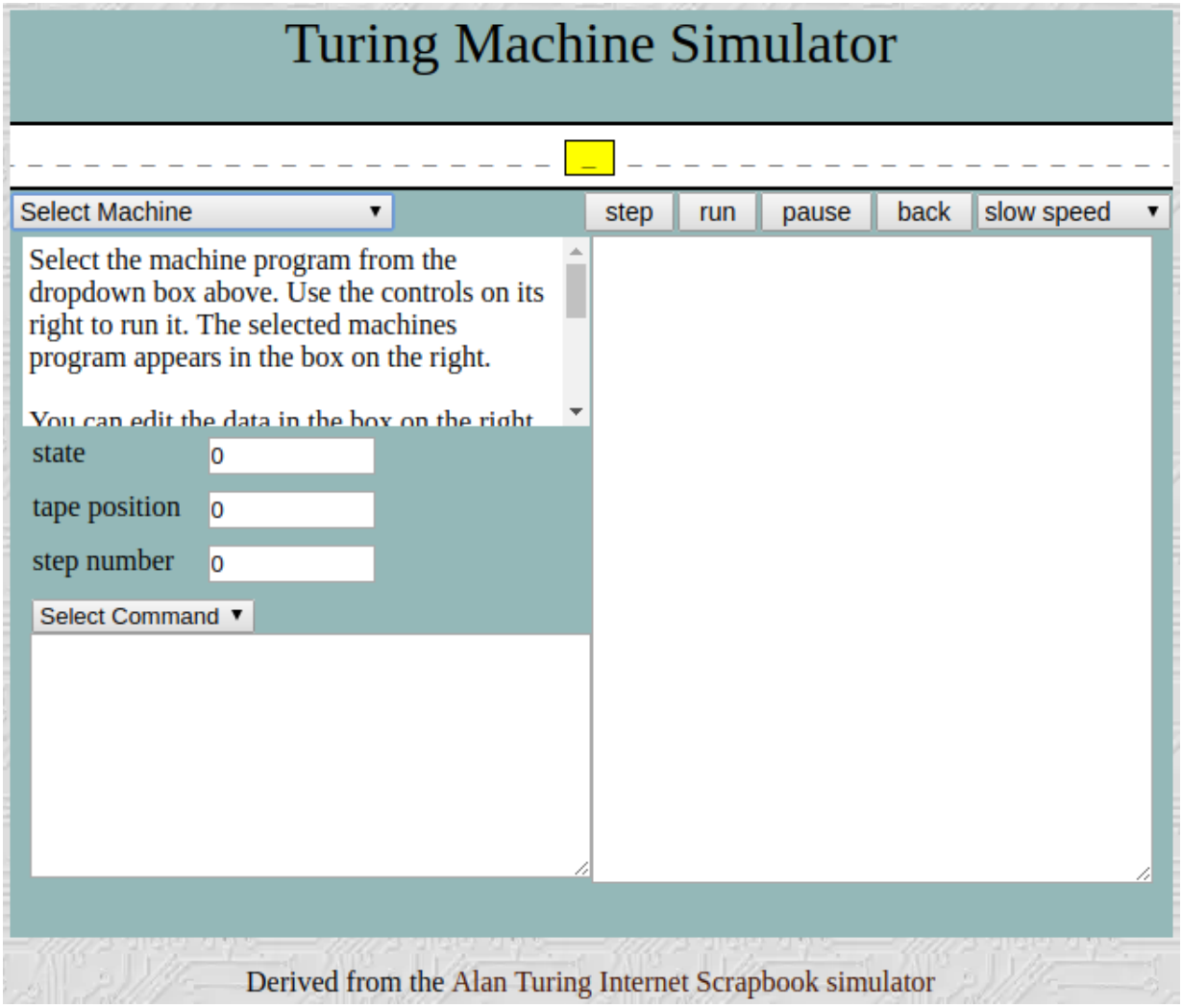
Seguindo para a parte final do projeto, "o teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no

ambiente para o qual foi projetado.” (Dias, 2008). Portanto, serão realizados testes de funcionalidade e testes de usabilidade para garantir que o objetivo principal da aplicação seja alcançado.

3.3 Projetos Relacionados

Como base de desenvolvimento e aplicação das funcionalidades da máquina de Turing foi levado em consideração dois projetos de simulador de máquina de Turing. O primeiro é de autoria de Paul Rendell e teve sua última atualização em junho de 2015. A aplicação funciona bem, embora seja bem simples e tenha poucas funcionalidades. O conceito dos controles de máquina Turing definidos neste projeto como *“Step” (passo a passo), “Run” (iniciar), “Pause” (pausar) e “Back” (voltar)* foram abstraídos para o projeto deste trabalho. Na *Figura 1* abaixo é possível visualizar a tela principal do projeto.

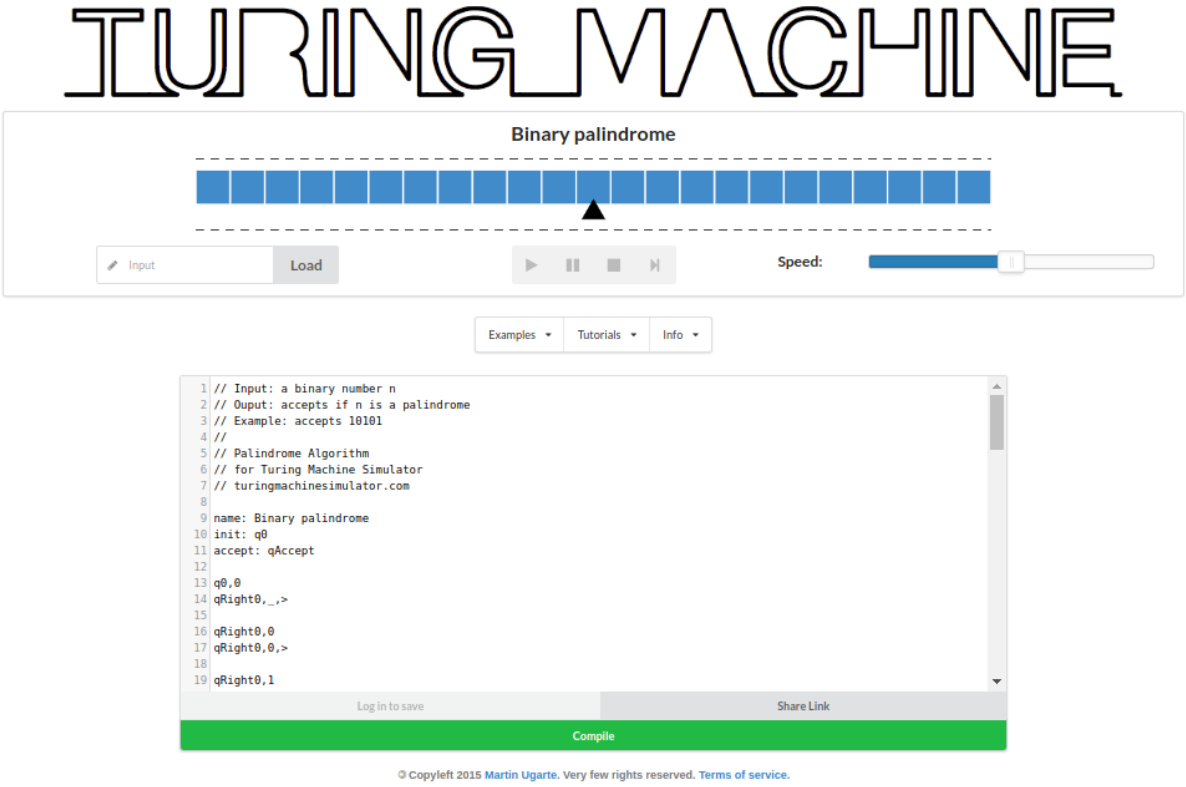
Figura 1 - Turing Machine Simulator, Paul Rendell



Fonte: Elaborado pelo autor

O segundo projeto é de autoria de Martin Ugarte e tem um visual mais sofisticado. Foi deste projeto que foi abstraído o conceito de carregar um máquina de Turing que já estivesse pronta. A aplicação é excelente, porém, como a primeira citada acima, é em inglês e o construção da máquina é feita em cima da sintaxe imposta pelo autor e sem nenhum recurso para auxiliar a criação de estados e transições da máquina, deixando que a usabilidade seja um pouco complexa. A *Figura 2* abaixo mostra um pouco mais da aplicação.

Figura 2 - Turing Machine, Martin Ugarte



Fonte: Elaborada pelo autor

3.4 Desenvolvimento e Resultados

Pensando na usabilidade e em uma leitura mais amigável do que a aplicação, por objetivo, pretende oferecer, dividimos a mesma em quatro seções muito importantes: *Tutorial e Máquina de Turing*, no qual a *Figura 3* abaixo exemplifica.

Figura 3 - Tela Inicial

Fonte: Elaborada pelo autor

- **Tutorial:** *a* seção *Tutorial* tem como objetivo imergir o usuário em um tipo de estudo dirigido resumido, porém cuidadosamente detalhado, para que seja absorvido uma carga de conhecimentos e conceitos básicos sobre a máquina de Turing, antes mesmo de começar a usar o *Simulador* ou o *Construtor*. Neste estudo dirigido será possível aprender basicamente o que é uma máquina de Turing e como usar as seções *Simulador* e *Construtor*, através de slides e um vídeo, para abstrair tais conhecimentos. As figuras *Figura 4, Figura 5, Figura 6, Figura 7, Figura 8, Figura 9, Figura 10, Figura 11, Figura 12 e Figura 13* mostram um pouco melhor do que é a realmente a seção.

Figura 4 - Tutorial – Slides 01

Fonte: Elaborada pelo autor

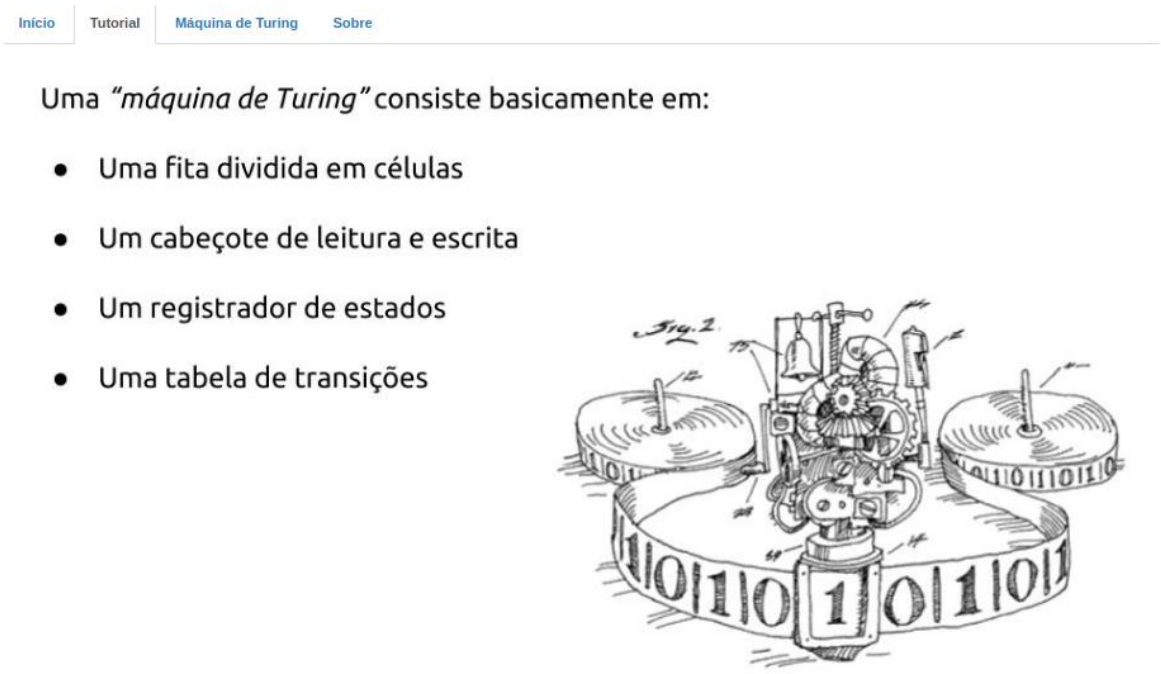
Figura 5 - Tutorial – Slides 02

Fonte: Elaborada pelo autor

Figura 6 – Tutorial – Slides 03

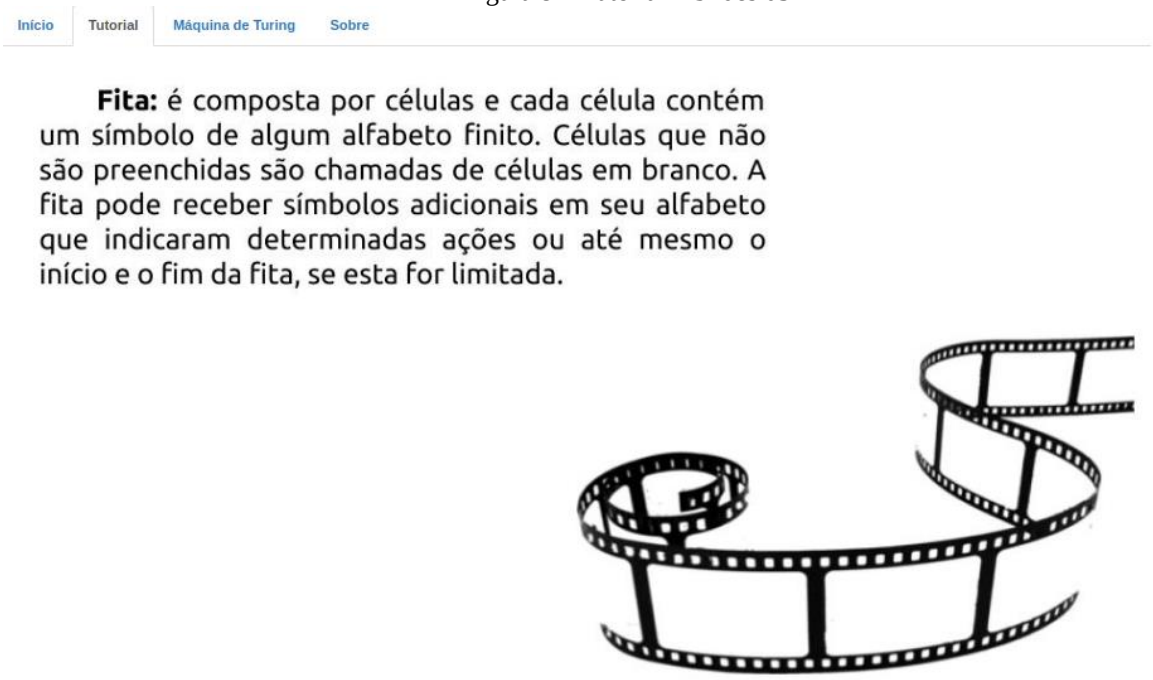
Fonte: Elaborada pelo autor

Figura 7 – Tutorial – Slides 04



Fonte: Elaborada pelo autor

Figura 8 – Tutorial – Slides 05

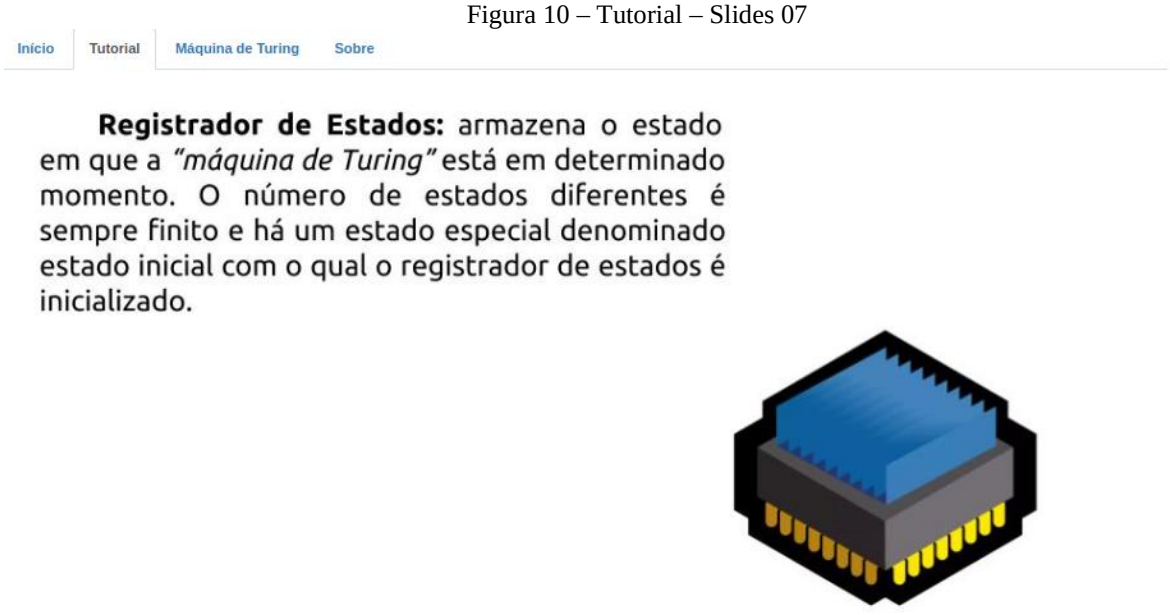


Fonte: Elaborada pelo autor

Figura 9 – Tutorial – Slides 06

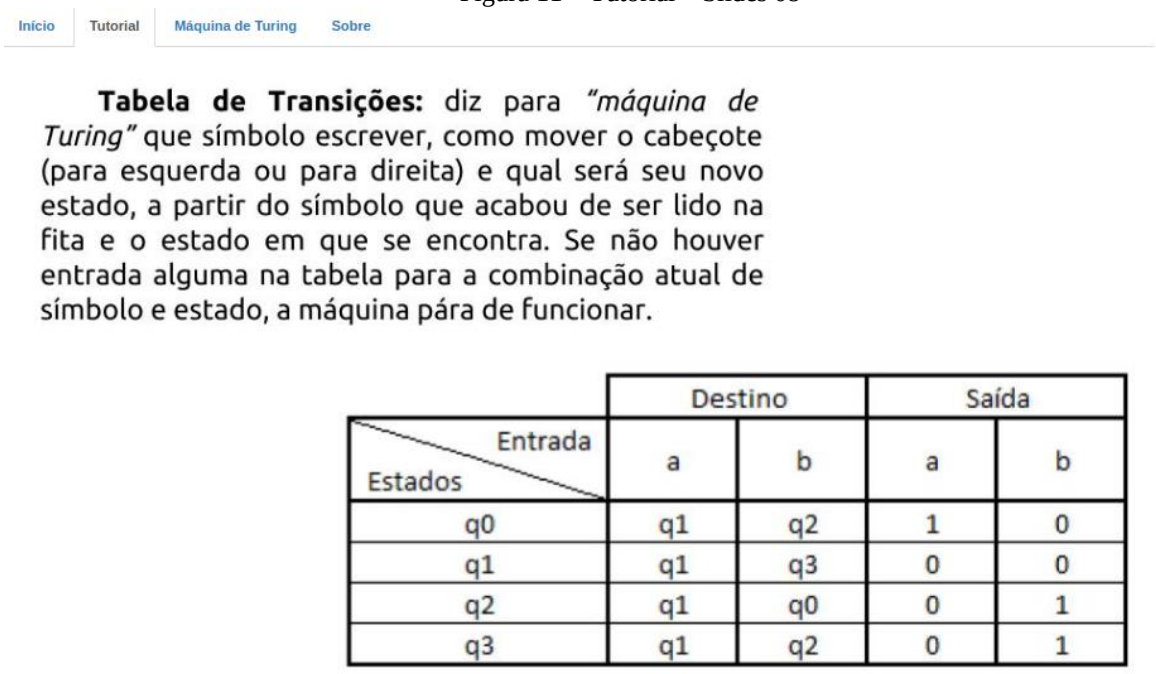
Fonte: Elaborada pelo autor

Figura 10 – Tutorial – Slides 07



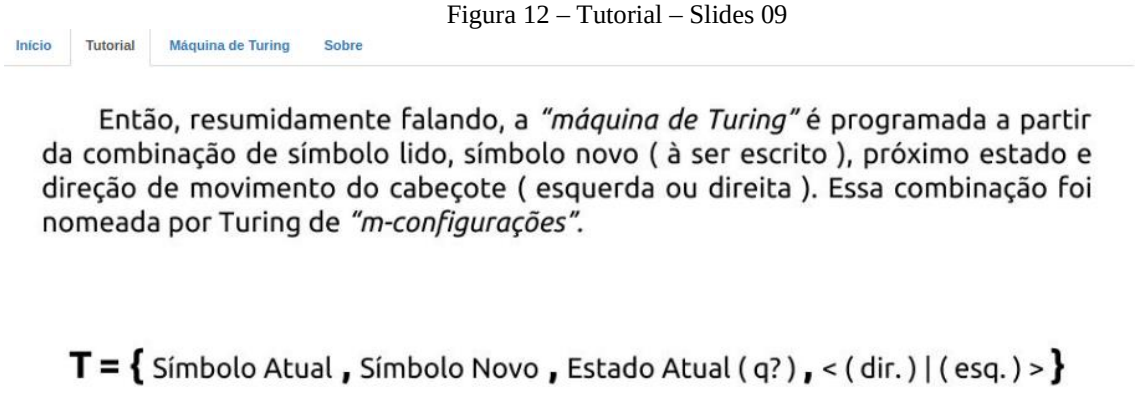
Fonte: Elaborada pelo autor

Figura 11 – Tutorial – Slides 08



Fonte: Elaborada pelo autor

Figura 12 – Tutorial – Slides 09



Fonte: Elaborada pelo autor

Figura 13 – Tutorial – Slides 10

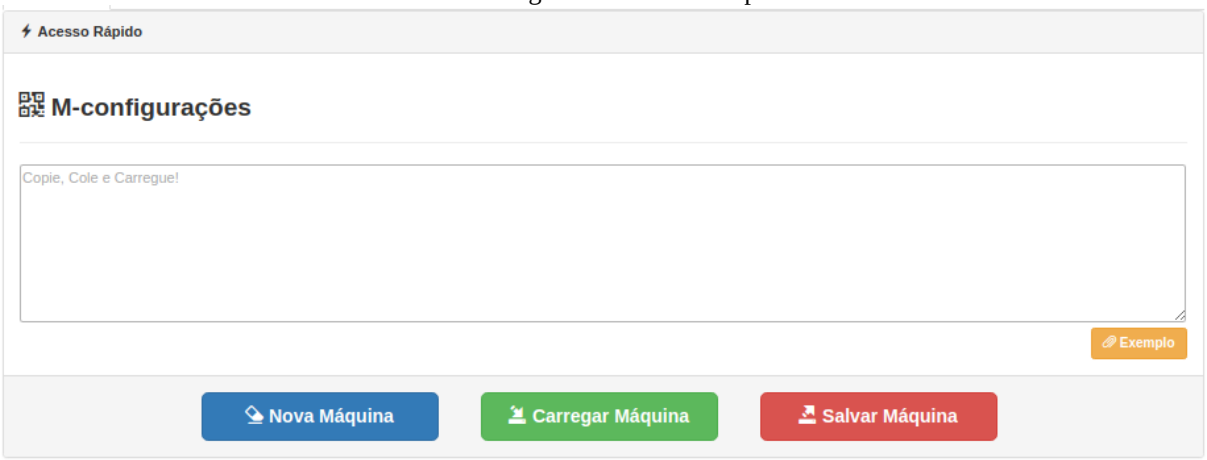
Fonte: Elaborada pelo autor

Figura 14 – Tutorial – Vídeo

Fonte: Elaborada pelo autor

- **Máquina de Turing:** a seção *Máquina de Turing* é dividida em três subseções muito importantes para o entendimento dos conceitos da máquina de Turing.
    - **Acesso Rápido*:*** *a seção Acesso Rápido* tem como objetivo trazer um tipo de "teste rápido", no caso de usuários que já aprenderam ou saibam usar a máquina, ou até porventura não dependem da seção *Construtor* para criar suas *"m-configurações"* (lógicas estados e transições) para máquina. Então, como podemos observar na *Figura 15 abaixo*, a seção é composta por basicamente:

Figura 15 - Acesso Rápido



Fonte: Elaborada pelo autor

- **Exemplo:** um botão para baixar um arquivo de texto (.txt) com uma máquina de Turing programada. Se o conteúdo do arquivo de texto (.txt) for inserido na caixa de texto *"M-Configurações"* e em seguida clicarmos no botão *"Carregar M-Configurações"* a aplicação irá carregar as configurações na seção *Construtor* e montar o escopo de animação da máquina na seção *Simulador*.
- **M-Configurações:** uma caixa de texto para inserir as m-configurações anteriormente programadas;
- **Nova Máquina:** um botão para limpar todas as entradas do *Construtor* e iniciar a criação de uma nova máquina;
- **Carregar Máquina:** um botão para carregar as configurações de máquina de Turing que já foram programadas;

- **Salvar Máquina:** um botão para salvar as configurações de máquina de Turing que foram programadas pelo nas seções *Construtor* e *Simulador*.

- **Construtor:** Na seção *Construtor*, a aplicação permite configurar uma máquina de Turing para uma determinada função a escolha do usuário. A princípio, a seção *Tutorial* já terá ensinado como programar uma máquina de Turing através da criação de estados da máquina e transições de estados na máquina, que segundo Turing são as *"configurações-m"*. O programa de máquina de Turing a ser ensinado inicialmente será algo bem simples como um "detector de sentenças equivalentes", que também está disponibilizado no botão *"Exemplo Pronto"* da seção *Acesso Rápido*. Assim a seção *Construtor* é composta, como mostra a *Figura 16*, abaixo, pelos elementos:

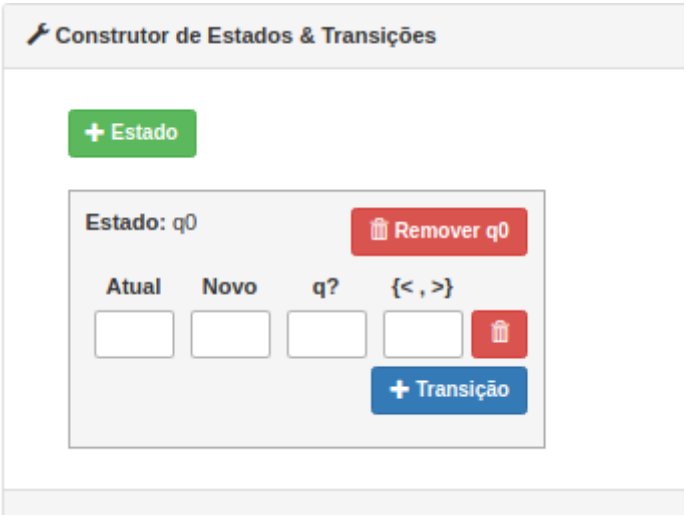
Figura 16 - Construtor

Fonte: Elaborada pelo autor

- **Adicionar Estado:** um botão para adicionar os estados da máquina. Ao adicionar um estado é criado um "*container"* de opções *(caixa de opções)* para configurar o estado;
- **Remover Estado:** um botão para remover o estado e o correspondente *"container"* de opções;
- **Adicionar Transição:** um botão para adicionar uma transição no *"container"* de opções do estado. Ao adicionar uma

transição é inserida uma “linha” com caixas de texto à serem preenchidas para configurar a transição. Estas linhas são uma abstração simplificada das “*tuplas”* da máquina de Turing, que no caso serão compostas, de acordo com a *Figura 17*, por:

Figura 17 – Estados e Transições



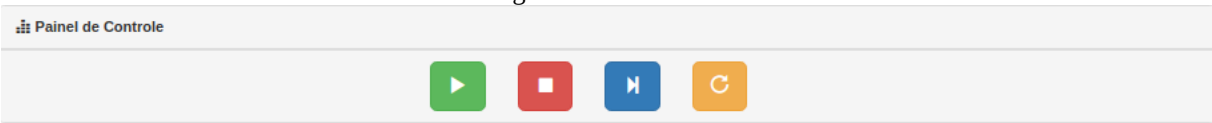
Fonte: Elaborada pelo autor

- **Atual:** uma caixa de texto para inserir o valor atual na fita lido pelo leitor;
- **Novo:** uma caixa de texto para inserir o novo valor que irá sobrescrever o valor atual lido da fita;
- **q?:** uma caixa de texto para inserir o próximo estado após a transição;
- **{< , >}:** uma caixa de texto para ser inserido a direção em que o leitor da máquina deve seguir para ler o próximo valor da fita. A sintaxe deste campo foi definida da seguinte forma: “<” para esquerda e “>” para direita (sem aspas);
- **Remover Transição:** um botão para remover a transição correspondente ao botão.

- ***Simulador*:** Na seção do Simulador *haverá* um espaço para a inserção do estado inicial da máquina e da sequência de dados de entrada que

configuram a fita da máquina. Por fim, após os atributos da máquina estiverem completamente definidos é apenas clicar no botão de iniciar para ver a máquina em funcionamento. O Simulador tem alguns controles de máquina e elementos de tela muito importantes, que podem ser observados na *Figura 18 abaixo*, eles são:

Figura 18 – Painel de Controle



Fonte: Elaborada pelo autor

- **Estado Inicial:** uma caixa de texto para inserir o estado inicial da máquina;
- **Fita da Máquina:** uma caixa de texto para inserir a sequência de valores (qualquer carácter antes definido nas transições da máquina);
- **Iniciar:** um botão com a funcionalidade de dar início aos procedimentos e funcionamento da máquina;
- **Parar:** um botão com a funcionalidade de parar todo o processo que estiver sendo executado pela máquina;
- **Passo a Passo:** um botão com a funcionalidade de executar a máquina de modo que o usuário consiga visualizar literalmente “passo a passo” os processos de transição da máquina e movimentação do leitor na fita;
- **Reiniciar:** um botão com a funcionalidade de reiniciar a máquina para o seu estado inicial anterior ao momento de sua inicialização.
- **Máquina de Turing (Animação):** um espaço reservado na tela onde ocorrerão as animações correspondentes aos processos de transição e movimentação do leitor na fita. Podemos dividir a animação, de acordo com a *Figura 20*, em:

Figura 19 – Visualização dos Processos da Máquina

Fonte: Elaborada pelo autor

- **Estado Atual:** espaço na tela onde é informado o estado atual em que a máquina está processando as transições;
- **Posição do Leitor:** espaço na tela onde é informado a posição em que o leitor está na fita;
- **Resultado:** espaço na tela onde é informado o status da máquina durante e no final da análise da fita;
- **Leitor:** espaço onde é simulado a animação de um cabeçote (olho) de leitura percorrendo a fita;
- **Fita da Máquina:** espaço onde é mostrado os valores da fita e as alterações feitas pelas transições da máquina.

3.5 Código Fonte

Para um melhor entendimento técnico da aplicação esta seção tem como objetivo disponibilizar o código fonte utilizado no desenvolvimento do projeto. O código abaixo é referente ao comportamento dos elementos de interface da aplicação, sejam elementos ou animações gráficas ou criação, remoção e atualização de elementos.

```
var states = 0;

function addState(n) {
    var div = document.createElement('div');
    div.setAttribute('id', 'stateBox-' + n);
    div.setAttribute('class', 'col-xs-6 col-sm-4 stateBox');
    var stateBox = document.createElement('div');
    stateBox.setAttribute('id', 'stateBox-' + n + '-data');

    var transTable = document.createElement('table');
    transTable.innerHTML = '<tr>' +
        '<td>' +
        '<b data-toggle="tooltip" data-placement="top" title="Símbolo
atual">Atual</b>' +
        '</td>' +
        '<td>' +
        '<b data-toggle="tooltip" data-placement="top" title="Novo
símbolo">Novo</b>' +
        '</td>' +
        '<td>' +
        '<b data-toggle="tooltip" data-placement="top" title="Próximo
estado">q?</b>' +
        '</td>' +
        '<td>' +
        '<b data-toggle="tooltip" data-placement="top" title="Direção do
cabeçote de leitura">{< , >}</b>' +
        '</td>' +
        '</tr>';
    stateBox.appendChild(transTable);
    transTable.setAttribute('id', 'transTable-' + n);
    transTable.setAttribute('class', 'transTable');

    div.innerHTML = '<b data-toggle="tooltip" data-placement="top"
title="Estado atual">Estado: </b>q' + n;
    var removeStateButton = document.createElement('button');
    removeStateButton.setAttribute('class', 'removeStateButton btn btn-
sm btn-danger');
    removeStateButton.setAttribute('style', 'margin-bottom: 10px;
float:right;');
    removeStateButton.setAttribute('data-toggle', 'tooltip');
    removeStateButton.setAttribute('data-placement', 'top');
    removeStateButton.setAttribute('title', 'Remover estado');
    removeStateButton.innerHTML = '<span class="glyphicon glyphicon-
trash" aria-hidden="true"></span> <b>Remover q' + n + '</b>';
    removeStateButton.onclick = function() {
        removeState(n);
        removeStateBoxElement(div);
    };

    var addTransitionButton = document.createElement('button');
    addTransitionButton.setAttribute('class', 'addTransitionButton btn
btn-sm btn-primary');
    addTransitionButton.setAttribute('data-toggle', 'tooltip');
    addTransitionButton.setAttribute('data-placement', 'top');
    addTransitionButton.setAttribute('title', 'Adicionar transição');
    addTransitionButton.innerHTML = '<span class="glyphicon glyphicon-
plus" aria-hidden="true"></span><b> Transição</b>';
    addTransitionButton.onclick = function() {
        addTransition(n);
    };
```

```
function addTransition(n) {
    var transTable = document.getElementById('transTable-' + n);
    var tableRow = transTable.insertRow(-1);

    var charSeen = document.createElement('input');
    charSeen.setAttribute('class', 'dataArea');
    charSeen.setAttribute('maxlength', '1');
    charSeen.onchange = limitLength(charSeen);
    charSeen.addEventListener('input', function() {
        if (!((n + '_' + charSeen.value) in ruleSet)) {
            charSeen.disabled = true;
            limitLength(charSeen);
            addRule(charSeen);
        } else {
            charSeen.value = '';
        };
    });
    tableRow.insertCell(0).appendChild(charSeen);

    var charNext = document.createElement('input');
    charNext.onchange = limitLength(charNext);
    charNext.addEventListener('input', function() {
        limitLength(charNext);
        addRule(charNext);
    });
    charNext.setAttribute('class', 'dataArea');
    charNext.setAttribute('maxlength', '1');
    tableRow.insertCell(1).appendChild(charNext);

    var stateNext = document.createElement('input');
    stateNext.setAttribute('class', 'dataArea');
    stateNext.setAttribute('maxlength', '1');
    stateNext.addEventListener('input', function() {
        addRule(stateNext);
    });
    tableRow.insertCell(2).appendChild(stateNext);

    var dirNext = document.createElement('input');
    dirNext.setAttribute('class', 'dataArea');
    dirNext.setAttribute('maxlength', '1');
    dirNext.onchange = limitLength(dirNext);
    dirNext.addEventListener('input', function() {
        limitLength(dirNext);
        addRule(dirNext);
    });
    tableRow.insertCell(3).appendChild(dirNext);

    var removeTransButton = document.createElement('button');
    removeTransButton.setAttribute('class', 'removeTransButton btn btn-
sm btn-danger');
    removeTransButton.setAttribute('data-toggle', 'tooltip');
    removeTransButton.setAttribute('data-placement', 'top');
    removeTransButton.setAttribute('title', 'Remover transição');
    removeTransButton.innerHTML = '<span class="glyphicon glyphicon-
trash" aria-hidden="true"></span>';
    removeTransButton.onclick = function() {
        removeRule(removeTransButton);
        removeTransitionElement(tableRow);
    };
    tableRow.insertCell(4).appendChild(removeTransButton);
    transTable.appendChild(tableRow);
```

```
    return tableRow;
};

function limitLength(element) {
    if (element.value.length > 1) {
        element.value = element.value.substring(element.value.length -
1, element.value.length);
    };
};

function addRule(ta) {
    var cells = ta.parentNode.parentNode.childNodes;
    var stateBox =
ta.parentNode.parentNode.parentNode.parentNode.parentNode;
    var state =
stateBox.getAttribute('id').substring(stateBox.getAttribute('id').indexO
f('-') + 1);
    var charSeen = cells[0].childNodes[0].value;
    var charNext = cells[2].childNodes[0].value;
    var stateNext = cells[3].childNodes[0].value;

    if (stateNext == 'A' || stateNext == 'a') {
        stateNext = -1;
    };

    if (stateNext == 'R' || stateNext == 'r') {
        stateNext = -2;
    };

    var dirNext = cells[4].childNodes[0].value;

    ruleSet[state + "_" + charSeen] = [charNext, stateNext, dirNext];
    console.log(ruleSet);
};

function removeRule(ta) {
    var cells = ta.parentNode.parentNode.childNodes;
    var stateBox =
ta.parentNode.parentNode.parentNode.parentNode.parentNode;
    var state =
stateBox.getAttribute('id').substring(stateBox.getAttribute('id').indexO
f('-') + 1);
    var charSeen = cells[0].childNodes[0].value.substring(0, 1);
    var k = state + '_' + charSeen;
    deleteItem(ruleSet, k);
    console.log(ruleSet);
};

function removeState(n) {
    for (var k in ruleSet) {
        if (k.substring(0, k.indexOf('_')) == n + '') {
            deleteItem(ruleSet, k);
        };
    };
    console.log(ruleSet);
};

    } else {
        obj.splice(k, 1);
```

```
    };
};


function playButton() {
    playMachine();
    drawTuringMachine();
};

function stepByStepButton() {
    stepByStep();
    drawTuringMachine();
};

function stopButton() {
    stopMachine();
    drawTuringMachine();
};

function resetButton() {
    resetMachine();
    drawTuringMachine();
};

function stateMachine() {
    state = document.getElementById('initialState').value;
};

function removeStateBoxElement(tableRow) {
    tableRow.parentNode.removeChild(tableRow);
};

function removeTransitionElement(tableRow) {
    tableRow.parentNode.removeChild(tableRow);
};

function stateBoxUpdate(x) {
    currentState = document.getElementById('initialState').value;
};

function tapeBoxUpdate(x) {
    tape = document.getElementById('tape').value;
};

function clearStateBoxes() {
    document.getElementById('stateBoxes').innerHTML = '';
    states = 0;
    addState(0);
};


function newMachine() {
    clearRules();
    clearStateBoxes();
    resetMachine();
    drawTuringMachine();
};

function saveMachine() {
    resetMachine();
    document.getElementById('tmTextCode').value = outputMachineCoding();
    drawTuringMachine();
```

```
    drawTuringMachine();
};

function loadMachine() {
    newMachine();
    var stateInitial = document.getElementById('tmTextCode').value;
    readMachineCoding(stateInitial);
    document.getElementById('initialState').value = currentState;
    document.getElementById('tape').value = tape;
    drawTuringConstructor();
    drawTuringMachine();
};

function drawTuringConstructor() {
    for (var k in ruleSet) {
        if (k.indexOf('_') > 0) {
            /*
                Montando a Tupla de Transição: {
                    Estado Atual, Símbolo Lido, Próximo Símbolo, Próximo
Estado, Próximo Direção
                }
            */
            addToState = k.substring(0, k.indexOf('_'));
            addCharSeen = k.substring(k.indexOf('_') + 1);
            addNextChar = ruleSet[k][0];
            addNextState = ruleSet[k][1];
            addNextDir = ruleSet[k][2];

            // Se não houver um elemento para receber o estado definido,
criar um...
            stateBox = document.getElementById('stateBox-' +
addToState);
            if (stateBox == null) {
                stateBox = addState(addToState);
            };

            // Criar uma linha para a m-configuração...
            tr = addTransition(addToState);
            tr.cells[0].childNodes[0].value = addCharSeen;
            tr.cells[2].childNodes[0].value = addNextChar;
            tr.cells[3].childNodes[0].value = addNextState;
            tr.cells[4].childNodes[0].value = addNextDir;
        };
    };
};
```

O código abaixo é referente ao comportamento da máquina de Turing implementada no projeto. A o desenvolvimento da aplicação à nível de código obteve ótimo resultado em relação a abstração dos conceitos da máquina de Turing. A seguir podemos ter uma melhor compreensão do código.

```
var ruleSet = {};

var currentState = 0;

var tape = '';

var head = 0;

var timeSpeed = null;

function drawTuringMachine() {
    if (currentState == -1) {
        line0 = '<span class="glyphicon glyphicon-map-marker" aria-
hidden="true"></span><b> Estado Atual: </b>qA' +
'          <span
class="glyphicon glyphicon-eye-close" aria-hidden="true"></span><b>
Posição do Leitor: </b>' + head + 'º';
    } else if (currentState == -2) {
        line0 = '<span class="glyphicon glyphicon-map-marker" aria-
hidden="true"></span><b> Estado Atual: </b>qR' +
'          <span
class="glyphicon glyphicon-eye-close" aria-hidden="true"></span><b>
Posição do Leitor: </b>' + head + 'º';
    } else {
        line0 = '<span class="glyphicon glyphicon-map-marker" aria-
hidden="true"></span><b> Estado Atual: </b>q' + currentState +
'          <span
class="glyphicon glyphicon-eye-close" aria-hidden="true"></span><b>
Posição do Leitor: </b>' + head + 'º';
    };

    if (currentState == -1) {
        line1 = '<span class="glyphicon glyphicon-edit" aria-
hidden="true"></span><b> Resultado: </b>' + '<span class="glyphicon
glyphicon-thumbs-up" aria-hidden="true"></span><font color="#449d44"><b>
APROVADO!</b></font>';
        stopMachine();
    } else if (currentState < -1) {
        line1 = '<span class="glyphicon glyphicon-edit" aria-
hidden="true"></span><b> Resultado: </b>' + '<span class="glyphicon
glyphicon-thumbs-down" aria-hidden="true"></span><font
color="#c9302c"><b> REJEITADO!</b></font>';
        stopMachine();
    } else {
        if ((tape != '' || tape != null) && head != 0) {
            line1 = '<span class="glyphicon glyphicon-edit" aria-
hidden="true"></span><b> Resultado: </b>' + '<span class="glyphicon
glyphicon-search" aria-hidden="true"></span><font color="#286090"><b>
ANALISANDO...</b></font>';
        } else {
            line1 = '<span class="glyphicon glyphicon-edit" aria-
hidden="true"></span><b> Resultado: </b>' + '<span class="glyphicon
glyphicon-hourglass" aria-hidden="true"></span><font color="#286090"><b>
...</b></font>';
        };
    };

    i = 0;
    line2 = '';
    while (i < head) {
        line2 += ' ';
```

```
        i += 1;
    };
    line2 += '        <span
class="glyphicon glyphicon-eye-open" aria-hidden="true"></span>';

    line3 = '<span class="glyphicon glyphicon-barcode" aria-
hidden="true"></span><b> Fita</b>: ' + tape;

    tmAnimation = document.getElementById('tmAnimation');
    tmAnimation.innerHTML = line0 + '<br><br>' + line1 + '<br><br>' +
line2 + '<br>' + line3;
};

function playMachine() {
    stopMachine();
    timeSpeed = setInterval(function() {
        stepByStep();
        drawTuringMachine();
    }, 50);
};

function stopMachine() {
    clearInterval(timeSpeed);
};

function stepByStep() {
    // Quando ultrapassar o tamanho da string de entrada...
    while (tape.length <= head + 1) {
        tape += ' ';
    };

    // Lê o valor da célula atual da fita e guarda no array de regras...
    currentChar = tape.charAt(head);
    nextStep = ruleSet[currentState + '_' + currentChar];

    if (nextStep == null || head < 0) {
        currentState = -2;
        return;
    };

    // Atualiza o valor da célula atual lida pelo leitor da máquina...
    tape = tape.substring(0, head) + nextStep[0] + tape.substring(head +
1);

    // Usa a regra guardada para atualizar o estado atual...
    dir = nextStep[1];
    if (dir == 'a' || dir == 'A') {
        currentState = -1;
    } else if (dir == 'r' || dir == 'R') {
        currentState = -2;
    } else {
        currentState = nextStep[1];
    };

    // Atualiza a posição do leitor da máquina baseado na direção...
    if (nextStep[2] == '>') {
        head += 1;
    };

    if (nextStep[2] == '<') {
```

```
        head -= 1;
    };
};

function resetMachine() {
    currentState = document.getElementById('initialState').value;
    tape = document.getElementById('tape').value;
    head = 0;
};

function clearRules() {
    ruleSet = {};
};

function readMachineCoding(tmTextCode) {
    currentState = tmTextCode.substring(0, tmTextCode.indexOf('\n'));
    tmTextCode = tmTextCode.substring(tmTextCode.indexOf('\n') + 1);

    tape = tmTextCode.substring(0, tmTextCode.indexOf('\n'));
    tmTextCode = tmTextCode.substring(tmTextCode.indexOf('\n') + 1);

    while (tmTextCode.length > 2) {
        var currentTransition = [];
        currentTransition.push(tmTextCode.substring(0,
tmTextCode.indexOf(',')));
        tmTextCode = tmTextCode.substring(tmTextCode.indexOf(',') + 1);

        currentTransition.push(tmTextCode.substring(0,
tmTextCode.indexOf(',')));
        tmTextCode = tmTextCode.substring(tmTextCode.indexOf(',') + 1);

        currentTransition.push(tmTextCode.substring(0,
tmTextCode.indexOf(',')));
        tmTextCode = tmTextCode.substring(tmTextCode.indexOf(',') + 1);

        currentTransition.push(tmTextCode.substring(0,
tmTextCode.indexOf(',')));
        tmTextCode = tmTextCode.substring(tmTextCode.indexOf(',') + 1);

        currentTransition.push(tmTextCode.substring(0,
tmTextCode.indexOf(',')));
        tmTextCode = tmTextCode.substring(tmTextCode.indexOf(',') + 2);

        ruleSet[currentTransition[0] + '_' + currentTransition[1]] =
[currentTransition[2], currentTransition[3], currentTransition[4]];
    };
};

function outputMachineCoding() {
    var aux = currentState + '\n' + tape + '\n';
    for (var k in ruleSet) {
        if (k.indexOf('_') > 0) {
            aux += k.substring(0, k.indexOf('_')) + ',';
            aux += k.substring(k.indexOf('_') + 1) + ',';
            aux += ruleSet[k][0] + ',';
            aux += ruleSet[k][1] + ',';
            aux += ruleSet[k][2] + ',\n';
        };
    };
    return aux;
};
```

O código abaixo é referente a inicialização da máquina feita a partir de chamadas de métodos implementados nos códigos acima.

```
window.onload = function() {
    addState(0);
    stateBox = document.getElementById('initialState');
    stateBox.onchange = stateBoxUpdate(stateBox);
    stateBox.addEventListener('input', function() {
        stateBoxUpdate(stateBox);
    });
    tapeBox = document.getElementById('tape');
    tapeBox.onchange = tapeBoxUpdate(stateBox);
    tapeBox.addEventListener('input', function() {
        tapeBoxUpdate(tapeBox);
    });
    stateBoxUpdate(stateBox);
    tapeBoxUpdate(tapeBox);
    drawTuringMachine();
};

$(document).ready(function() {
    $('#owl-demo').owlCarousel({
        navigation: true, // Show next and prev buttons
        slideSpeed: 300,
        paginationSpeed: 400,
        singleItem: true
            // "singleItem:true" is a shortcut for:
            // items : 1,
            // itemsDesktop : false,
            // itemsDesktopSmall : false,
            // itemsTablet: false,
            // itemsMobile : false
    });
});
```

Todo o restante do código fonte da aplicação, incluindo páginas em HTML5, definições de estilo em CSS3 e outros elementos utilizados no projeto estão disponíveis no link compartilhado do GitHub https://github.com/wellmmeR/tm-web-simulator-unasp.

4 CONCLUSÃO

Concluímos que toda a descoberta de Alan Turing ao redor da “máquina universal” é hoje utilizada das mais diversas formas, modelos e marcas, pois todo o conceito da máquina de Turing hoje é convertido no que chamamos de computadores. Turing, provavelmente, não tinha ideia do quanto o seu trabalho contribuiria para estudos, pesquisas e aplicações científicas. Portanto, o projeto desenvolvido por meio deste trabalho apenas é uma tentativa de contribuir na prorrogação do legado deixado por Turing. A aplicação web do simulador de máquina

Turing é nada mais que uma iniciativa de um projeto maior que um dia poderá vir a ser um tipo de “Espaço de Aprendizado sobre Turing”, onde trabalhos futuros poderão incrementar outras aplicações que possam ajudar no ensino e aprendizagem de outros conceitos deixados por ele.

5 REFERÊNCIAS

American Heritage Dictionary. **AHDictionary of The English Language.** Copyright 2016 Houghton Mifflin Harcourt. All rights reserved. Disponível em: https://ahdictionary.com - Acesso em 30 de agosto de 2016.

BARRETO, JORGE M. **Cálculo Lambda.** UFSC - Instituto de Física. Disponível em: http://www.inf.ufsc.br/~j.barreto/PF/CalLambda.htm - Acesso em 21 de Novembro de 2016.

BOEHM, B.l W. **"Software Process Management: Lessons Learned from History"** in ICSE '87 Proceedings of the 9th International Conference on Software Engineering, 1987.

DIAS N., A. C. **Introdução a Teste de Software**. Engenharia de Software Magazine. Nº 1, 2008.

JACOBSON, I. & Pan-Wei, Ng. **Aspect-Oriented Software Development with Use Cases (Addison-Wesley Object Technology Series)**, Publisher: Addison-Wesley Professional © 2004. ISBN: 0321268881.

LEAVITT, David. **O homem que sabia demais**. Tradução de Samuel Dirceu. 1ª edição. Ribeirão Preto, São Paulo. Editora Novo Conceito Ltda., 2007.

MELO, IZABELA. **"Relembrando conceitos básicos sobre autômatos finitos".** 2011. Departamento de Sistemas & Computação da UFCG, Disponível em: http://www.dsc.ufcg.edu.br/~pet/jornal/marco2011/materias/recapitulando.html - Acesso em 21 de Novembro de 2016.

QUEIROZ, RUY J.G.B. de. **"Problemas Decidíveis e Problemas Indecidíveis: O Legado de Alan Turing"**, 2012. Disponível em: http://www.ufrgs.br/alanturingbrasil2012/presentation-RuyQueiroz-ptBR.pdf - Acesso em 21 de Novembro de 2016.

SETZER, V. W. **“Alan Turing e a Ciência da Computação”.** Depto. de Ciência da Computação da USP. Disponível em: https://www.ime.usp.br/~vwsetzer/Turing-teatro.html. Acesso em 30 de agosto de 2016.

RENDELL, PAUL. **Turing Machine Simulator,** 2015. Disponível em: http://rendell-attic.org/gol/TMapplet/index.htm - Acesso em 24 de Novembro de 2016.

TURING, A. M. **“On Computable Numbers, with an Application to the Entscheidungsproblem”.** Recebido em 28 de Maio de 1936 - Lido em 12 de Novembro de 1936. Proceedings of the London Mathematical Society.

UGARTE, MARTIN. **Turing Machine.** Disponível em: https://turingmachinesimulator.com/ - Acesso em 24 de Novembro de 2016.

WINTERBOTHAM, F. W. **“Enigma - O Segredo de Hitler”.** Biliex, 1978.