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 Aprendizado

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 Aprendizado

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.

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, Lucas e Gabriel. Sem muitas delongas agradeço imensamente a minha fiel companheira 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 meus amados pais por todo apoio, investimento e confiança.

RESUMO

O legado deixado por Alan Turing (1912–1954), com certeza, é 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. No entanto, pensando em simplificar e auxiliar no estudo da máquina de Turing que este trabalho propõe uma aplicação web voltada para o aprendizado dos conceitos e funcionamento da máquina de Turing, incluindo um tutorial e um construtor e simulador de máquina de Turing.

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 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. However, thinking of simplifying and assisting in the study of the Turing machine, this project proposes a web application focused on learning the concepts and functioning of the Turing machine, including a tutorial and a Turing machine builder and simulator.

Keywords: Turing Machine; Simulator; Web Application.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

De acordo com a versão online do *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 seja conhecido o significado da palavra “algoritmo”, ao voltar para o século XIII onde tudo isso estava no escuro da mente de alguns homens ousados, contemplamos uma história por trás do que hoje é chamado de computador.

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, segundo LEAVITT (2007) havia um jovem britânico, por nome Alan Mathison Turing, que talvez por seu constante isolamento social, não estava de nenhum dos lados em relação a solução do *Entscheidungsproblem*, no contrário, o encarava como um mero problema que almejava uma resposta. Talvez pelo motivo claro de Turing não considerar o problema esperando um possível resultado negativo ou positivo, que ele obteve sucesso ao enfrentar de uma maneira totalmente inovadora. Todavia, foi com este 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 o funcionamento da “máquina universal”, e desbanca o problema de decisão.

1.1 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 deixados 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, porém demanda um aprofundamento técnico matemático para compreender certas representações, equações e símbolos usados por Turing para explanar suas ideias. 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.

1.2 OBJETIVOS

Portanto, pensando carinhosamente no legado e todo conhecimento 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 auxiliar na aprendizagem dos fundamentos da teoria da computação deixados por sua descoberta.

1.2.1 Objetivo Geral

Desenvolver uma aplicação web para simplificar e auxiliar no aprendizado dos conceitos e funcionamento da máquina de Turing, de forma que tais sejam abstraídos e aplicados na aplicação de forma coerente.

1.2.2 Objetivos Específicos

- Desenvolver uma estrutura web que facilite o usuário na navegação e usabilidade da aplicação;
- Desenvolver um espaço de aprendizado teórico dos conceitos e funcionamento da máquina de Turing;

- Desenvolver um espaço de construção de estados e funções de transição para facilitar programação da máquina de Turing;
- Desenvolver um espaço para simulação da máquina programada no espaço de construção de estados e funções de transição.

1.3 RECURSOS E VIABILIDADE

Como recurso necessário para a viabilização deste trabalho será utilizado um computador, ou notebook, no qual as ferramentas necessárias para o desenvolvimento e implementação dos objetivos propostos serão instaladas e configuradas , sendo elas:

- Visual Studio Code - versão 1.7.2
- Javascript - versão 1.8.5
- jQuery - versão 3.1.1
- jQuery UI - versão 1.11.4
- Bootstrap - versão 3.3.7

1.4 ORGANIZAÇÃO DO TEXTO

O trabalho a seguir foi dividido em capítulos. O primeiro capítulo apresenta uma breve introdução ao tema, uma justificativa ao projeto proposto, assim como os objetivos, especificando tanto o geral quanto os específicos, e finalizando ao detalhar os recursos e viabilidade do projeto. O segundo capítulo é composto pela relação do referencial teórico utilizado como base do estudo preparatório para o desenvolvimento da aplicação. O terceiro capítulo vem trazendo alguns trabalhos relacionados ao tema do projeto. O quarto capítulo vem abordando a metodologia aplicada ao projeto. O quinto capítulo explica o desenvolvimento da aplicação web, detalhando cada parte implementada. O sexto capítulo será discutido a análise dos resultados em relação aos testes aplicados em cima da aplicação web. O sétimo capítulo é o desfecho do tema, comentando todas as ideias propostas cumpridas ao decorrer do trabalho.

2 REFERENCIAL TEÓRICO

2.1 *Entscheidungsproblem*: O Problema de Decisão

Conforme HILBERT & ACKERMANN (1928), o *Entscheidungsproblem ("Problema de Decisão"),* é resolvido quando conhecemos um procedimento que permite, para qualquer expressão lógica dada, decidir sua validade ou satisfatibilidade. BOOLOS, BURGESS & JEFFREY (2007) completam afirmando que "uma expressão apenas é satisfazível se houver a possibilidade de uma interpretação ou modelo para torná-la verdadeira".

Em 1928, David Hilbert propôs seu décimo desafio à comunidade matemática, chamado *"Problema de Decisão"* ou *"Entscheidungsproblem"*, do original alemão. O desafio se baseava em um certo algoritmo, por assim dizer, que recebe como entrada dados em linguagem formal e dados matemáticos, que outrora retornam ao usuário um sinal de "verdadeiro ou falso" quanto a determinada validação. 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.

QUEIROZ (2012) 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." (QUEIROZ, 2012)*

Antes da questão imposta pelo "*Problema de Decisão"* ser respondida, a noção do significado de algoritmo teria de ser formalmente definida. Tais definições foram feitas, de acordo com LEAVITT (2007) e TURING (1936), por Alonzo Church em 1936 com o conceito de *"calculabilidade efetiva"* baseado em seu "*lambda calculus"* e por Alan Turing, no mesmo ano, com seus conceitos ousados de computabilidade e máquinas de Turing. É importante notar que Turing imediatamente reconheceu equivalência de seus conceitos com os modelos de

computação, além de ter comparado seus resultados obtidos com os de Church, sendo também equivalentes em sua essência.

Seguindo LEAVITT (2007), uma resposta negativa foi dada ao *“Problema de Decisão”*, a partir dos resultados apresentados independentemente por Alonzo Church e Alan Turing em 1936. De um lado, Church provava que não há função computável que decida por duas expressões de cálculo lambda quer elas sejam equivalente ou não. Do outro lado, Turing comprimiu a questão da existência de que um algoritmo ou método geral capaz de solucionar o *“Problema de Decisão”* a uma simples questão de existir um método geral que decida se qualquer máquina de Turing é interrompida ou não. Ou seja, quer um algoritmo seja entendido como sendo equivalente à uma máquina de Turing, e tem resultado negativo como resposta, a questão de existir um algoritmo para o *“Problema de Decisão”* também deve ser negativa.

Aplicando seus conceitos das máquinas de Turing ao *Entscheidungsproblem (“Problema de Decisão”*), TURING (1936) conclui dizendo:

> "*Correspondendo a cada máquina de computação “M”, construímos uma fórmula “Un (M)” e mostramos que, se existe um método geral para determinar se “Un (M)” é provável, então há um método geral para determinar se “M” nunca imprime 0* ".

2.2 Hipótese Turing-Church

De acordo com MENEZES (2000), a máquina de Turing concebida em 1936 equivale à um modelo abstrato de computação, com o objetivo de explorar ao máximo os limites da capacidade de expressar solução de problemas. Os resultados de Turing tratam-se então de uma proposta de definição formal da noção intuitiva de algoritmo. Todo o esforço gerado a partir dos resultados obtidos de trabalhos independentes como o de Alan Turing e o de Alonzo Church serviram como reforço para a *Hipótese*

Turing-Church, no qual diz, basicamente, que a capacidade computacional representada por uma máquina de Turing é o alcance máximo que pode ser atingido por qualquer outro método mecânico ou dispositivo capaz de expressar algoritmos.

2.3 Máquina de Turing

Segundo DIVERIO & MENEZES (2011), a máquina de Turing é um modelo computacional universalmente conhecido e aceito como formalização de algoritmo. Portanto, trata-se de um simples mecanismo que formaliza a ideia de um indivíduo capaz de realizar cálculos.

TURING (1936) descreve que 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 ao dizer:

> *"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". (LEAVITT, 2007)*

Todavia, para um melhor entendimento do conceito de máquina de Turing é interessante dividí-lo três partes: noção intuitiva, noção de máquina e modelo formal.

I. Noção Intuitiva

Baseando-se na interpretação de DIVERIO & MENEZES (2011), Turing partiu do pressuposto de analisar a situação de um indivíduo equipado com um instrumento de escrita e um apagador, no qual realiza cálculos em uma folha de papel dividida em quadrados. De início, é viável supor que a folha de papel contém apenas os dados iniciais de determinado problema. Assim, presumimos que o trabalho deste indivíduo pode ser resumido em sequências de operações simples como:

- Ler um símbolo de um quadrado;
- Modificar um símbolo de um quadrado;
- Mover os olhos para outro quadrado.

Logo que encontrado alguma representação satisfatória para a resposta almejada, os cálculos deste indivíduo são finalizados. Portanto, para tornar viável cada parte deste procedimento, hipóteses a seguir são aceitáveis:

- A natureza limitada e bidimensional da folha de papel não são critérios essenciais para os cálculos. Desde já assumindo que a folha de papel pode consistir de uma fila infinita e seccionada, ou melhor dizendo, dividida em quadrados;
- O conjunto de símbolos pode ser finito, podendo-se utilizar de sequências de símbolos;
- O conjunto de estados da mente do indivíduo durante o processo de cálculo é finito. Notando que dentre esses estados há dois especiais, o “estado inicial” e o “estado final”, no qual correspondem respectivamente ao início e fim do processo de cálculo do indivíduo;
- Todo o comportamento do indivíduo a cada momento é definido apenas pelo estado atual e pelo símbolo para o qual sua atenção está voltada;

- O indivíduo é capaz de observar e modificar o símbolo de apenas um quadrado de cada vez, assim como dar atenção à somente um quadrado adjacente.

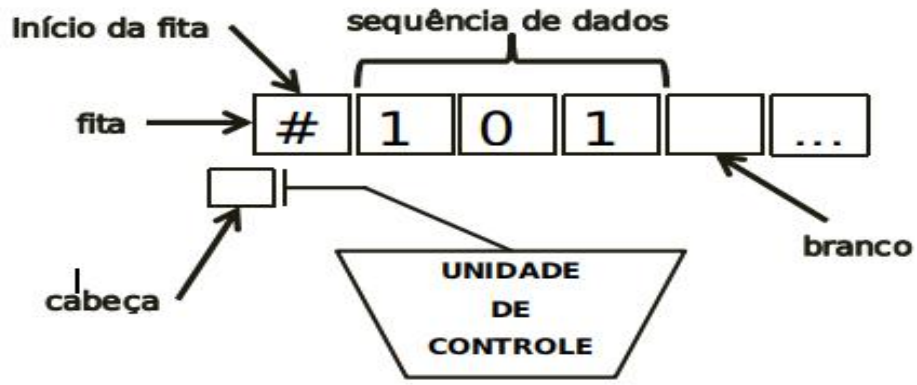
II. Noção de Máquina

Ainda na ideia de DIVERIO & MENEZES (2011), a noção de um indivíduo calculando pode ser facilmente vista como uma máquina, composta por partes muito importantes, como segue:

- Uma *fita*, utilizada simultaneamente como mecanismo de memória e entrada e saída de dados;
- Um *unidade de controle*, que reflete o estado atual da máquina, possui uma *unidade de leitura e escrita*, por vezes chamada de *cabeça*, a qual acessa somente uma célula da fita de cada vez e pode movimentar-se para esquerda ou para direita;
- Um *programa* ou *função de transições*, onde é determinado anteriormente o estado inicial e o estado final da máquina, as ações que irão ser realizadas em determinado estado atual da máquina, incluindo comandos de leitura, escrita e sentido de movimento da cabeça.

A *fita* é finita para esquerda e infinita para direita, estritamente dividida em células, na qual cada um contém um símbolo. Os símbolos, por sua vez, podem pertencer ao alfabeto da máquina, ao alfabeto auxiliar ou até mesmo ser *“branco”* ou *“marcador de início da fita”.* Até então, a sequência de dados, a ser processada pela máquina, preenche as células mais à esquerda após o *marcador de início da fita*, como ilustrado na *Figura 1* abaixo.

Figura 1 - Ilustração de Máquina de Turing



Fonte: Elaborada pelo autor.

A *unidade de controle* possui um número finito e pré-determinado de estados. A *unidade de leitura e escrita*, ou *cabeça*, pode ler somente um símbolo de cada vez e escrever um novo símbolo em cima no símbolo lido. Após a leitura e escrita, a cabeça se move para esquerda ou para direita. Estes comandos são definidos pelo *programa* ou *função de transição*. O *programa* é uma função que determina o símbolo a ser escrito e o próximo estado da máquina de acordo com o estado atual e o símbolo lido.

III. Modelo Formal

De acordo com SIPSER (2007), pode-se dizer de modo formal que uma máquina de Turing é uma representada por 7-upla, ou tupla de 7 elementos:

$$M = (Q, \sum, \Gamma', \delta, q0, qA, qR)$$

Sendo que $Q, \sum, \Gamma'$ são todos conjuntos finitos, onde:

i. Q representa o conjunto de estados da máquina;
ii. $\sum$ representa o alfabeto de entrada da máquina, não contendo o símbolo especial *"branco"* β;
iii. Γ' representa o alfabeto da fita, onde β pertence a Γ' e $\sum$ está incluído em Γ';

iv. ***δ*** é uma função parcial (função de transição) que define que representa as transições da máquina, sendo esta:

δ: (Qatual , Γ'lido) > (Qnovo , Γ'novo , {E,D})

Ou seja, cada transição da máquina pode ser representada com a função acima dado o estado atual ***Qatual*** da máquina e o símbolo lido ***Γ'lido***. Assim, a máquina tem o comportamento de ir para um novo estado ***Qnovo***, escrever por cima do símbolo lido ***Γ'lido*** um novo símbolo ***Γ'novo*** e direcionar a *cabeça (unidade de leitura e escrita)* para esquerda ***{E}*** ou direita ***{D}*** da *fita*.

v. ***q0*** representa o estado inicial da máquina, ou seja, o ponto de partida.

vi. ***qA*** representa o estado de aceitação da máquina, pode ser definido como um estado final da máquina.

vii. ***qR*** representa o estado de rejeição da máquina, também pode ser definido como um estado final da máquina.

Para o funcionamento de uma máquina de Turing é necessário que as funções de transição estejam bem definidas. DIVERIO & MENEZES (2011) explicam que cada função de transição considera o estado atual ***Qatual*** e o símbolo lido ***Γ'lido*** na fita para determinar o novo estado ***Qnovo***, o símbolo à ser escrito na fita ***Γ'novo*** e o sentido de movimento ***{E,D}*** da cabeça. Com base na perspectiva de DIVERIO & MENEZES (2011), o processamento de uma máquina de Turing ***M = (Q, ∑, Γ', δ, q0, qA, qR)*** para uma palavra, ou sequência de símbolos, ***W*** consiste na sucessiva na sucessiva aplicação da função de transição a partir do estado inicial ***q0*** e da cabeça posicionada na célula mais à esquerda da fita, até ocorrer uma condição parada, seja esta condição um estado de aceitação ***qA*** ou um estado de rejeição ***qR***. O processamento de ***M*** para a entrada de dados ***W*** pode parar ou ficar em *loop* infinito.

DIVERIO & MENEZES (2011) concluem observando que dentre as mais relevantes literaturas que tratam sobre o assunto de máquina de Turing, pode haver diversas variações sobre na adoção de uma definição para máquina de Turing. Da perspectiva formal, tais variações são provenientes do modo como é expressado a tupla de configuração de uma máquina de Turing. Da perspectiva prática, as variações mais significativas estão relacionadas às características da fita e ao movimento da cabeça, como por exemplo:

- Inexistência do marcador de início e fim da fita, onde no lugar do símbolo de início e fim da fita é sobreposto, respectivamente, o primeiro e o último símbolo de entrada da fita. Neste caso, é de extrema cautela manter o controle do movimento da cabeça ao atingir os limites da fita.
- Cabeça não se move quando está lendo ou escrevendo algo na fita. Esta variação tem como objetivo simplificar a especificação da função de transição, ou até mesmo reduzir o número de transições necessárias.

2.4 Teoria dos Autômatos

Segundo SIPSER (2007), a teoria dos autômatos lida diretamente com as definições e propriedades de modelos de computação. Tais modelos cumprem um papel em inúmeras áreas aplicadas da Ciência da Computação, como explica SIPSER (2007):

> *"Um modelo, chamado autômato finito, é usado para processamento de texto, compiladores, e projeto de hardware." (SIPSER, 2007).*

Para SIPSER (2007), a teoria dos autômatos é um excelente tópico apropriado para iniciar estudos sobre teoria da computação. Pois, as teorias de computabilidade e complexidade pedem por uma definição precisa de um computador e a teoria dos autômatos permite praticar com

definições formais de computação à medida que é introduzido conceitos relevantes à outras áreas não-teóricas da Ciência da Computação.

2.5 Autômatos Finitos

Em uma abordagem, recapitulando os conceitos básicos de autômatos finitos, MELO (2011) ao indagar se já paramos para pensar em como funcionam as portas automáticas, que abrem e fecham a partir de sensores, ou até mesmo as lavadoras de louça e roupa, termômetros eletrônicos, relógios digitais, calculadoras e máquinas de venda automática, descreve que todos esses dispositivos eletromecânicos têm um controlador que nada mais é do que um autômato finito.

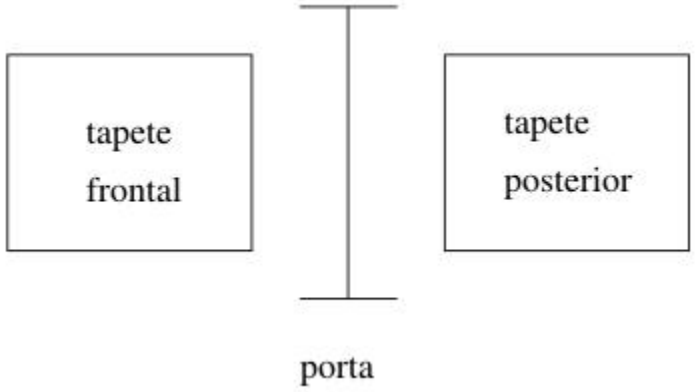
De acordo com MELO (2011, apud SIPSER, 2007), além de dispositivos controladores, os autômatos finitos também são extremamente relevantes no reconhecimento de padrões em dados, como: análise léxica de um compilador, projeção de uma nova linguagem para uma aplicação específica, ou seja, criação de compiladores, processamento de voz e muitos outras aplicações.

Segundo SIPSER (2007), um autômato finito é o modelo computacional mais simples e que também pode ser chamado de máquina de estados finitos, onde faz-se relação aos conceitos de máquina de Turing. Eles são exemplos de modelos de computadores com uma quantidade de memória extremamente limitada, como é o caso dos dispositivos citados como exemplo por MELO (2011).

SIPSER (2007) traz como exemplo de autômato finito um controlador para uma porta automática, frequentemente postas em entradas e saídas de supermercados, shoppings, lojas, etc. Essas portas automáticas abrem deslizando quando um indivíduo está se aproximando e fecham na ausência de tal. Geralmente, uma porta automática tem um tapete na frente para detectar a presença de uma pessoa que está próxima a atravessar a passagem. Um outro tapete está localizado atrás da passagem de modo que o controlador pode manter a porta aberta o

tempo suficiente para que a pessoa atravesse toda a passagem e também de modo que a porta não atinja alguém que está atrás no momento que ela abre. Vejamos na *Figura 2* um exemplo:

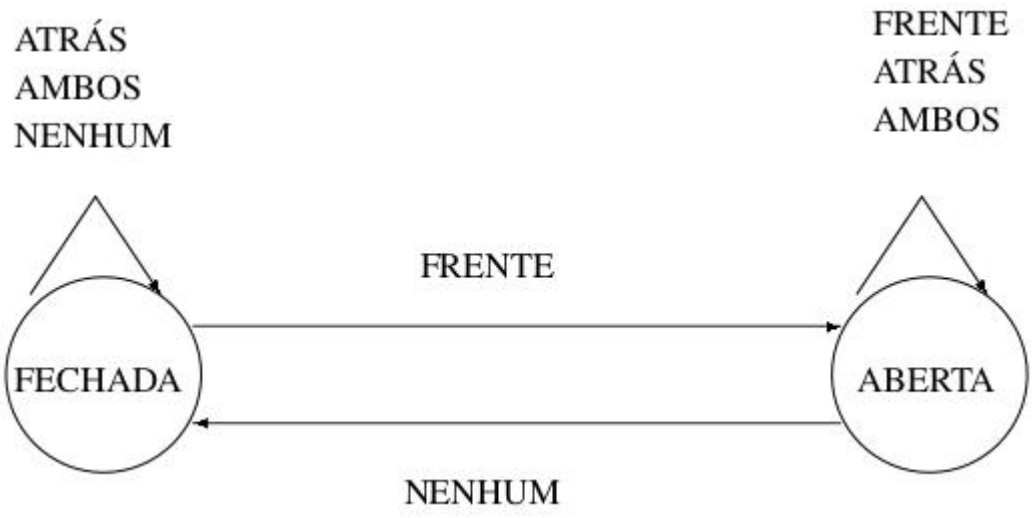
Figura 2 - Conceito de Autômato Finito em uma Porta Automática



Fonte: SIPSER (2007)

De acordo com SIPSER (2007), o controlador da porta automática está em um dos dois estados, como mostra a *Figura 3*:

Figura 3 - Implementação do Autômato Finito de uma Porta Automática



Fonte: SIPSER (2007).

- “**ABERTA**” ou “**FECHADA**”, representando então a condição atual da porta em determinado momento.

Como demonstrado nas *Figura #* a seguir, existem quatro condições possíveis para o controlador:

- “**FRENTE**”: representado uma ou mais pessoas que está pisando no tapete na frente da passagem;
- “**ATRÁS**”: representando uma ou mais pessoas que está pisando no tapete após a passagem;
- “**AMBOS**”: representando as pessoas que estão pisando em ambos os tapetes;
- “**NENHUM**”: representando que ninguém está pisando em qualquer dos tapetes.

Ainda no exemplo de SIPSER (2007), o controlador da porta automática move de estado para estado, dependendo da entrada que ele recebe. Ou seja, de acordo com a *Figura 4*, o controlador tem as seguintes possibilidades:

Figura 4 - Tabela de Transição do Autômato Finito de uma Porta Automática

estado	NENHUM	FRENTE	ATRÁS	AMBOS
FECHADA	FECHADA	ABERTA	FECHADA	FECHADA
ABERTA	FECHADA	ABERTA	ABERTA	ABERTA

Fonte: SIPSER (2007).

a) Para o estado “**FECHADA**” e recebendo uma entrada “**NENHUM**” ou “**ATRÁS**”, ele permanece no estado “**FECHADA**”;
b) Se a entrada “**AMBOS**” é recebida, ele permanece “**FECHADA**” porque a porta pode estar em risco de atingir alguém sobre o tapete de trás.

c) Se a entrada “**FRENTE**” chegar, ele move para o estado “**ABERTA**”.
d) Para o estado “**ABERTA**”, se a entrada “**FRENTE**”, “**ATRÁS**”, ou “**AMBOS**” é recebida, ele permanece no estado “**ABERTA**”;
e) Se a entrada “**NENHUM**” chegar, ele retorna ao estado “**FECHADA**”.

Segundo SIPSER (2007), os autômatos finitos assim como a máquina de Turing podem ser descritos formalmente, seguindo praticamente o mesmo modelo de tupla, porém com algumas simplificações, ou seja, um autômato finito é definido por um 5-tupla, ou tupla de 5 elementos:

$$A = (Q, \sum, \delta, q0, F)$$

Sendo que ***Q*** *e* **∑** são conjuntos finitos, onde:

i. ***Q*** representa o conjunto de estados do autômato;
ii. **∑** representa o alfabeto de entrada do autômato;
iii. ***δ*** é uma função de transição, que define o comportamento do autômato para determinada estado e determinada entrada, sendo esta:

$$\delta: (Qatual , \sum lido) > Qnovo$$

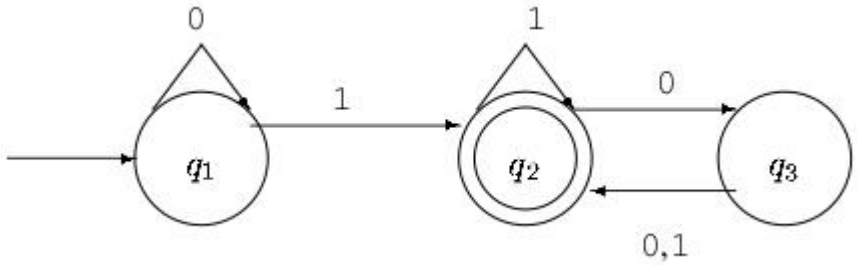
Ou seja, cada transição do autômato pode ser representada com a função acima dado o estado atual ***Qatual*** da máquina e o símbolo lido ***∑lido***. Assim, o autômato tem o comportamento de ir para um novo estado ***Qnovo***.

iv. ***q0*** representa o estado inicial do autômato, ou seja, o ponto de partida.
v. **F** representa os estados finais, sejam eles estado aceitação ou estado de rejeição.

Como exemplo de SIPSER (2007), pode-se usar a definição formal para descrever autômatos finitos individualmente especificando cada uma

das partes descritas na 5-tupla. Assim, tomando como exemplo o autômato finito ***A1 = (Q, ∑, δ, q1, F)*** da *Figura 5* abaixo, e o definindo formalmente, tem-se:

Figura 5 - Exemplo de Autômato Finito no Modelo Formal



Fonte: SIPSER (2007).

- ***Q = {q1 , q2 , q3}***;
- ***∑ = {0, 1}***;
- ***δ*** é descrita pela *Figura 6* abaixo:

Figura 6 - Tabela de Transição de um Autômato Finito Formal

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

Fonte: SIPSER (2007).

- ***q1*** é o estado inicial;
- ***F = {q2}***.

2.6 Javascript

A linguagem *front-end Javascript* foi criada por um programador da *Netscape* em 1995, originalmente chamado de *Livescript* e posteriormente

tendo seu nome mudado para *Javascript*. DUCKETT (2014) comenta que no inicio a linguagem tinha o objetivo principal de garantir a segurança de alguma da validação da entrada, que anteriormente eram deixadas para as linguagens do lado do servidor, como o Perl. FLANAGAN & FERGUSON (2002) introduzem o *Javascript* dizendo:

> *"Javascript foi originalmente implementado como parte dos navegadores web para que scripts pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido." (FLANAGAN & FERGUSON, 2002).*

De acordo com o site PYPL (2016), o *Javascript* é atualmente a quinta linguagem de programação mais utilizada no mundo e em constante ascensão, sendo a principal para *client-side* em navegadores web. Começou também a ser bastante utilizada do lado do servidor através de ambientes como o *Node.js*.

2.7 jQuery

A biblioteca intitulada *jQuery* é um simplificador da linguagem *Javascript* usada para interagir com o *HMTL* com intuito de melhorar a experiência do usuário. Conforme o site oficial do *jQuery*:

> *"jQuery é uma biblioteca Javascript rápida, compacta e rica em recursos. Ele torna as coisas como manipulação HTML, manipulação de eventos, animação e chamadas Ajax muito mais simples com uma API fácil de usar que funciona em uma infinidade de navegadores. Com uma combinação de versatilidade e extensibilidade, jQuery mudou a maneira pela qual milhões de programadores escrevem Javascript." (jQuery, 2016).*

2.8 HTML5

De acordo com o site oficial *W3C*, que desde o princípio está junto com *WhatWG*, do inglês "*Web Hypertext Application Technology Working Group"*, ou *Grupo de Trabalho de Tecnologia de Aplicação de Hipertexto da Web*, trabalhando para a evolução do *HTML*, pode-se definir a linguagem de marcação como:

> *"HTML é a linguagem de marcação principal da internet. Originalmente, o HTML foi projetado primeiramente como uma linguagem para descrever semanticamente originais cientificos. Seu design geral, no entanto, permitiu que ele fosse adaptado, ao longo dos anos subsequentes, para escrever uma série de outros tipos de documentos e até mesmo aplicações web." (W3C, 2016).*

2.9 Bootstrap

De acordo com o blog oficial do *Twitter*, o *framework* teve início em um projeto da rede social *Twitter* para atender demandas internas, e ao ser liberado para a comunidade acabou se tornando um dos *frameworks open-source* mais populares do mundo. Segundo o blog oficial do *Twitter*:

> "O *Bootstrap* é o *framework* mais popular do mundo para a criação de sites e aplicativos responsivos e móveis. Dentro você encontrará *HTML*, *CSS* e *Javascript* de alta qualidade para tornar o início de qualquer projeto mais fácil do que nunca." (Twitter, 2011).

2.10 CSS

Segundo o site oficial W3C, que possui os direitos do *CSS*, a linguagem de folhas de estilo para apresentação de documentos em linguagens de marcação é um simples mecanismo para adicionar estilo como fontes, cores e espaçamento em ambientes web. De acordo com LIE & BOS (2005) sobre a facilidade do uso do *CSS*:

> *“Quase todos os navegadores hoje em dia suportam CSS e muitas outras aplicações também. Para escrever CSS, você não precisa de mais do que um editor de texto, mas há muitas ferramentas disponíveis que o tornam ainda mais fácil.” (BOS, 2016).*

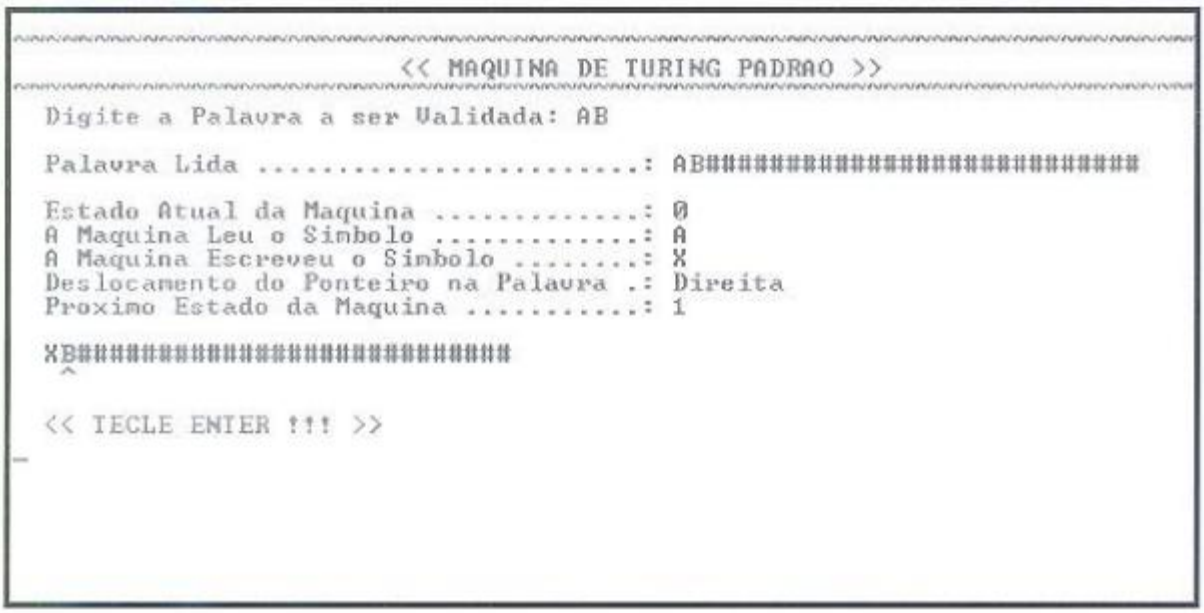
3 TRABALHOS RELACIONADOS

3.1.1 Um Simulador Para a Máquina de Turing

De acordo com GARCIA & LIMA (2003), no trabalho são introduzidos o conceito de conjuntos, alfabetos, palavras e linguagens através de um simulador para a máquina de Turing, assim chamada por ter sido criada por Alan M. Turing, que em 1936 mostrou a comunidade científica a possibilidade de se construir um modelo computacional suficientemente genérico, em sua essência, capaz de implementar qualquer “função computável”.

Portanto, o trabalho expõe o desenvolvimento de um simulador de máquina de Turing para ambiente DOS. O simulador tem por objetivo validar palavras a partir da leitura do autômato fornecida por um arquivo padrão, informando ao usuário se esta é aceita ou rejeitada mostrando passo à passo as transições do autômato, por fim auxiliando a aprendizagem dos alunos de Ciência da Computação. Abaixo é possível ver um das telas da aplicação de GARCIA & LIMA (2003) através da *Figura 7.*

Figura 7 - Um Simulador Para a Máquina de Turing



Fonte: GARCIA & LIMA (2003).

3.1.2 Aplicações Web: Turing Machine Simulator

A primeira aplicação web relacionada ao projeto deste trabalho é de autoria de MORPHETT (2016). A aplicação é um simulador de máquina de Turing que de acordo com MORPHETT (2016) é operada a partir dos seguintes passos:

i. Carregue um dos programas de exemplo, disponíveis na aplicação, ou escreva o seu próprio na área do programa da máquina de Turing (uma espaço que lembra um bloco de notas com linhas enumeradas).
ii. Digite algo na área “*Initial Input*”, no qual este será escrito na fita inicialmente como entrada para a máquina.
iii. Clique em “*Reset*” para inicializar a máquina.
iv. Clique em “*Run*” para iniciar a máquina de Turing e executá-la até que ela pare em algum estado de aceitação ou rejeição.
v. Clique em “*Pause*” para interromper a máquina Turing enquanto ela estiver em execução. Alternativamente, clique em “*Step*”

para executar uma única etapa da transição da máquina de Turing.

vi. Clique em “*Reset*” para restaurar a máquina de Turing para seu estado inicial para que ela possa ser executada novamente.

A aplicação de MORPHET (2016) tem como sintaxe de programação da máquina de Turing os seguintes parâmetros:

a) Cada linha deve conter uma tupla do formulário: '<estado atual> <símbolo atual> <novo símbolo> <direção> <novo estado>';

b) Pode-se usar qualquer número ou palavra para <estado atual> e <novo estado>, por exemplo: 1, 0, a, estado1, q1, etc. Os rótulos de estado diferenciam maiúsculas de minúsculas;

c) Pode-se usar qualquer caractere para <símbolo atual> e <novo símbolo>, ou “_” (*underline*) para representar um caractere em branco. Os símbolos também diferenciam maiúsculas de minúsculas.

d) <direção> deve ser “L”, “R” ou “*”, indicando “mover para a esquerda”, “mover para a direita” ou “não se mover”, respectivamente.

e) Qualquer coisa depois de um “;” (ponto e vírgula) é considerado um comentário e é ignorado.

f) A máquina somente para quando atinge qualquer estado que comece com “halt” (parada), por exemplo: “halt”, “halt-accept” (parada-aceita) ou “halt-reject” (parada-rejeitada).

g) “*” Pode ser usado como um caractere “coringa” em <símbolo atual> ou <estado atual> para corresponder a qualquer caractere ou estado.

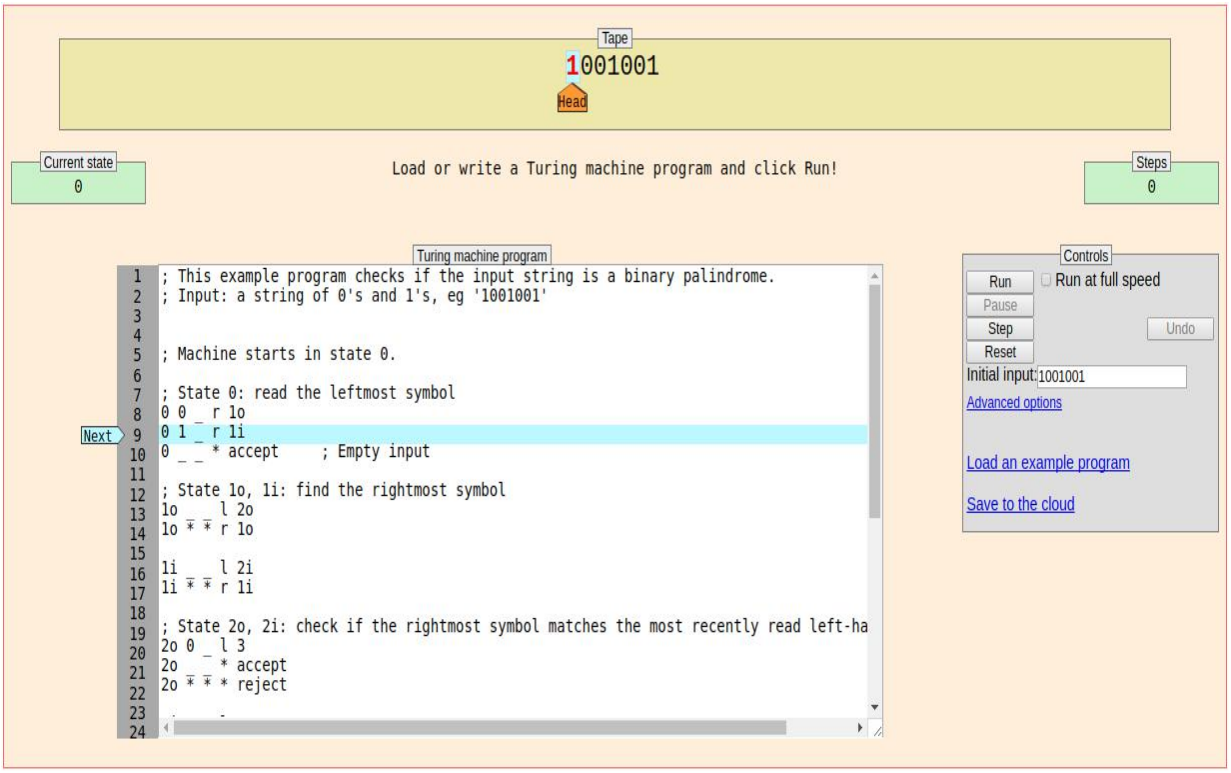
h) “*” Pode ser usado em <novo símbolo> ou <novo estado> para significar “não muda”.

i) “!” Pode ser usado no final de uma linha para definir um ponto de interrupção, por exemplo: “1 a b r 2!”. A máquina pausará automaticamente após a execução desta linha.

j) Pode-se especificar a posição inicial para a cabeça usando “*” na entrada inicial.

Segundo MORPHETT (2016), a aplicação foi escrita utilizando linguagem HTML e Javascript com jQuery, e encontra-se disponível no site oficial da aplicação: http://morphett.info/turing/turing.html, ou no link de versionamento do GitHub: https://github.com/awmorp/jsturing. A *Figura 8* abaixo traz uma noção de como é a simulador.

Figura 8 - Turing Machine Simulator, de Anthony Morphett

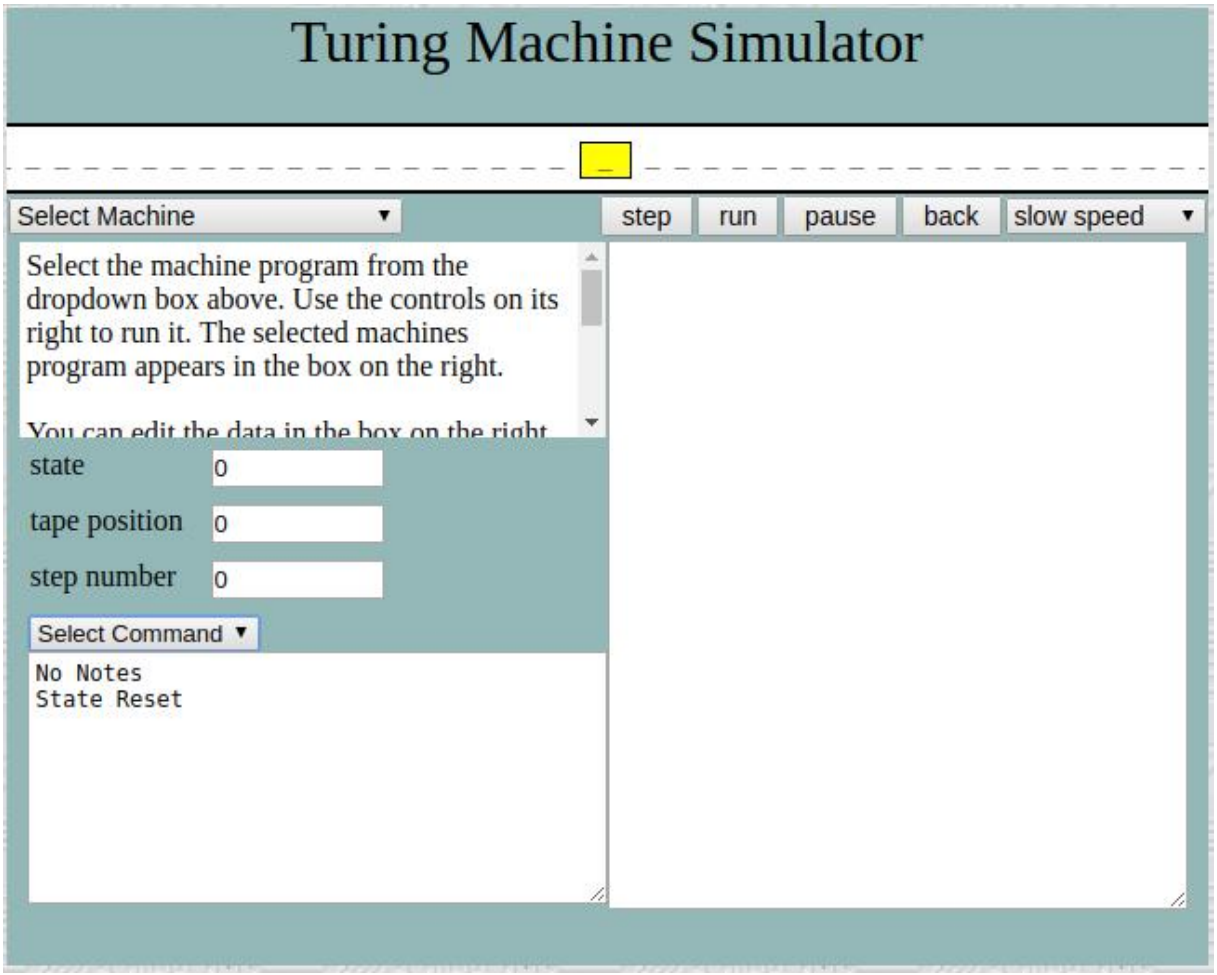


Fonte: MORPHETT (2016).

A segunda aplicação web relacionada ao projeto deste trabalho é de autoria de RENDELL (2015). A aplicação tem poucas informações no que se diz relacionado à como usar, tutorial, sintaxe, etc. Porém traz recursos muito semelhantes a aplicação de MORPHETT (2016). Tendo como funcionalidade adicional um controlador de velocidade para a máquina de Turing, no qual irá ditar o tempo de execução dos processos em relação a

animação realizada na parte superior. A *Figura 9* abaixo mostra de relance a aplicação de RENDELL (2015).

Figura 9 - Turing Machine Simulator, de Paul Rendell



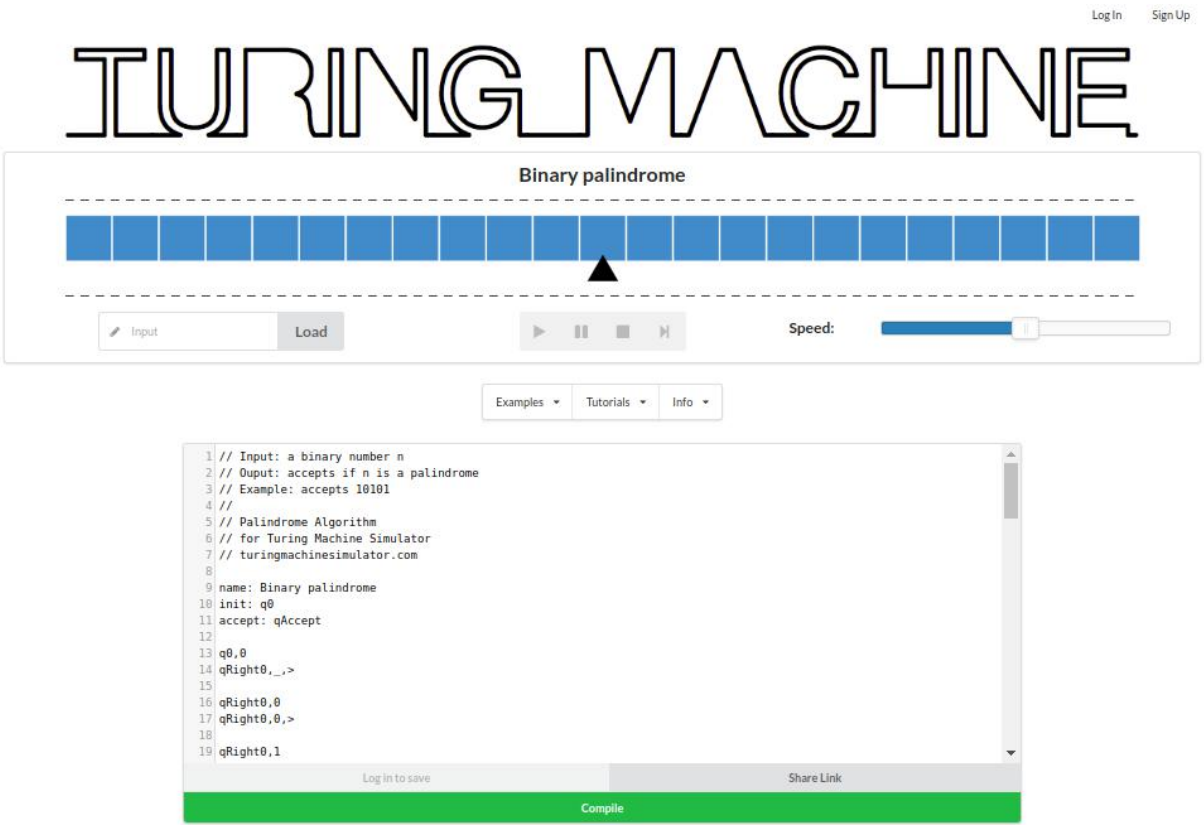
Fonte: RENDELL (2015).

A aplicação de RENDELL (2015) é usada como base de referência do livro *"Alan Turing: O Enigma"*, de Andrew Hodges (2014), e pode ser acessada através do site oficial da aplicação pelo link: http://rendell-attic.org/gol/TMapplet/index.htm.

Por fim, a terceira e última aplicação web, no entanto, não menos relevante, é de autoria de MATTE & UGARTE (2016). O simulador de MATTE & UGARTE (2016) foi projetado e desenvolvido para ser uma plataforma em que o usuário possa programar, compilar, armazenar e

compartilhar de máquinas de Turing. De acordo com informações do site oficial da aplicação de MATTE & UGARTE (2016), algumas universidades ao redor do mundo estão fazendo uso da aplicação para demonstrar os conceitos da máquina de Turing. O grande diferencial da aplicação de MATTE & UGARTE (2016) para as anteriores, além do visual muito mais amigável e usual, é o sistema de login e armazenamento das máquinas de Turing implementadas, além de uma seção com um tutorial técnico e ligeiramente explicativo. A *Figura 10* ilustra com é a aplicação de MATTE & UGARTE (2016).

Figura 10 - Turing Machine Simulator, de Martin Ugarte & José Antônio Matte



Fonte: MATTE & UGARTE (2016).

A simulador de MATTE & UGARTE (2016) pode ser acessado através do site oficial da aplicação pelo link: https://turingmachinesimulator.com/.

4 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, que de acordo com BOEHM (1987) é comumente adotado pela facilidade de implementação e organização do cronograma de atividades. 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 aprendizado dos conceitos e funcionamento da máquina de Turing.

Devido à uma melhor portabilidade do projeto em geral, mais apropriadamente do código-fonte da aplicação, será utilizado o editor de código intitulado Visual Studio Code, em sua versão 1.7.2, 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, em sua versão 1.8.5, com a biblioteca jQuery, na versão 3.1.1, para controle e tratamento de elementos HTML5, o jQueryUI, na versão .11.4, para controle e tratamento de folhas de estilo do CSS3, HTML5 para construção de páginas web, além do pacote de temas e estilos do Bootstrap, na versão 3.3.7. Sendo importante notar que todas as ferramentas utilizadas no desenvolvimento da aplicação são de categoria *open-source* como também o sistema operacional Linux Ubuntu, em sua versão 16.10.

Seguindo para a parte final da aplicação, de acordo com DIAS (2008), 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.” Portanto, serão realizados testes de funcionalidade e testes de usabilidade para garantir que o objetivo principal da aplicação seja alcançado.

5 DESENVOLVIMENTO

5.1 Aplicação Web: Simulador de Máquina de Turing

Pensando na usabilidade e em uma leitura mais amigável do que a aplicação, por objetivo, pretende oferecer, dividimos a mesma em seções muito importantes: *Tutorial e Máquina de Turing,* no qual a *Figura 11* abaixo exemplifica.

Figura 11 - Tela inicial da Aplicação

Fonte: Elaborado pela autor.

I. **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* da seção *Máquina de Turing*. 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. A figura *Figura 12* mostra um pouco melhor do que se trata realmente a seção.

Figura 12 - Tutorial da Aplicação

Fonte: Elaborado pelo autor.

II. **Máquina de Turing:** a seção *Máquina de Turing* é dividida em três subseções muito importantes para o entendimento dos conceitos da máquina de Turing.

a) **Acesso Rápido*:*** *a subseção Acesso Rápido* tem como objetivo trazer um tipo de “teste rápido”, no caso de usuários que já aprenderam ou saibam usar a aplicação, ou até porventura não dependem da seção *Construtor* para construir máquinas de Turing. Então, como podemos observar na *Figura 13 abaixo,* a seção é composta por basicamente:

Figura 13 - Acesso Rápido da Aplicação

Fonte: Elaborado pelo autor.

1) **Exemplo:** um botão para baixar um arquivo de texto (.txt) com uma máquina de Turing programada de exemplo, no caso, a máquina exemplo é programada para validar palavras iguais. Se o conteúdo do arquivo de texto (.txt) for inserido na caixa de texto *"Funções de Transição"* e em seguida clicarmos no botão *"Carregar Máquina"* a aplicação irá carregar as funções de transição na subseção *Construtor* e montar o escopo de animação da máquina na subseção *Simulador.*
2) **Funções de Transição:** uma caixa de texto para inserir as funções de transição uma vez programadas;
3) **Nova Máquina:** um botão para limpar todas as entradas do *Construtor* e iniciar a criação de uma nova máquina;
4) **Carregar Máquina:** um botão para carregar as funções de transição da máquina de Turing uma vez programadas;
5) **Salvar Máquina:** um botão para salvar as funções de transição da máquina de Turing que foram programadas nas subseções *Construtor* e *Simulador.*

b) **Construtor:** Na subseção *Construtor*, a aplicação permite programar as funções de transição de uma máquina de Turing de forma completamente visual. Esta ferramenta de construção de funções de transição e o grande diferencial das aplicação web citadas no capítulo 3 de Trabalhos Relacionados. A princípio, a seção *Tutorial* já terá ensinado como programar uma máquina de Turing através da construção de estados e transições da máquina, onde juntas compõe o que TURING (1936) chamou de "m-configurações", ou como abordado no trabalho, "funções de transição". Um programa de máquina de Turing "detector de sentenças equivalentes" estará disponível no botão *Exemplo* anteriormente citado. Assim, a subseção *Construtor* é composta, como mostra a *Figura 14* abaixo, pelos elementos:

Figura 14 - Construtor de Funções de Transição

Fonte: Elaborado pelo autor.

1) **Adicionar Estado:** um botão para adicionar os estados da máquina. Ao adicionar um estado é criado um "*container*" com opções de criar, deletar e editar transições dentro do estado;

2) **Remover Estado:** um botão para remover o estado e o correspondente *"container"* de transições;
3) **Adicionar Transição:** um botão para adicionar uma transição no *"container"* de transições do estado. Ao adicionar uma transição é inserida uma "linha" com caixas de texto à serem preenchidas para configurar a transição. Esta linha é uma abstração da função de transição do modelo formal da máquina de Turing, que no caso serão compostas, de acordo com a *Figura 15,* por:

Figura 15 - Container Transições do Estado

Fonte: Elaborado pelo autor.

i. **Atual:** uma caixa de texto para inserir o valor atual lido na fita pela cabeça;
ii. **Novo:** uma caixa de texto para inserir o novo valor que irá sobrescrever o valor atual lido da fita;
iii. **q?:** uma caixa de texto para inserir o próximo estado após a transição;
iv. **{< , >}:** uma caixa de texto para ser inserido a direção em que a cabeça 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);

v. **Remover Transição:** um botão para remover a transição correspondente ao botão.

A relação que cada item, ou componente do *“container”* de transições de cada estado com a função de transição, representada anteriormente no modelo formal no sub-capítulo 2.3, é aplicada de forma fiel quanto ao conceito. Para o qual:

δ: (Qatual , ľlido) > (Qnovo , ľnovo , {E,D})

Onde ***δ*** representa cada linha de transição do *“container” do estado atual* ***Qatual****, visto que* ***ľlido*** é equivalente ao símbolo “**Atual**” lido na fita, ***ľnovo*** é equivalente ao símbolo “***Novo***” à ser escrito na fita***, Qnovo*** é equivalente ao novo estado ***q?*** e ***{E,D}*** indicam o sentido de movimento da cabeça equivalente à **{<, >}**.

c) ***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 visualizar a máquina em funcionamento. O *Simulador* tem alguns controles de máquina e elementos de tela muito importantes, que podem ser observados na *Figura 16 abaixo*, eles são:

Figura 16 - Simulador de Máquina de Turing

Fonte: Elaborado pelo autor.

1) **Estado Inicial:** uma caixa de texto para inserir o estado inicial da máquina;
2) **Fita da Máquina:** uma caixa de texto para inserir a sequência de valores (qualquer carácter antes definido nas transições da máquina);
3) **Iniciar:** um botão com a funcionalidade de dar início aos procedimentos e funcionamento da máquina;
4) **Parar:** um botão com a funcionalidade de parar todo o processo que estiver sendo executado pela máquina;
5) **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 da cabeça na fita;

6) **Reiniciar:** um botão com a funcionalidade de reiniciar a máquina para o seu estado inicial anterior ao momento de sua inicialização.
7) **Máquina de Turing (Animação):** um espaço reservado na tela onde ocorrerão as animações correspondentes aos processos de transição e movimentação da cabeça na fita. Podemos dividir a animação, de acordo com a *Figura 17,* em:

Figura 17 - Animação da Máquina de Turing

Fonte: Elaborado pelo autor.

i. **Estado Atual:** espaço na tela onde é informado o estado atual em que a máquina está processando as transições;
ii. **Posição da Cabeça:** espaço na tela onde é informado a posição em que a cabeça está na fita;
iii. **Resultado:** espaço na tela onde é informado o status da máquina durante e no final da análise da fita;
iv. **Cabeça:** espaço onde é simulado a animação de um cabeçote (olho) de leitura percorrendo a fita;

v. **Fita da Máquina:** espaço onde é mostrado os valores da fita e as alterações feitas pelas transições da máquina.

Agora que sabemos o que cada elemento da tela significa, é importante mostrar um exemplo simples de máquina de Turing usando a aplicação. Vamos partir do pressuposto de uma máquina de Turing que apenas aceite número 1 como entrada e se for encontrado algum número 0 na sequência da entrada a máquina entra em estado de rejeição. A *Figura 18* abaixo mostra como foi implementado a lógica das funções de transição da máquina.

Figura 18 - Máquina de Turing que Aceita Apenas Número 1 de Entrada

Fonte: Elaborada pelo autor.

Nota-se que no estado q0 foi definido que o símbolo “_” seria o símbolo de início da fita, mas se por acaso tal símbolo não fosse digitado na sequência de entrada, ao invés fosse digitado 0 ou 1 a máquina entenderia que tal símbolo está mais a esquerda da fita. Para o estado q1 foi definido que enquanto encontrarmos o número 1 escrevemos “#” por cima e continuaríamos num loop dentro do estado q1, mas se for

encontrado “-”, simbolizando agora o símbolo de fim da fita, iríamos para o estado qA de aceitação. Porém, caso encontrado algum número 0 durante o processo cairíamos no estado qR de rejeição. Vejamos o resultado para uma sequência de entrada igual a “_1111111_” na *Figura 19* abaixo.

Figura 19 - Resultado da máquina para uma sequência "_1111111_"

Fonte: Elaborado pelo autor.

Agora vejamos o resultado da máquina para uma sequência de entrada igual a “_1110_” na *Figura 20* abaixo.

Figura 20 - Resultado da máquina para um sequência de entrada "_1110_"

Fonte: Elaborado pelo autor.

5.2 Código-fonte

Para um melhor entendimento técnico da aplicação esta seção tem como objetivo disponibilizar o código-fonte utilizado no desenvolvimento do projeto através do link compartilhado abaixo do controlador de versão GitHub. https://github.com/wellmmeR/tm-web-simulator-unasp

6 ANÁLISE E RESULTADOS

6.1 Dificuldades e Desafios

No decorrer do desenvolvimento houveram algumas dificuldades e desafios quanto a implementação da aplicação. O maior desafio enfrentado acredito ter sido a implementação dos conceitos e

funcionamento da máquina de Turing em uma linguagem simples como o Javascript. Cada abstração de conceito aplicado em alguma parte da aplicação foi pensada com cautela para que não houvesse erros de implementação. Um dos desafios foi criar e desenvolver a subseção *Construtor*, onde houvesse uma ferramenta gráfica para construção de estados e transições da máquina de Turing. A solução encontrada foi tratar cada transição como um objeto de cinco atributos (estado atual, símbolo lido, símbolo novo, estado novo, direção da cabeça). Assim, seria mais fácil na hora de montar os *"containers".*

A maior dificuldade foi na hora de achar algum meio de fazer a animação da máquina fluir de forma coerente, pois a cabeça, simbolizada por um ícone de um olho, teria que andar por cima de cada caractere individualmente, porém como fazer isso acontecer sendo que cada caractere tem um espaço em *pixels* na tela diferente um do outro, por exemplo, a letra "i" é menor em espaçamento de *pixels* do que a letra "w". A solução foi tratar os caracteres da fita com uma propriedade de fonte do CSS chamada *"monospace"* no qual cada caractere teria um espaço de *pixels* em comum um com o outro. Logo, isso proporcionaria que o olho, também sendo um caractere importado da coleção de ícones do Bootstrap também tivesse a mesma proporção de *pixels* de cada caractere.

6.2 Resultados Obtidos

Para avaliar se o nível de aprendizado sobre máquina de Turing que a aplicação poderia proporcionar para um usuário foi aplicado um questionário de avaliação de conhecimentos sobre Alan Turing e máquina de Turing antes e após o uso da aplicação, por categoria de área de formação ou estudo, através da Google Forms, uma ferramenta Google para criação de questionários.

As orientações e perguntas elaboradas para o questionário foram divididas em três partes:

1) **Parte 1:** Orientações iniciais sobre como responder o questionário e um campo para digitar o email do avaliador, para evitar avaliações repetidas;
2) **Parte 2:** Perguntas à serem respondidas antes de usar a aplicação, das quais foram:
    a) Qual área do conhecimento corresponde a sua formação ou estudo?
        1. Ciências Exatas e da Terra
        2. Ciências Biológicas
        3. Engenharias
        4. Ciências da Saúde
        5. Ciências Agrárias
        6. Ciências Sociais Aplicadas
        7. Ciências Humanas
        8. Linguística, Letras e Artes
        9. Outros
    b) Qual a opção que corresponde ao seu conhecimento atual sobre Alan Turing e máquina de Turing antes de usar a aplicação?
        1. Não sei nada.
        2. Já ouvi falar.
        3. Possuo conhecimento básico.
        4. Possuo conhecimento aprofundado.

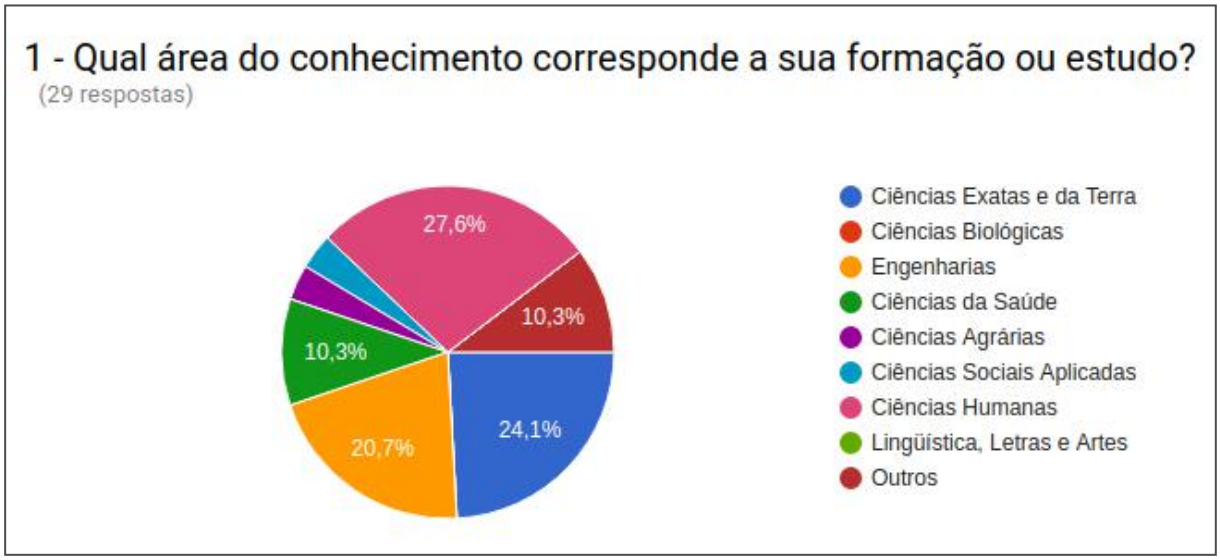
3) **Parte 3:** Pergunta à ser respondidas após usar a aplicação, das quais foram:

a) Qual a opção que corresponde ao seu aprendizado sobre Alan Turing e máquina de Turing após de usar a aplicação?

1. Não aprendi nada.
2. Aprendi pouco.
3. Agora tenho noção do que se trata.
4. Aprendi muito.

O questionário foi aplicado de forma livre com pessoas convidadas de diversas áreas de estudo ou formação. Ao total 29 pessoas usaram a aplicação e responderam o questionário, tendo como resultado um gráfico de respostas por área de estudo ou formação, onde 27,6% são de Ciências Humanas, 24.1% são de Ciências Exatas e da Terra, 20,7% são de Engenharias, 10,3% são de Ciências da Saúde e o percentual restante dividido entre Ciências Agrárias e Sociais Aplicadas, como mostra a *Figura 21* a seguir.

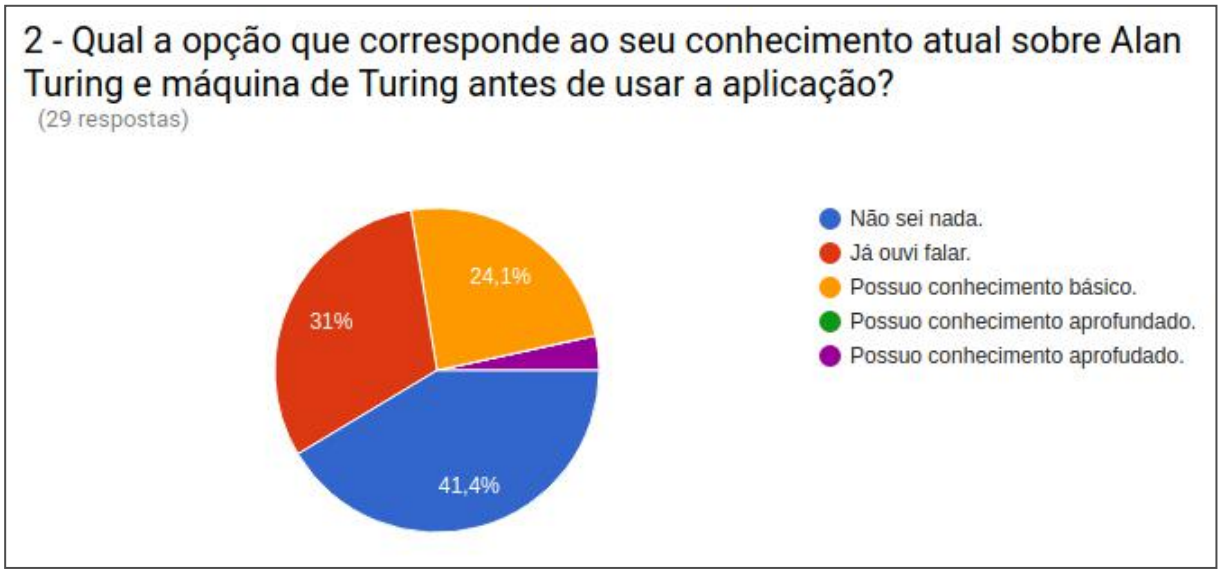
Figura 21 - Gráfico de Respostas por Área de Estudo ou Formação



Fonte: Elaborada pelo autor.

Outro gráfico de respostas gerado foi com relação a segunda pergunta da *Parte 2* do questionário, onde 41,4% não sabiam nada à respeito de Alan Turing ou máquina de Turing, 31% já tinham ouvido falar, 24,1% possuíam conhecimento básico e o percentual restante possuíam conhecimento aprofundado sobre o assunto, como podemos ver na *Figura 22* a seguir.

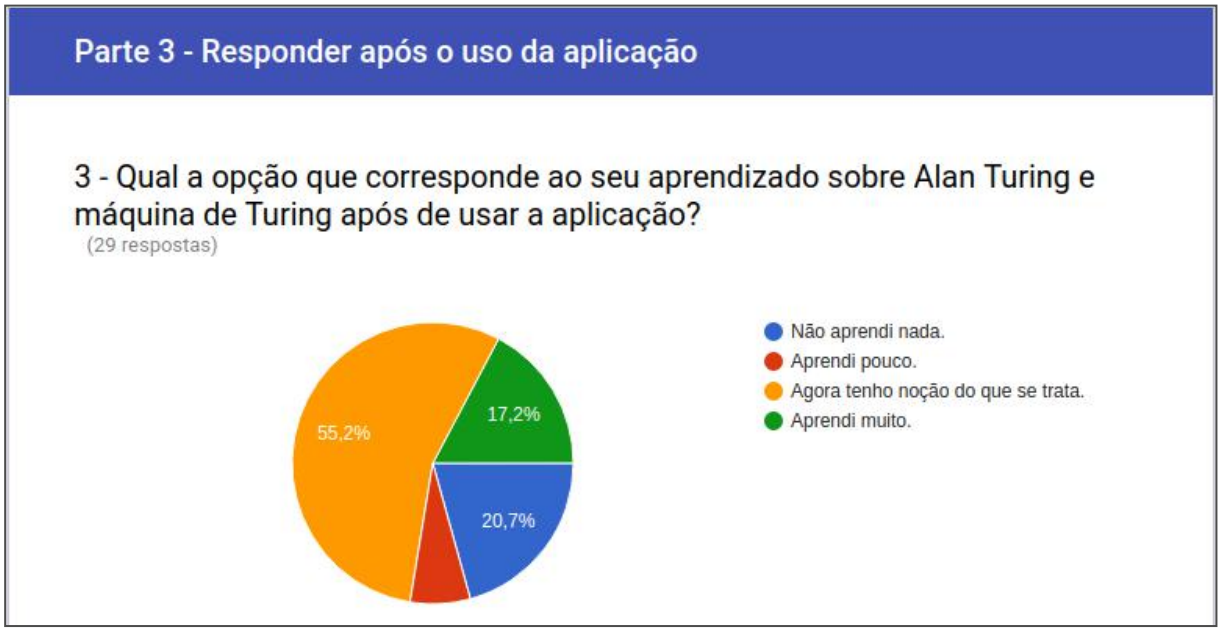
Figura 22 - Gráfico de Respostas da 2ª Pergunta da Parte 2 do Questionário



Fonte: Elaborado pelo autor.

Por último, obteve-se um gráfico de respostas gerado a partir da terceira pergunta da *Parte 3* do questionário, onde 55,2% agora tem noção do que se trata o assunto sobre Alan Turing e a máquina de Turing, 20,7% declararam não ter aprendido nada, 17,2% declararam ter aprendido muito e o percentual restante declararam ter aprendido pouco. como podemos ver na *Figura 23* a seguir.

Figura 23 - Gráfico de Respostas da 3ª Pergunta da Parte 3 do Questionário



Fonte: Elaborado pelo autor.

Visto, um total de 79,3% dos usuários avaliadores, somando 55,2% dos que agora tem noção do que se trata, 17,2% dos que aprenderam muito e 6,9% dos que aprenderam pouco, tem-se por resultado superficial desta avaliação de que para boa parte dos usuários a aplicação teve algum efeito na aprendizagem sobre a pessoa de Alan Turing e os conceitos e funcionamento da máquina de Turing.

7 CONCLUSÃO

Concluímos este trabalho com bons resultados vindos tanto da avaliação do nível de aprendizagem obtido com o uso da aplicação quanto com cumprimento dos objetivos propostos por este trabalho citados no capítulo 1.2 e seus sub-capítulos 1.2.1 e 1.2.2.

No projeto foi desenvolvido uma estrutura web com facilidade quanto a usabilidade e navegação da aplicação. Também foi desenvolvido um espaço de aprendizado teórico dos conceitos e funcionamento da máquina de Turing, no caso a seção *Tutorial* da aplicação; um outro

espaço para construção de estados e funções de transição, através de uma ferramenta gráfica que facilitasse a programação da máquina de Turing; e um último espaço para simulação da máquina programada no espaço de construção de estados e funções de transição.

Com todos esses objetivos propostos concluídos é relevante colocar, também pelo resultado das avaliações do questionário, que a aplicação pode assumir um papel bastante importante no processo de aprendizagem dos conceitos e funcionamento da máquina de Turing, mais especificamente se direcionando à alunos de Ciência da Computação.

7.1 Trabalhos Futuros

Com base na conclusão do trabalho, é importante ressaltar que novas funcionalidades poderão ser incrementadas e/ou melhoradas, seno algumas delas:

- Implementação de um *web-service* com banco de dados e acesso com login para que o usuário possa guardar as máquinas construídas na aplicação;
- Implementação de um botão de “passo para trás” que volte a máquina para um estado e transição anterior à atual;
- Melhorar a animação dos processos da máquina de Turing;
- Implementação de um espaço para visualização de um autômato finito correspondente a máquina de Turing construída;
- Implementação da função “não se mover” como parâmetro de sentido de movimento da cabeça;
- Migração do projeto de jQuery para Angular.

8 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.

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

BOOLOS, George S., BURGESS, John P., JEFFREY, Richard C. Computability and Logic. 5th Edition, 2007. Princeton University, New Jersey. ISBN: 9780521877527

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

DIVERIO, Tiaraju Asmuz; MENEZES, Paulo Blauth. Teoria da Computação: Máquinas Universais e Computabilidade, 2011. 3ª Edição, Volume 5. Editora Bookman.

DUCKETT, Jon. Javascript and JQuery: Interactive Front-End Web Development, 2014. WILEY - ISBN: 978-1-118-53164-8

FLANAGAN, David; FERGUSON, Paula (2002). Javascript: The Definitive Guide, 4ª Edição. O'Reilly & Associates [S.l.] - ISBN 0-596-00048-0.

GARCIA, Dayane O.; LIMA, Rosana L. Um Simulador Para A Máquina de Turing, 2003. Trabalho de Graduação de Bacharel em ciência da Computação, Universidade Tuiuti do Paraná, Faculdade de Ciências

Exatas e de Tecnologia. Disponível em: http://tcconline.utp.br/wp-content/uploads/2009/10/FACET_CC_2003_UM_SIMULADOR_PARA_A_MAQUINA_DE_TURING.pdf - Acesso em 12 de Outubro de 2016.

HILBERT, David; ACKERMANN, Wilhelm. Grundzüge der Theoretischen Logik (Princípios de Lógica Matemática). Springer-Verlag, ISBN: 0-8218-2024-9.

HODGES, Andrew. Alan Turing: The Enigma - The Book That Inspired the Film: The Imitation Game, 2014. Princeton University Press. ISBN: 9781400865123.

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

jQuery Official Site. jQuery: write less, do more. Copyright 2016 The jQuery Foundation. jQuery License. Disponível em: https://jquery.com/ - Acesso em 20 de Novembro de 2016.

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

LIE, Hakon Wium; BOS, Bert. Cascading Style Sheets: Designing for the Web, 3ª edição. Addison-Wesley, 2005 - ISBN 0321193121

MATTE, José Antônio; UGARTE, MARTIN. Turing Machine. Disponível em: https://turingmachinesimulator.com/ - Acesso em 24 de Novembro de 2016.

MENEZES, Paulo F. Blauth. Linguagens Formais e Autômatos, 3ª edição, 2000. EDITORA Sagra Luzzato. UFRGS - Instituto de Informática.

MORPHETT, Anthony. Turing Machine Simulator, 2016. Disponível em: http://morphett.info/turing/turing.html - Acesso em 3 de Outubro de 2016.

PYPL. PopularitY Programming Languages. Pierre Carbonnelle, 2016. Disponível em: http://pypl.github.io/PYPL.html - Acesso em 27 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.

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

SIPSER, Michael. Introdução à Teoria da Computação. 2ª edição. Editora Thomson, 2007.

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.

Twitter Official Blog. The Twitter Developer Blog. Disponível em: https://blog.twitter.com/2011/bootstrap-from-twitter - Acesso em 24 de Novembro de 2016.

W3C Official Site. Copyright 2016, W3C - (MIT, ERCIM, Keio, Beihang). Disponível em: https://www.w3.org/ - Acesso em 30 de Novembro de 2016.

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