UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LEANDRO JUSTIN VIEIRA

MÁQUINAS UNIVERSAIS: DEMONSTRAÇÃO, ÁNALISE E EQUIVALÊNCIA

CRICIÚMA
2018

LEANDRO JUSTIN VIEIRA

MÁQUINAS UNIVERSAIS: DEMONSTRAÇÃO, ÁNALISE E EQUIVALÊNCIA

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Esp. Sergio Coral

CRICIÚMA

2018

LEANDRO JUSTIN VIEIRA

MÁQUINAS UNIVERSAIS: DEMONSTRAÇÃO, ÁNALISE E EQUIVALÊNCIA

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Teoria da Computação.

Criciúma, 29 de junho de 2018.

BANCA EXAMINADORA

Prof. Esp. Sérgio Coral - UNESC - Orientador

Prof. Esp. Fabricio Giordani - UNESC

Prof. Esp. Gilberto Vieira da Silva - UNESC

“Dedico esse trabalho a todos que me apoiaram, em especial meus professores, amigos e familiares”.

AGRADECIMENTOS

Agradeço a todos as pessoas que fizeram parte durante a caminhada da minha graduação sendo esses amigos, familiares e professores, pessoas que me apoiaram e auxiliaram para que obtivesse êxito na conclusão do curso de Ciência da Computação.

Aos meus colegas Tiago Aleff da Silva, Diego Pedro Marques, Raul Porto, Marcos Paulo e Andrews Duarte parceiros de estudos que conheci através do curso e auxiliaram durante toda a trajetória da graduação e minha estadia em Criciúma, esses serão lembrados durante o resto da minha vida.

Agradeço ao meu orientador Sergio Coral, por acreditar em meu potencial e compartilhar seus conhecimentos esses que foram fundamentais para a conclusão desse trabalho.

Ao professor Fabricio Giordani por participar do meu desenvolvimento pessoal como programador, conhecimento esse que foi utilizado na construção do trabalho.

Por fim agradeço a Deus por me auxiliar durante essa caminhada.

“Há um teorema conhecido que diz que qualquer computador é capaz de emular qualquer outro computador”

Arthur C. Clarke

.

RESUMO

A ciência da computação é fundamentada em modelos computacionais construídos na primeira metade do século XX, em 1936 foi publicado a Máquina de Turing modelo conhecido e aceito como a formalização de um algoritmo, esse que pertence à uma classe chamada de máquinas universais, detentora do maior poder computacional até hoje. A teoria da computação tem um papel importante na construção do conhecimento e proporciona desenvolvimento do raciocínio logico e formal, sendo que esse é cada vez mais necessário para a computação. É apresentado quatro modelos de máquinas universais (Máquina de Turing, Máquina de Post, Autômato de duas pilhas e Máquina de Norma), esses que são contextualizados e a partir disso desenvolvido simuladores para cada, além de demostrar também como um modelo pode ser simulado por outra máquina universal, afim de comprovar a Tese de Church. Ao final da pesquisa, foi desenvolvido um protótipo web responsável pelos simuladores e que efetua a conversão de Máquina de Post para Máquina de Turing e Autômato de Duas Pilhas para Máquina de Turing.

Palavras-chave: Tese de Chruch. Teoria da Computação. Máquinas Universais. Máquina de Turing. Simulação e Conversão.

ABSTRACT

Computer science is based in computational models built in the first half of the 20th century. In 1963, the Turing Machine was published, a model known and accepted as the formalization of an algorithm, that which belongs to a class known as the universal machines, possessing the greatest computational power to this day. The computation theory has an important role in the construction of knowledge and provides development of logic and formal thinking, that being more and more necessary for computation. Four models of universal machines are presented (Turing's Machine, Post's Machine, Two-Cell Automaton and Norma's Machine), those which are contextualized and from that, simulators were developed, other than also demonstrating how a model can be simulated by another universal machine, in order to prove Church's Thesis. At the end of the research, a web prototype was developed, responsible for the simulators and that effectuates a conversion from Post's Machine to Turing's Machine and from the Two-Cell Automaton also to Turing's Machine.

Key Words: Church's Thesis. Computation Theory. Universal Machines. Turing's Machine. Simulation and Conversion.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MN	Máquina de Norma
MP	Máquina de Post
MT	Máquina de Turing

SUMÁRIO

1 INTRODUÇÃO

A ciência da computação é fundamentada em modelos computacionais que foram construídos na primeira metade do século XX, entretanto sua origem é milenar, sendo desenvolvida em diversas regiões e ao longo da história, estando presentes nas culturas mesopotâmia, egípcia, grega, babilônica, indiana, chinesa e asteca. A ciência da computação atual possui duas ênfases, a teórica baseada em fundamentos e modelos computacionais, e a ênfase prática que aplica a teoria na construção de projetos de sistemas computacionais. É importante ressaltar que a ênfase teórica não depende de instrumentos computacionais ou máquinas tecnologias, ou seja, não necessita de computadores para ser formulada (hardwares e softwares). (DIVERIO; MENEZES, 2011).

A ênfase teórica baseou-se em diversas áreas para fundamentar os modelos computacionais atuais, esses que hoje são a base da Teoria da Computação, como por exemplo, na biologia (modelos para redes de neurônios), na eletrônica (teoria do chaveamento), na matemática (lógica), na linguística (gramáticas para linguagens naturais) e em outros. (DIVERIO; MENEZES, 2000).

Como dito os modelos utilizados atualmente foram construídos na primeira metade do século XX, aparentemente nesse momento histórico específico sentiu-se uma grande necessidade de caracterizar o que significaria ser computável, definir um modelo de computação suficientemente genérico para especificar qualquer função computável. É possível notar que existiu essa necessidade pois diversos pesquisadores trabalharam simultaneamente nesse problema, entretanto, mesmo os modelos sendo construídos independentemente, eles se mostraram matematicamente equivalentes, ou seja, parecia já existir de fato um conceito característicos de computabilidade, que podia ser formulado de diversas maneiras, contudo cada modelo nasceu de um contexto de pesquisa distinto, sendo portanto único, tendo papéis diferentes na evolução da ciência da computação, e cada um evidência com maior clareza aspectos específicos da noção da computabilidade. (SILVA; MELO, 2006).

Entre os modelos desenvolvidos podemos citar o Cálculo Lambda (Alonzo Church, 1936), funções Recursivas (Stephen Cole Kleene, 1936), Máquina de Turing (Alan Turing, 1936), Máquina Norma (Richard Bird, 1976), Sistema Canônico de Post (Emil Leon Post, 1936), entre outros.

Dentre os modelos citados o trabalho de Turing é universalmente conhecido e aceito como a formalização de um algoritmo, que são procedimentos sistemáticos, utilizados para resolver uma determinada questão a partir de um número finitos de passos. (SILVA; MELO, 2006). A máquina de Turing é considerado um formalismo simples, universalmente conhecido e provavelmente o mais usado como modelo teórico de computação. Além disso a sua fundamentação teórica serviu para o desenvolvimento dos computadores, ou seja, a partir dos estudos de Turing foi possível desenvolver o computador que temos hoje. (DIVERIO; MENEZES, 2011). Além disso, mesmo o modelo sendo proposto em 1936 até hoje não se conseguiu nenhum outro tipo de máquina que tenha um poder computacional maior que a máquina de Turing. (VIEIRA, 2006).

Entretanto existe uma dificuldade no que diz respeito ao compreendimento desses conteúdos, pois os mesmos possuem uma complexidade maior devido que para seu entendimento é necessário compreender modelos matemáticos, diferente de outras disciplinas e áreas da computação. (VIEIRA; VIEIRA; VIEIRA, 2003).

Deste modo, o trabalho busca apresentar os modelos descritos, demonstrando a equivalência entre os mesmos, desenvolvendo um simulador para a máquina de Turing, Norma, Post e autômato de duas pilhas com possíveis conversões entre os modelos simulados, demonstrando assim a equivalência entre os modelos e de certo modo as diferenças em seus processos.

1.1 OBJETIVO GERAL

Desenvolver uma aplicação que simule e demostre as máquinas universais e suas equivalências.

1.2 OBJETIVOS ESPECÍFICOS

a) compreender as maquinas universais: Maquina de Turing, Norma, Post e autômato de duas pilhas.
b) entender conceitos e teorias como a tese de church, computabilidade, problema de parada, problemas de decisão.

c) encontrar a equivalência entre os maquinas universais e demonstrar os mesmos.
d) desenvolver uma aplicação que demostre os modelos e apresente conversões entre eles.

1.3 JUSTIFICATIVA

A teoria da computação caracteriza-se pelo estudo formal dos processos de computação, suas capacidades e suas limitações (CINTRA; BRONFMAN; DIVERIO, 1996), sendo de grande importância para a ciência da computação, pois fundamenta e conceitua o embasamento teórico para um correto e amplo entendimento da ciência envolvida na computação, além de proporcionar um desenvolvimento do raciocínio logico e formal, sendo que esse é cada vez mais necessário para a computação. (DIVERIO; MENEZES, 2000). Além de que diferentes modelos de computação acabam se mostrando mais adequados para explicar diferentes paradigmas de programação, portanto, é importante que se tenha o conhecimento de diversos modelos. (SILVA; MELO, 2006).

Ou seja, todos esses fundamentos estão ligados com o que os cientistas da computação desenvolvem hoje, por exemplo, os conceitos da máquina de Turing podem auxiliar na percepção de como se dá o fluxo do software, conhecer e entender teorias como a de problemas intratáveis também é de suma importância, pois permite ao acadêmico que se deparar com um problema, conseguir identificar se é possível desenvolver uma solução para o mesmo, pois esse pode pertencer a uma classe de problemas intratáveis, sendo então necessário descobrir um modo diferente do habitual para contornar o problema. (HOPCROFT; ULLMAN; MOTWANI, 2002).

.

1.4 ESTRUTURA DO TRABALHO

O trabalho é dividido em sete capítulos, onde o primeiro capítulo apresenta uma introdução ao trabalho os objetivos gerais e específicos e sua justificativa.

O capítulo segundo aborda o início da teoria da computação e é dado enfoque em três áreas centrais da teoria da computação e demonstrado a ligação entre elas.

No terceiro capítulo é apresentado conceitos de programa, máquinas e computação, sendo que ao final é apresentado conceitos de equivalência entre programa e máquina.

O quarto capítulo inicia o estudo das máquina universais, é descrito sobre essa classe de máquinas e logo após é conhecido quatro modelos de máquinas universais, sendo esses, Máquina de Turing, Máquina de Post, Autômato de Duas Pilhas e Máquina de Norma.

O quinto capítulo apresenta a equivalência de cada máquina universal descrita no capítulo quarto com a máquina de Turing.

No sexto capítulo encontram-se trabalhos correlatos a esta pesquisa.

Por último, o sétimo capítulo traz o trabalho proposto, a metodologia para o desenvolvimento deste trabalho, os recursos que serão necessários e o cronograma correspondente.

2 TEORIA DA COMPUTAÇÃO

Historicamente o marco inicial da teoria da computação se tem com o trabalho desenvolvido por David Hilbert, publicado em 1900, que apresentava 23 problemas, sendo o décimo denominado de Entscheidungsproblem (problema da decibilidade). A partir desse estudo em 1936, Alan Turing propõem uma máquina que futuramente seria aceita como a formalização do algoritmo. (DIVERIO; MENEZES, 2011).

A teoria da computação torna-se essencial para a ciência, pois apresenta conceitos que fundamentam a base de uma boa implementação, conceitos obtidos a partir de sistemas equacionados pela lógica, engenharia e matemática, além de que os estudos apresentados servem como base para a compreensão de uma variedade de fenômenos computacionais, que dão suporte para o desenvolvimento de inúmeras aplicações tais como problemas NP, especificação e verificação de programas, análise sintática em compiladores, criptografia, análise de algoritmos (complexidade computacional) e projeto de linguagens de programação de alto nível. (CINTRA; BRONFMAN; DIVERIO, 1996).

Nas próximas sessões será dado enfoque em três áreas centrais da teoria da computação, sendo essas, autômatos, computabilidade e complexidade.

2.1 COMPLEXIDADE

Os estudos de Turing possibilitaram a automatização da resolução de problemas, entretanto um problema computacional possui diferentes variáveis, tornando-o as vezes mesmo que computacionalmente decidível (que possui uma solução), inviável, devido a quantidade de recursos que o mesmo necessita para encontrar uma solução, ou seja, a complexidade, analisa os recursos requeridos para resolver um determinado problema, tendo como medidas principais o tempo e memória gasta para a solução do problema. (SIPSER, 2007).

O estudo da complexidade é válido porque no momento que se pretende utilizar máquinas para tratar problemas reais, é natural ter uma preocupação com os processos computacionais envolvidos, não basta somente obter um algoritmo capaz de encontrar a solução, mais sim obter algoritmos “eficientes”, que alçassem a solução em um tempo e com quantidade de memória finitos. (LEAL, 2002).

O estudo da complexidade apresenta formas de classificação dos problemas de acordo com a sua dificuldade computacional, a partir desses esquemas pode-se encontrar subsídios para apontar um problema como computacionalmente difícil ou não. Além de apresentar uma gama de opções para a resolução de problemas computacionais de solução difíceis, como a diminuição do escopo do problema, analisar se a complexidade não está ligada diretamente com o pior caso (combinações das entradas de variáveis, apresentada de forma que o algoritmo necessite maior quantidade de recursos para chegar a solução), em determinadas situações o melhor e o caso médio satisfazem a finalidade da aplicação (algoritmo), verificar se uma resolução parcial não atende ao seu propósito ou entender qual o motivo da complexidade e alterar por algo que seja mais facilmente solúvel. (SIPSER, 2013).

2.2 COMPUTABILIDADE

A Teoria da complexidade descrita acima está diretamente ligada a computabilidade, enquanto a complexidade trata de classificar os problemas como fáceis ou difíceis o intuito da computabilidade é investigar a existência de uma solução para o problema, ou seja, tenta traçar os limites da computação, ou seja, determina o que pode ser implementado/desenvolvido em um computador. (DIVERIO; MENEZES, 2000).

Os resultados levantados pelo estudo da computabilidade, são apresentados em modelos teóricos que auxiliariam na construção dos computadores reais. (SIPSER, 2007).

2.3 AUTÔMATOS

A teoria dos autômatos apresenta modelos que são aplicados em diversas áreas computacionais, como é o caso do autômato finito (utilizado em compiladores), projetos de hardware, gramática livre-de-contexto (utilizada em linguagens de programação e inteligência artificial). O estudo de autômatos é um excelente início para quem busca compreender a teoria da computação, pois tanto a computabilidade como a complexidade necessitam de definições precisas para a

representação de um computador, essas que podem ser providas das definições formais de computação apresentadas pelo estudo de autômatos. (SIPSER, 2007).

3 PROGRAMAS, MÁQUINAS, COMPUTAÇÃO

Diferentes modelos de computadores podem ter diferentes arquiteturas (Harvard, Von Neumann) além de que existem linguagem de programação em abundância, logo, a formalização dos conceitos de programa e de máquina não podem ser baseados em qualquer linguagem ou em um computador real, é necessário algo genérico para sua representação, sendo utilizado então modelos matemáticos simples que descrevem suas características essenciais, permitindo um rápido entendimento de suas semânticas e facilitando a demonstração de seus resultados. (DIVERIO; MENEZES, 2000).

Um programa, pode ser visto como um conjunto de operações e testes, compostas de acordo com uma estrutura de controle, é dever do programa apresentar como devem ser as operações ou testes. As estruturas de controle podem ser definidas como (DIVERIO; MENEZES, 2000):

a) monolítico: baseada em desvios condicionais e incondicionais;
b) iterativo: possui estruturas de iteração de trechos de programas;
c) recursivo: baseado em sub-rotinas recursivas (que chamam a si mesmas);

As operações e testes definidas em um programa podem ser definidas executadas de forma (DIVERIO; MENEZES, 2000):

a) sequencial: A execução da operação ou teste, deve seguir uma ordem cronológica, ou seja, uma operação somente pode ser realizada após o encerramento da execução da operação ou teste anterior;
b) determinística ou escolha: A próxima operação ou teste a ser executado é determinado de acordo com uma escolha entre diversas alternativas de operações ou testes compostos;
c) concorrente: As operações podem sem executadas em qualquer ordem inclusive simultaneamente;

Um programa pode ser definido como uma sequência de símbolos ou um conjunto de indicadores, sendo incapaz de descrever e efetuar uma computação por si só, ou seja, encontrar uma solução sozinho, é necessário uma máquina para efetuar as ações descritas no programa, essa deve possuir uma estrutura que porte todas as operações do programa, que possa armazenar ou recuperar informações descritas em uma estrutura de memória, informações obtidas a partir dos

comandos/funções descritas no programa. É função da máquina possuir mecanismos e formas de interpretar cada comando descrito no programa, a fim de gerar de forma coerente uma resposta, para dar continuidade ao programa. Uma máquina é capaz de interpretar um determinado programa desde que possua uma interpretação para cada operação ou teste que constitui o programa. (DIVERIO; MENEZES, 2000).

O conceito de programa satisfaz à noção intuitiva de algoritmo, sendo que um algoritmo é descrito por uma sequências de operações detalhadas e não-ambíguas de modo que essas irão produzir um resultado, além de que deve ser constituído de um número finito de passos, executáveis em tempo finito. (PY, 2003).

Como foi citado anteriormente uma computação precisa de uma máquina capaz de interpretar cada comando descrito no programa, uma computação é um histórico da execução do programa pelos mecanismos fornecidos na máquina a partir de um valor inicial, sendo que é dito como função computada, uma computação que chega ao seu final, obtendo assim uma resposta para o valor inicial baseado nos comandos descritos no programa executados pelos mecanismos existentes na máquina. (DIVERIO; MENEZES, 2000).

Seja M uma máquina e P um programa descrito para M onde L é seu correspondente conjunto de instruções. Uma computação do programa P na máquina M, para uma entrada X (X $\in$ conjunto de valores aceitos por M), tem como resultado uma cadeia (finita ou infinita) de configurações/descrições definidos por (L, V), onde L é o rótulo/estado/instrução atual da máquina e V o valor da memória anterior a execução do L.

A partir das funções computadas é possível apresentar uma relação de equivalência em programas e máquinas (DIVERIO; MENEZES, 2000):

a) relação de equivalência forte de programas: Dois programas possuem essa relação caso as suas funções computadas coincidem em qualquer máquina.
b) relação de equivalência de programas em uma máquina: Dois programas possuem essa relação caso suas funções computadas coincidem para uma máquina específica.
c) relação de equivalência de máquina: Duas máquinas possuem esse grau de equivalência se podem se simular mutuamente.

4 MÁQUINAS UNIVERSAIS

Máquinas universais, classe de máquinas tão poderosa que até hoje não se conseguiu nenhum outro tipo de máquina que tenha maior poder computacional, será apresentado agora quatro modelos de máquinas universais (Máquina de Turing, Máquina de Post, Máquina de Norma, Autômatos de Duas Pilhas), após isso será demonstrado a equivalência entre essas máquina.

4.1 MÁQUINA DE TURING

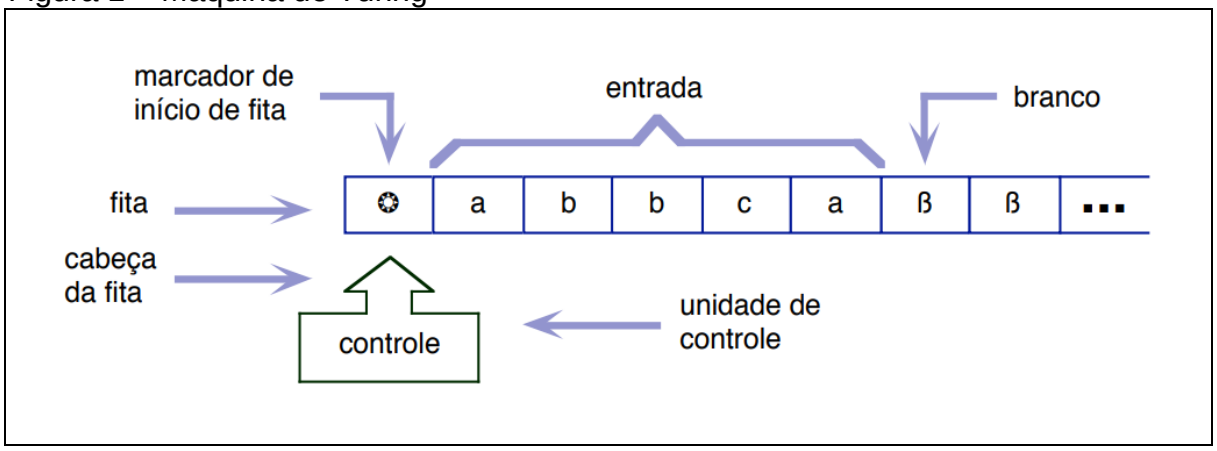
Em 1996 Alan Turing propôs uma máquina semelhante a um computador, máquina que possuía componentes como memória de trabalho (fita), unidade de controle, sistema de leitura e escrita, ou seja, componentes semelhantes aos computadores atuais, através dessa máquina é possível tornar o funcionamento de um algoritmo perceptível, sendo a mesma considerada como a formalização de um algoritmo (processamento efetivo ou função computável), ou seja, demonstra em um tempo finito, uma sequência de instruções que podem ser executadas mecanicamente, obtendo uma função computável (a partir de uma entrada se obtém uma saída). Segundo a Tese de Church, a máquina de Turing é considerada uma máquina universal, sendo essa uma classe de dispositivos computacionais que marcam o máximo que uma máquina pode computar, além de afirmar que para qualquer função computável é possível obter/construir uma máquina de Turing para processar essa função. (HOPCROFT; MOTWANI; ULLMAN, 2001; LEWIS; PAPADIMITRIOU, 2004; VIEIRA, 2006).

Uma máquina de Turing pode ser definida como uma máquina que opera em uma fita, na qual é possível escrever e ler valores (símbolos), sendo dividida em células que comportam um único elemento/símbolo, sendo essa infinita para a direita e finita para a esquerda com seu marco final para a esquerda o símbolo inicial da fita, é utilizada como dispositivo de entrada e saída para a memória de trabalho da máquina de Turing. Os símbolo possíveis na fita podem estar definidos no alfabeto da máquina de Turing, ou serem “branco” ou “marcador de início de fita”. (DIVERIO; MENEZES, 2011; PY, 2003; VIEIRA, 2006).

Além da fita a máquina de Turing possui uma unidade de controle, composta por um cabeçote responsável por efetuar a leitura e escrita na fita

podendo movimentar-se para a esquerda e para a direita, um registrador contendo o estado atual da máquina proveniente do programa, esse que será executado na máquina sendo responsável por definir o símbolo que será gravado na fita, indicar o próximo estado e a direção do movimento do cabeçote. Para facilitar a compreensão é possível comparar a máquina de Turing abreviada frequentemente por MT, como sendo uma pessoa que realiza cálculos em uma folha quadriculada, e a medida que realiza uma instrução/cálculo, apaga e escreve o valor resultante do cálculo em um quadrado da folha, além do valor o cálculo informa qual o próximo quadrado que terá seu valor substituído, esse processo ocorre até a pessoa encontrar um resultado final para seu cálculo ou perceber que para a combinação de valores que foi encontrado ele não possui um próximo cálculo\instrução. (DIVERIO; MENEZES, 2011; PY, 2003; VIEIRA, 2006).

Figura 1 – Máquina de Turing



Fonte: Menezes (2005).

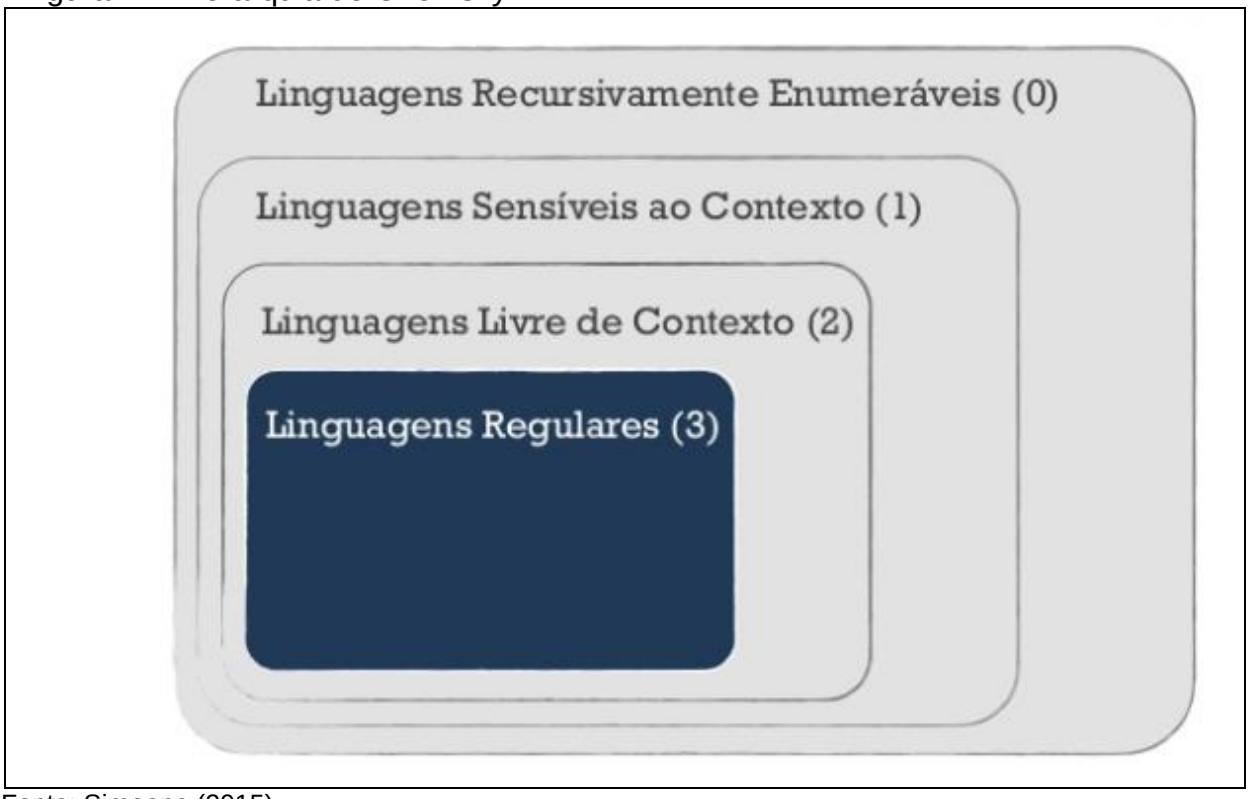
Ou seja a máquina de Turing pode ser vista informalmente composta por três componentes:

a) fita: dispositivo de entrada e saída e responsável pela memória de trabalho;
b) unidade de controle: Reflete o estado corrente da máquina, efetua a leitura e escrita da fita;
c) programa (função transição)**:** Conhecido também como função programa ou Função de Transição, define os estados da máquina, as gravações e o sentido do movimento do cabeçote (Unidade de controle), para determinar a continuidade da execução da máquina

(próximo estado, símbolo e movimento), o programa leva em consideração o símbolo atual (lido) e o estado corrente da máquina.

O programa interpretado por uma MT é classificado como uma linguagem do Tipo 0 conhecidas como Linguagens Recursivamente Enumeráveis explicando assim um dos pontos levantados na Hipótese de Church, de que uma MT é o dispositivo mais genérico e de maior poder computacional, visto que a classe das linguagens de Tipo 0 engloba todas as linguagens reconhecidas por máquinas e definidas computacionalmente, como é apresentado pela Hierarquia de Chomsky (MENEZES, 2000).

Figura 2 – Hierarquia de Chomsky



Fonte: Simeone (2015).

4.1.1 Definição Formal

Segundo Diverio e Menezes (2011), uma máquina de Turing é 8-upla:

$$M = (\sum, Q, \sqcap, q0, F, V, \beta, \varnothing)$$

onde:

$\sum$ alfabeto de símbolos de entrada;

Q conjunto de estados possíveis da máquina, o qual é finito;

⊓ programa ou função de transição;

⊓: Q x (∑ U V U { β, Ø }) ➡ Q x (∑ U V U { β, Ø }) x { E, D};

q0 estado inicial da máquina tal que q0 é elemento de Q;

F conjunto de estados finais tal que F está contido em Q;

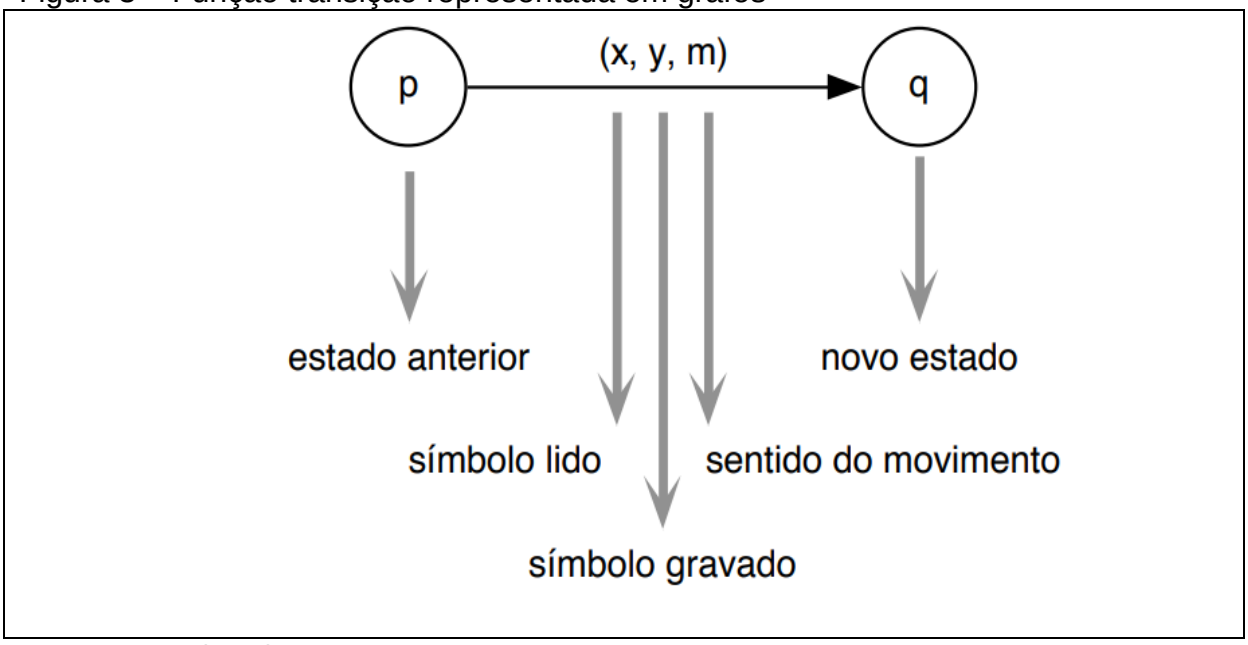
V alfabeto auxiliar;

β símbolo especial branco;

Ø símbolo especial marcado de início ou símbolo de início da fita.

A função programa ou função de transição pode ser demonstrada através de um grafo finito direto, onde o vértices é estado da máquina e as arestas são compostas por o símbolo lido, símbolo a ser gravado e o sentido de movimento do cabeçote, como é definido na função transição ⊓ = (Q x (∑ U V U {β, Ø}) ➞ Q x (∑ U V U {β, Ø}) x {E, D}). (DIVERIO; MENEZES, 2011).

Figura 3 – Função transição representada em grafos



Fonte: Menezes (2005).

A computação de uma MT definida como M = (∑, Q, ⊓, q_0, F, V, β, Ø), para uma palavra de entrada W ∈ ∑, equivale a constantes aplicações das funções de transições (⊓) com o cabeçote responsável pela leitura, estando posicionado no símbolo inicial da fita. (MENEZES, 2005).

Pode-se assumir para a entrada W duas situações, *aceita* caso a palavra obtenha um resultado ao ser processada em M, ou seja, alcance um estado de

parada, *rejeita* caso a palavra não alcance um estado de parada após a sua execução, a rejeição pode acontecer de duas formas, caso o símbolo lido e estado corrente em que a máquina se encontrada, não possuam uma função transição definida para esse conjunto, ou a função transição indique que o cabeçote se movimente para a esquerda, sendo que este já se encontrada na última célula a esquerda, para essas duas situações a palavra é rejeitada. (MENEZES, 2005)

4.1.2 Exemplo de Programa

A seguir é apresentado uma MT que executa a soma de dois valores representados pelo caracteres '⁕', onde M = (∑, Q, ⊓, q_0, F, V, β, Ø), sendo:

∑ = {⁕};

Q = {q_0, q_1, q_2};

⊓ representado pela figura 4 e a tabela 1.

q0 estado inicial.

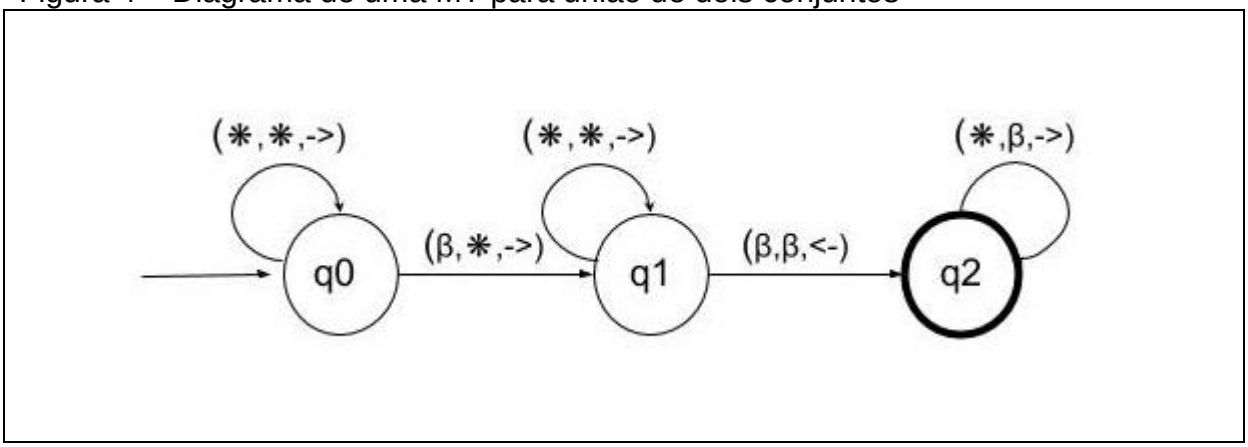
F = {q_2};

V = {};

β símbolo branco;

Ø estado inicial da máquina.

Figura 4 – Diagrama de uma MT para união de dois conjuntos



Fonte: Do Autor.

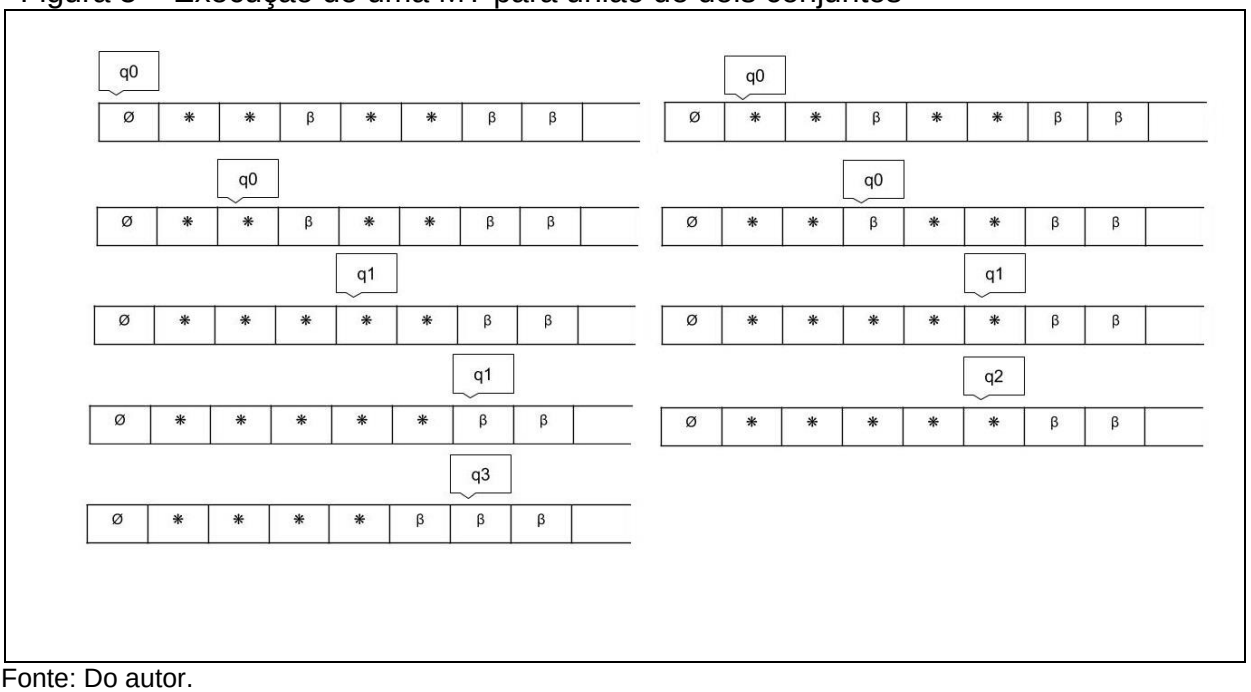
Tabela 1 – Tabela de funções transição de uma MT para união de dois conjuntos

	Ø	⁕	β
q_0	(q_0, Ø, D)	(q_0, ⁕, D)	(q_1, ⁕, D)
q_1		(q_1, ⁕, D)	(q_2, β, E)
q_2		(q_2, β, E)	ACEITA

Fonte: Do autor.

Para uma palavra de entrada W = ⁕⁕β⁕⁕ββ executada na MT descrita acima, sua execução terminaria no estado q2, com o resultado de ⁕⁕⁕⁕ escrito na fita, como é demonstrado figura 5.

Figura 5 – Execução de uma MT para união de dois conjuntos



Fonte: Do autor.

4.2 MÁQUINA DE POST

Emil Post em 1936, mesmo ano que Turing apresentou a sua máquina universal, publicou um artigo sobre o que poderia ser computável e o problema da solubilidade. No trabalho de Post é apresentado a máquina de Post, que será denominada como MP durante o trabalho, essa máquina apresenta um conceito de

algoritmo, sendo considerada um máquina universal como à máquina de Turing, entretanto a MP consiste em um modelo mais simples em relação ao controle de memória e o detalhamento das instruções, essa simplicidade acaba tendo um custo, normalmente uma MP necessita de uma maior quantidade de instruções e memória para chegar aos mesmos resultados de uma MT (TORRES, 2002).

4.2.1 Definição Formal

Segundo Diverio e Menezes (2011), a Máquina de Post é uma tripla:

M = (Σ, D, #), onde:

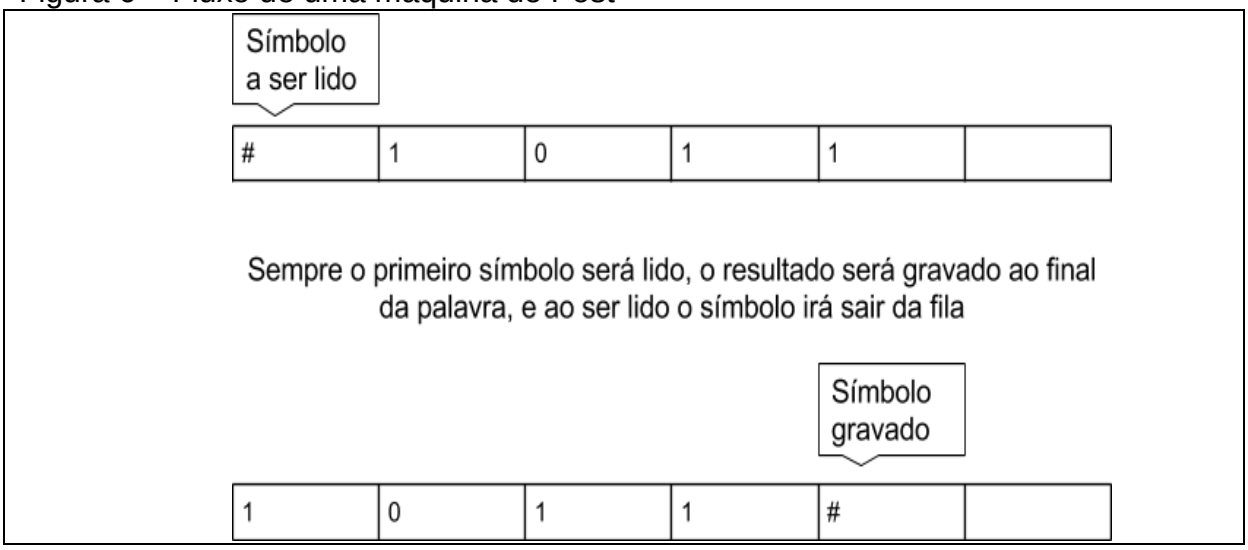
Σ = alfabeto de símbolos de entrada;

D = programa ou diagrama de fluxos construído a partir de componentes elementares denominados **partida, parada, desvio** e **atribuição;**

= símbolo auxiliar.

A estrutura de uma MP é composta por uma variável de entrada X pertencente ao alfabeto de símbolos aceitos pela MP (X ∈ Σ), uma fita de tamanho finito que é utilizada como entrada, saída e memória de trabalho, igualmente como em uma MT, contudo sua fita possui um comportamento em fila, o primeiro símbolo a entrar é o primeiro a sair, sendo essa a grande diferença entre as duas máquina, o comportamento em fila de sua fita causa um impacto na sua forma de execução, ou seja, sempre será lido o primeiro símbolo e a saída será gravada no final da fita. (EXATAS, 2015).

Figura 6 – Fluxo de uma máquina de Post

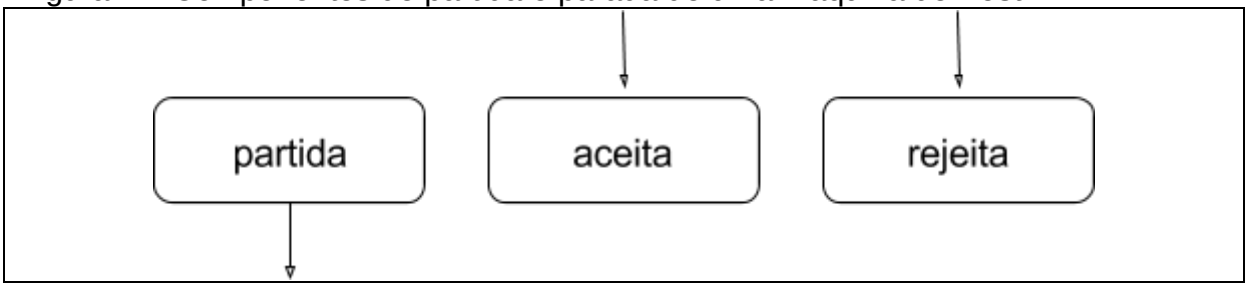


Fonte: Do autor.

A figura 6 mostra o funcionamento em fila de uma MP defina em (Σ, D, #) para uma entrada X = #1011, onde o primeiro símbolo da palavra é o símbolo auxiliar, esse é lido e retirado da fila, e devido a execução de uma instrução definida em D é gravado ao final da palavra de entrada X.

As instruções do programa ou diagrama de fluxos para uma MP definida como M = (Σ, D, #) são armazenados em D, sendo obtidos por componentes elementares, responsáveis por formar um fluxo que determina a execução da MP, os componentes são classificados em instruções de partida, parada, desvio e atribuição. O componente de partida, demarca o início da MP, ou seja, o início do programa. Já o componente de parada, representa o final da execução da MP, podendo ser do tipo aceitação ou rejeição, um diagrama de fluxo pode conter um único ponto de partida (instrução de partida), entretanto não existe um limite para instruções de parada (zero ou mais), uma computação em uma MP tem sua execução terminada no momento em que a variável de entrada X é aceita ou rejeitada pela máquina (alcança uma instrução de aceitação ou rejeição), lembrando que é possível uma MP entrar em um loop infinito, ou seja, sua execução nunca parada, sendo então infinita (DIVERIO; MENEZES, 2011).

Figura 7 – Componentes de partida e parada de uma máquina de Post

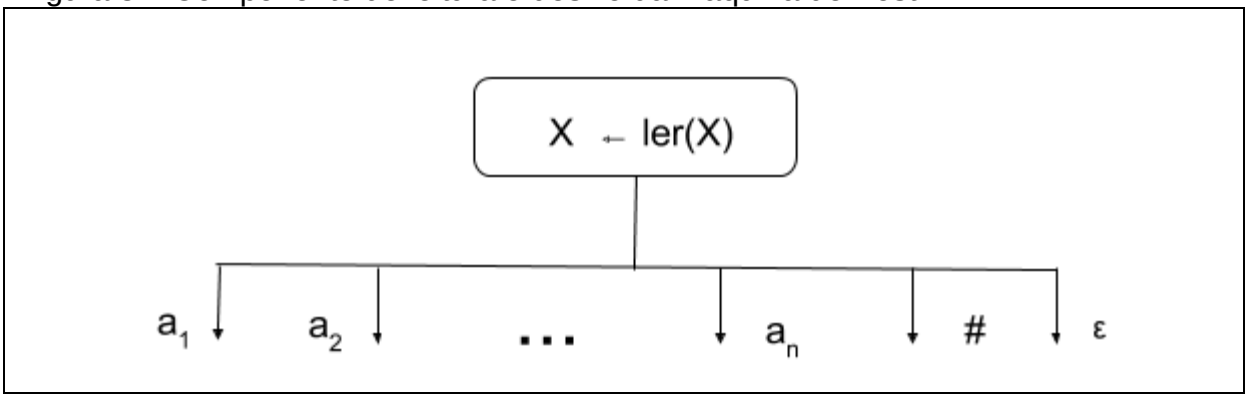


Fonte: Adaptada de Diverio e Menezes (2011).

A instrução de teste ou desvio é responsável por determinar o andamento do programa, qual caminho o mesmo irá seguir, caminho representado por uma aresta de ligação entre os componentes elementares definidos em D, sendo que para a tomada de decisão de qual caminho/instrução irá a MP é analisado o primeiro símbolo da fita obtido a partir da instrução de leitura X ← ler(X), visto que a fita tem um funcionando de pilha, a leitura é destrutiva, ou seja, o símbolo é lido e retirado da pilha, consequentemente sendo removido de X, entretanto o símbolo retirado determinada o caminho seguinte do programa, sendo este ∈ X ∈ Σ, logo, é necessário para cada elemento do alfabeto definido em Σ um caminho (aresta na

instrução de desvio), conclui-se então que se n = cardinalidade de Σ, para uma instrução de teste/desvio é necessário n + 2 caminhos/arestas, visto que além do alfabeto é possível que o símbolo lido seja o símbolo auxiliar (#) e o símbolo vazio/branco (β). (DIVERIO; MENEZES, 2011).

Figura 8 – Componente de leitura e desvio da máquina de Post



Fonte: Adaptada de Diverio e Menezes (2011).

4.2.2 Exemplo de Programa

Da mesma forma que foi apresentado para a MT é apresentado um máquina de MP, capaz de executar a união de dois valores representados pelo carácter '⁕', sendo M uma MP definida em:

M = (Σ, D, #),

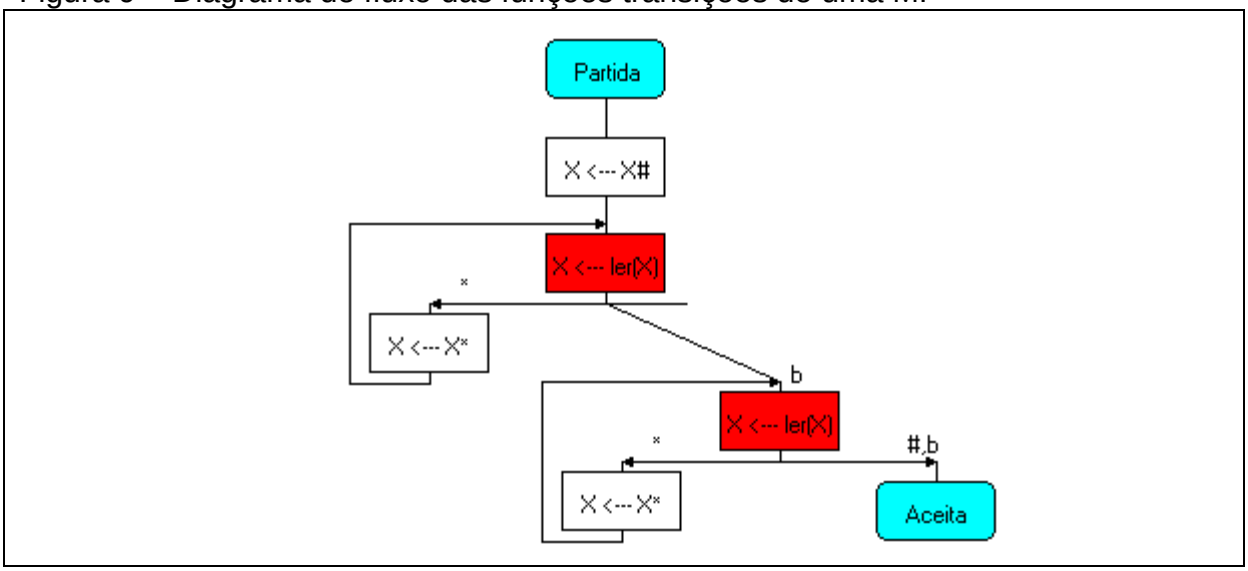
sendo:

∑ = {⁕, β};

D é descrita pela figura 9;

símbolo de início de fita.

Figura 9 – Diagrama de fluxo das funções transições de uma MP



Fonte: Do autor.

Para uma palavra de entrada X = ✳✳β✳✳ββ executada na MP descrita acima, sua execução terminaria no estado de aceitação, com o resultado sendo ✳✳✳✳ escrito na fita.

4.3 AUTÔMATO COM DUAS PILHAS

Uma máquina de duas pilhas possui o mesmo poder computacional que a máquina de Turing e a máquina de Post, sendo então considerada uma máquina universal. É possível classificar o poder computacional de uma máquina com pilhas de acordo com a quantidade de pilhas com que a mesma trabalha (DIVERIO; MENEZES, 2000).

a) máquina finita: corresponde a uma Máquina sem pilha, como será visto adiante a pilha serve como memória auxiliar e memória de trabalho, por não possuir pilha uma máquina finita tem seu poder computacional restrito, ou seja, dependendo do que é necessário computar, essa máquina não teria subsídios suficientes para a efetuar a tarefa, como é o caso de não existe máquina finita capaz de reconhecer um duplo balanceamento como em {an,bn | n > 0 }, entretanto é uma máquina fundamental para o estudo das Linguagens formais, analisadores léxicos;

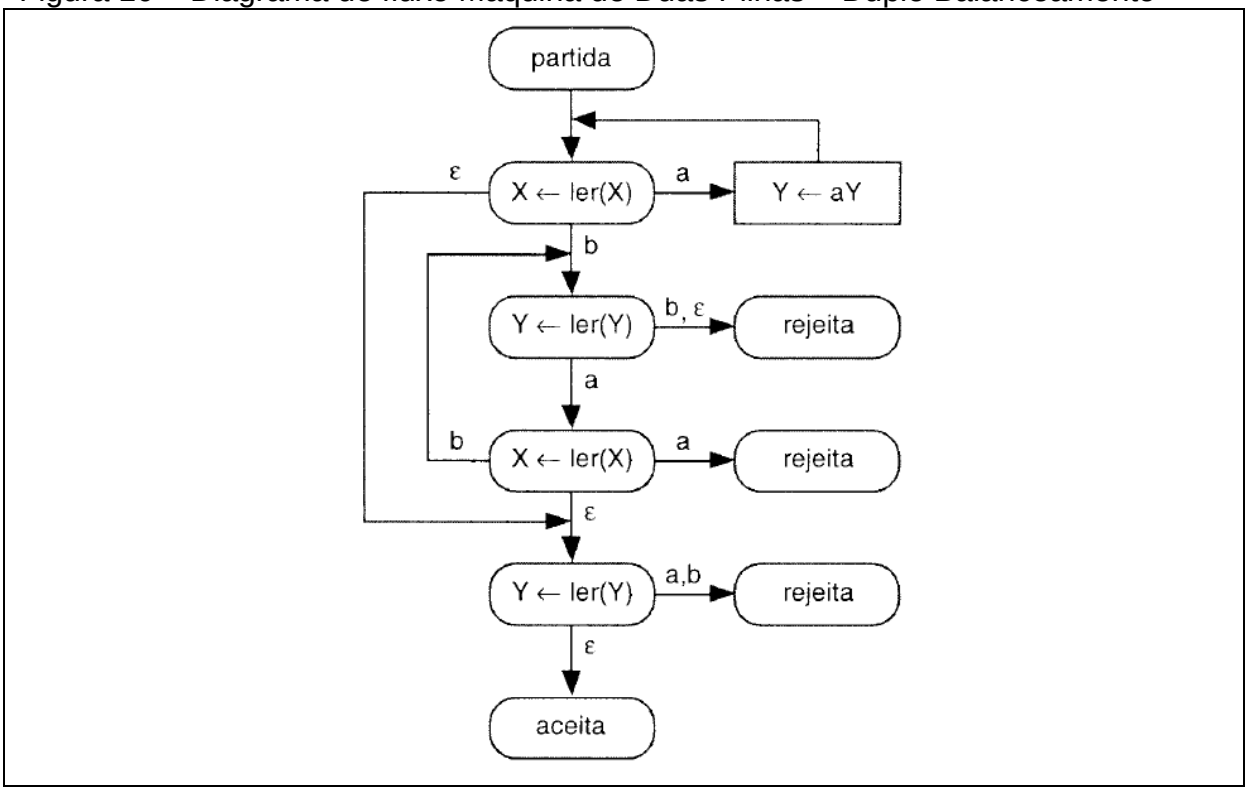
b) máquina com uma pilha: possui uma poder computacional maior que a de uma Máquina finita, entretanto sua capacidade computacional continua sendo restrita.

c) máquina com duas pilhas: Considerado uma máquina universal como o mesmo poder computacional da máquina de Turing e a máquina de Post.

Máquina com Mais de duas pilhas: Uma máquinas com mais de duas pilhas, possui um poder computacional igual a uma máquina com duas pilhas, como já foi citado anteriormente, uma máquina universal é considerada o modelo como maior poder computacional que temos até hoje, logo uma máquina de duas pilhas já sendo uma máquina universal, não teria ganhos em relação a capacidade computacional ao ser adicionado mais pilhas. (DIVERIO; MENEZES, 2000).

Uma máquina de duas pilhas é uma dupla definida por M = (Σ, D), onde Σ representa os símbolos de entrada da fita, e D armazena as instruções do programa representadas na forma de fluxograma, sendo que os componentes existentes do fluxograma são partida, parada, desvio, empilha e desempilha, semelhantes aos apresentados na máquina de post. (DIVERIO; MENEZES, 2000).

Figura 10 – Diagrama de fluxo Máquina de Duas Pilhas – Duplo Balanceamento



Fonte: Diverio e Menezes (2000).

O autômato com duas pilhas máquina que será apresentada é similar a Máquina de duas pilhas entretanto a principal diferença está na forma em que é apresentado o programa, onde uma máquina de duas pilhas apresenta o programa no formato de fluxograma e um autômato com duas pilha usa uma notação de estados que lembra a forma que é abordada na máquina de Turing, sendo essa forma mais indicada para estudos teórico-formais. (DIVERIO; MENEZES, 2000).

4.3.1 Definição Formal

Segundo Autômato (2003), um autômato com duas pilhas M ou simplesmente Autômato com pilha M é uma 6-upla:

M = (Σ, Q, Π, q0, F, V) onde:

Σ alfabeto de símbolos de entrada;

Q conjunto de estados possíveis do autômato o qual é finito;

Π função programa ou função de transição: Π: Q x (Σ U {ε,?}) x (V U {ε,?}) x (V U {ε,?}) ⟶ Q x (V U {ε,?}) x (V U {ε,?}) a qual é uma função parcial.

q0 estado inicial do autômato tal que q0 é elemento de Q;

F conjunto de estados finais tal que F está contido em Q;

V alfabeto auxiliar.

Ainda segundo Autômato... (2003), a máquina de um autômato de duas pilhas é composta por:

a) fita: armazena os símbolos de entrada;
b) duas pilhas: memórias auxiliares usadas para leitura e gravação de símbolos pertencentes ao alfabeto da máquina, as pilhas assim com a fita, são divididas em células onde cada uma é preenchida por um símbolo pertencente ao alfabeto de símbolos aceitos pela máquina, sendo que para a pilha a leitura e gravação sendo se dá em seu início (topo);
c) unidade de controle: Composta por um conjunto de componentes responsáveis por efetuar o controle e a execução do programa na máquina, levando em consideração o estado atual, símbolo lido da fita e símbolo lido da pilha. A Unidade de controle possui uma cabeça de fita, responsável por efetuar a leitura de um único símbolo/célula da fita sendo a leitura sempre da esquerda para a direita (não alterando sua

movimentação como em uma máquina de Turing), além da cabeça de fita, a unidade de controle possui para cada pilha, um mecanismo capaz de ler e gravar símbolos na pilha, esse mecanismo é chamado de cabeça de pilha, o mecanismo faz a leitura da pilha sendo essa leitura destrutiva, no momento que é efetuada o símbolo lido é retirado da pilha.

d) programa ou função de Transição: conjuntos de instruções que definem a execução da máquina. Para que uma transição seja aceita/executada é consultado o estado atual, fita e o topo de ambas as pilhas, determinando assim o próximo estado.

A computação de um Autômato de duas pilhas representado por M, para uma de entrada definida em W, consiste na aplicação das funções programa para cada símbolo de W da esquerda para a direita, até ocorrer uma condição de parada, podendo então assumir um estado final, resultando na aceitação de W para M, ou na indefinição, onde não existe em M uma função que resolva o estado atual de W, além de que é possível que M execute infinitamente não alcançando um estado final ou chegando em um indefinição. (AUTÔMATO 2003).

A função programa (Π) é constituída por dois conjuntos, (estado atual (p), símbolo da fita (x), símbolo da pilha 1 (a1) e pilha 2 (a2)) e (próximo estado (q), escreve na pilha 1 (b1), escreve na pilha 2 (b2)), ou seja, Π(p, x, a1, a2) = {(q, b1, b2)}, é importante citar que uma função programa pode ser indefinida para alguns dos argumentos do primeiro conjunto, a omissões do parâmetro é representada por “?”, indicando o teste da correspondente pilha vazia ou toda palavra de entrada lida, além de ser possível definir para a função algum parâmetro com o símbolo ε para indicar que nenhuma gravação será feita ou que não será feita mudanças de estados, ou que não será feito a leitura da pilha. (AUTÔMATO... 2003).

Segundo Diverio e Menezes (2000), no exemplo,
Π(p, ?, a, ε) = {(q, ε, a)} indica que:
Se:

- no estado p;
- a entrada foi completamente lida (na fita);
- o topo da pilha 1 contém o símbolo a;
- não lê da pilha 2;

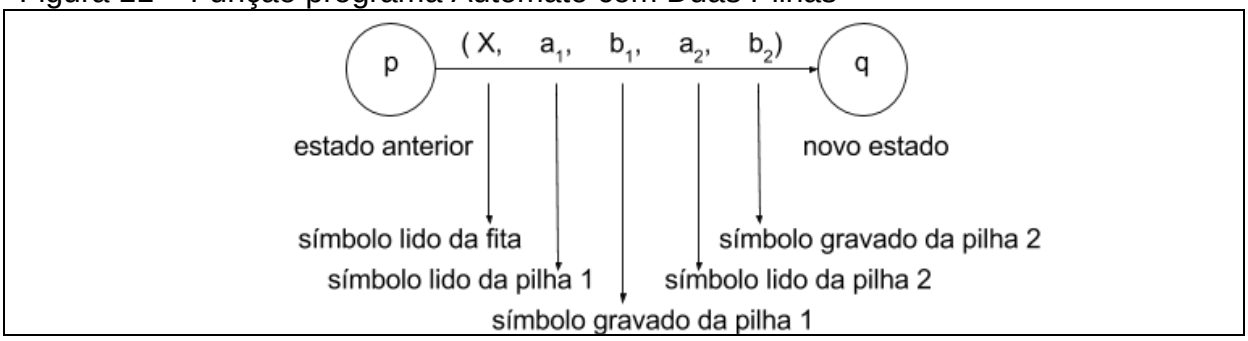
Então:

- assume o estado q;

- não grava na pilha 1;
- grava o símbolo b no topo da pilha 2.

É possível representar o conjunto de funções programa a partir de um grafo, como é demonstrado figura 11.

Figura 11 – Função programa Autômato com Duas Pilhas



Fonte: Diverio e Menezes (2000).

4.3.2 Exemplo de Programa

Da mesma forma que foi apresentado para a MT e MP é apresentado um autômato de duas pilhas, capaz de executar a união de dois valores representados pelo símbolo '⁕', sendo M um autômato de duas pilhas definido em M = (Σ, Q, Π, q0, F, V), sendo:

∑ = {⁕, β};

Q = {q0, q1, q2};

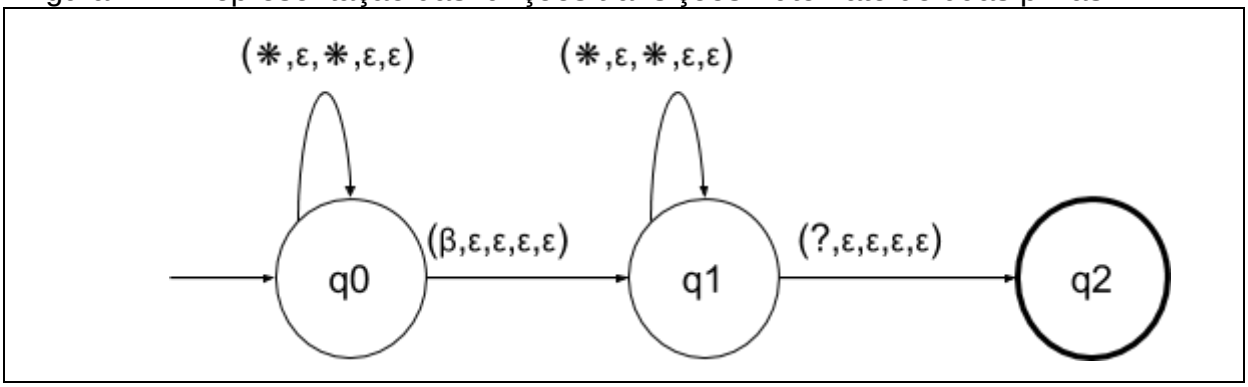
Π = {Π (q0, ⁕, ε) = {(q0, ⁕, ε)}, Π (q0, β, ε) = {(q1, ε, ε)}, Π (q1, ⁕, ε) = {(q1, ⁕, ε)}, Π (q1, ?, ε) = {(q2, ε, ε)}}

q0 = q0

F = {q2}

V

Figura 12 – Representação das funções transições Autômato de duas pilhas



Fonte: Diverio e Menezes (2000).

Nota-se que o valor resultante da união dos valores de '⁕', encontram-se na primeira pilha, e o exemplo utilizou somente uma das pilhas do autômato.

4.4 MÁQUINA DE NORMA

Richard Bird em 1976 propôs a máquina de norma, uma máquina de registradores (SILVA; MELO, 2006). A máquina de registradores comparado com os modelos que foram vistos até o momento (Máquina de Turing, Máquina de Post, Autômato de Duas Pilhas) e outros formalismos universais, é considerado um modelo mais recente, pois lembra a arquitetura básica dos computadores atuais. A estrutura de memória utilizada pela máquina de norma é composta por um conjunto de número infinito de registradores. Uma diferença presente na máquina proposta por Richard perante os modelos visto anteriormente é o fato do programa (função programa) e a máquina trabalharem de forma separada. (DIVERIO; MENEZES, 2000).

4.4.1 Definição Formal

Dada uma máquina de norma (registradores) $M = (\{R_k\}^{\infty}_{k=0}, \{\{e_k\}, \{s_k\}, \{z_k\}, \{add_k, sub_k\}\}^{\infty}_{k=0})$, onde (DIVERIO; MENEZES, 2000):

a) $\{R_k\}\infty k=0$: é o conjunto infinito de registradores, indexados por K, representados por letras maiúsculas (A, B, X, Y, …);
b) e_k: conjunto de funções de entrada sendo que, o resultado é aplicado na no registrador X e os demais terão seus valores zerados;

c) s_k: conjunto de funções de saída, recuperação do valor de Y;
d) z_k: conjunto de funções de teste, onde o resultado da execução em R resultará em um valor verdade, se $s_k(R_k) = 0$ será uma proposição verdadeira, caso contrário será falso;
e) add_k: Incrementa o valor de R_k;
f) sub_k: Decrementa o valor de R_k.

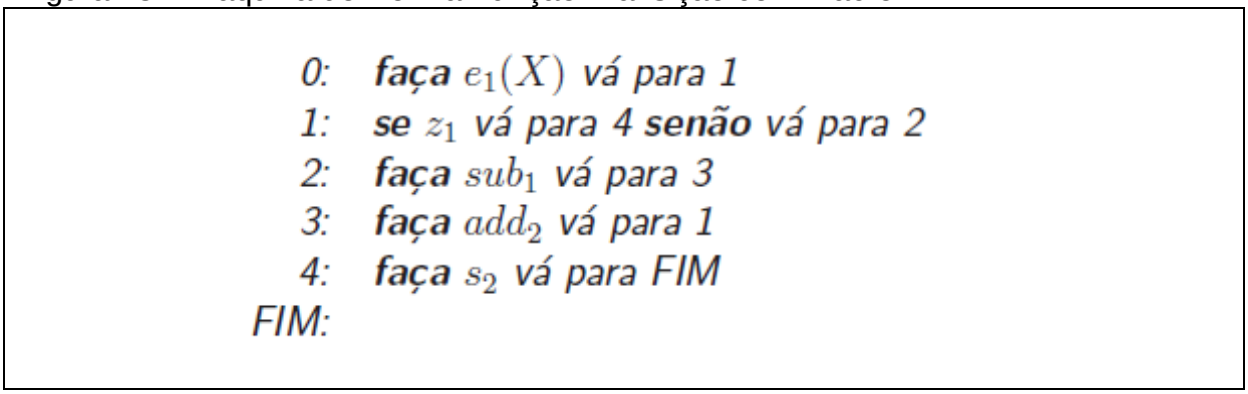
Um programa para uma máquina norma $M = (\{R_k\}^{\infty}_{k=0}, \{\{e_k\}, \{s_k\}, \{z_k\}, \{ad_k, sub_k\}\}^{\infty}_{k=0})$ pode ser definido por (X, Y, I, C, i0), onde (ALVES, 2007):

a) $X \in \{R_k\}^{\infty}_{k=0}$ sendo responsável por carregar o valor de entrada, todos os demais registradores são inicializados com o valor zero. Caso o valor de entrada for branco (ε), X não é modificado e o programa encerra sua execução;
b) $Y \in \{R_k\}^{\infty}_{k=0}$ responsável por armazenado a saída;
c) I conjunto de rótulos que identificam as funções executadas pela máquina.

Função programa: definida por $I \times (Q \times (I \cup (I \times I)) \cup \{\varepsilon\})$, composta por três elementos, tal que o primeiro elemento é rótulo da instrução atual, o segundo é uma função contida em $\{\{e_k\}, \{s_k\}, \{z_k\}, \{add_k, sub_k\}\}^{\infty}_{k=0}$, e o último elemento a próxima instrução. Caso a última instrução não for informada a instrução é considerada como uma instrução final. (ALVES, 2007).

Sobre a execução das funções essas podem definir adições, subtrações, multiplicações, divisões, armazenamento de valores, etc. Além de que os dados armazenados nos registradores podem estar definidos na forma de pilha, array's (unidimensionais e multidimensionais) R_{Kn}. Além de que é possível utilizar macros, na figura 13, *faça* e *se* são macros que não possuem significado semântico somente são utilizados para facilitar a legibilidade do programa, ou seja, caso retirados não influenciam em cada a execução da mesma. (ALVES, 2007; DIVERIO; MENEZES, 2000).

Figura 13 – Máquina de Norma Função Transição com Macro



Fonte: Alves (2007).

A figura 13, demonstra como pode ser definida uma máquina norma sendo (X, Y, I, C), onde:

a) $X \in \{R_k\}^{\infty}_{k=0}$;

b) $Y \in \{R_k\}^{\infty}_{k=0}$;

c) I = {0, 1, 2, 3, 4, FIM};

d) C = {(0, e1(X),1), (1, z1(4,2)), (2, sub1, 3), (3, add2, 1), (4, s2, FIM), (FIM, ε, ε)}.

Sendo e_1, z_1, sub_1, add_2, s_2 representações de função que seriam utilizadas, por exemplo e1 poderia ser substituído por X:= 5:

5 EQUIVALÊNCIA ENTRE AS MÁQUINAS UNIVERSAIS

No capítulo 4 foi visto modelos de máquinas universais, alguns destes modelos apresentam semelhanças, contudo cada modelo possui-a alguma singularidade perante a outra máquina, podendo ser sua forma de processar ou sua composição, entretanto todos os modelos possuem o mesmo poder computacional, ou seja, a classe dos algoritmos que podem ser descrita/computada por eles é a mesma, existindo então uma equivalência entre os modelos, sendo então possível uma máquina simular a outra, para isso, a máquina que agir como simulador deve reproduzir o comportamento, possuir o mesmo alfabeto com a inclusão de símbolos caso necessário e para cada transição da máquina a ser simulada possuir uma transição ou conjunto de transições capaz de efetuar a mesma ação sobre a palavra de entrada da máquina a ser reproduzida, sendo então capaz de simular perfeitamente o comportamento da outra dada uma entrada qualquer. (SIPSER, 2013).

5.1 AUTÔMATO DE DUAS PILHAS VS MÁQUINA DE TURING

Será provado a equivalência entre um Autômato de duas pilhas e uma Máquina de Turing, demonstrando que ambas possuem o mesmo poder computacional.

5.1.1 Máquina de Turing ⇾ Autômato de Duas Pilhas

Um autômato de duas pilhas definido em M = (Σ, Q, Π, q0, F, V) pode simular uma MT = (∑, Q, ⊓, q0, F, V, β, Ø) dada uma palavra de entrada W, a partir do fato que suas pilhas iriam simular a fita da MT, sendo que a primeira pilha simula os caracteres que estão à esquerda do cabeçote da MT e a segunda pilha os caracteres que estão a direita do cabeçote, ou seja, W estará contida inicialmente na segunda pilha, responsável pelos caracteres que estão à esquerda d o cabeçote, sendo o primeiro símbolo de W o primeiro da pilha. (AUTÔMATO, 2003).

Sobre a execução da máquina o movimento do cabeçote define em qual pilha seja gravado e desempilhar, se o cabeçote indicar para a esquerda será desempilhado da segunda pilha e empilhado na primeira, já se o movimento definido

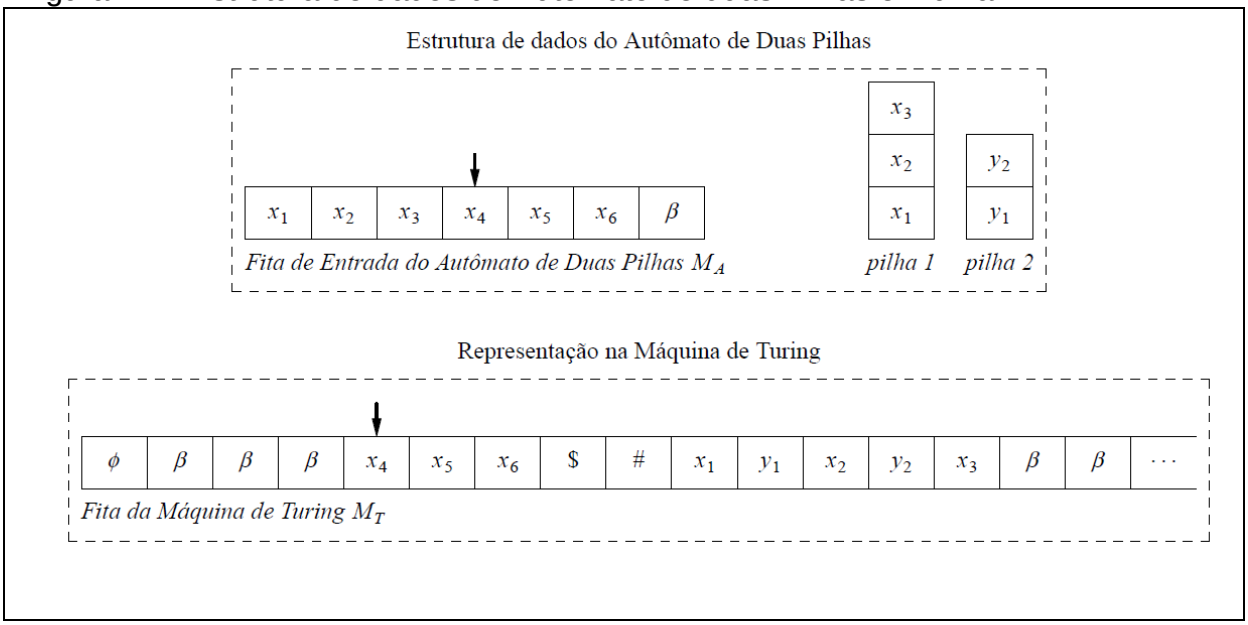
para o cabeçote for para a direita será desempilhado da primeira pilha e empilhado para na segunda. (AUTÔMATO..., 2003).

A função transação de uma MT definida em M = (∑, Q, ⊓, q0, F, V, β, Ø) é igual a (Q x (∑ U V U {β, Ø}) ➞ Q x (∑ U V U {β, Ø) x {E, D}), ou seja, ⊓ = (estado corrente, símbolo lido, novo estado, símbolo gravado, sentido do movimento), para um Autômato de duas pilhas simular a transação de M seguindo a definição de sua transação Q x (Σ U {ε, ?}) x (V U {ε,?}) x (V U {ε,?}) ⟶ Q x (V U {ε,?}) x (V U {ε,?}), ou seja, Π = (estado corrente, símbolo lido da fita, símbolo lido da primeira pilha, símbolo lido da segunda pilha, novo estado, símbolo gravado na primeira pilha, símbolo gravado na segunda pilha), é necessário que tanto o estado corrente como o estado novo serão os mesmos em ambas as funções, o símbolo lido da fita seja ε, indicando que não será lido a fita para o autômato de duas pilhas (visto que está estará em branco e o controle será feito somente pelas pilhas), e caso o movimento de MT seja para a esquerda o símbolo lido na fita de MT deverá ser lido na segunda pilha do autômato e o símbolo gravado na fita de MT será gravado na primeira pilha do autômato, e se o movimento for para a direita será o inverso o símbolo lido na MT deve ser lido na primeira pilha e o símbolo gravado na MT ser gravado na segunda pilha. (AUTÔMATO..., 2003).

5.1.2 Autômato de Duas Pilhas ⇾ Máquina de Turing

A simulação de um Autômato de duas pilhas em uma MT, é realizada de forma que a fita da MT represente tanto a palavra de entrada (armazenada na fita do autômato de duas pilhas), como as suas duas pilhas. A palavra de entrada ocupa as primeiras posições da fita, seguida de um símbolo (#) para demarcar o final da mesma, após isso é alocado na fita as pilhas, a primeira pilha correspondendo às células ímpares da fita após o símbolo de #, e a segunda pilha os símbolos pares após o símbolo #, como é demonstrado na imagem abaixo. (AUTÔMATO, 2003).

Figura 14 – Estrutura de dados do Autômato de duas Pilhas em uma MT

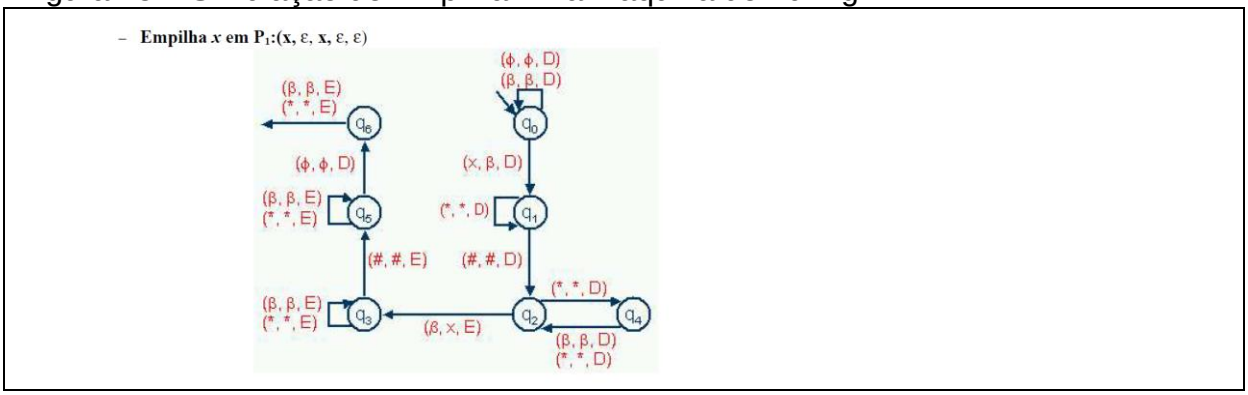


Fonte: Alves (2007).

Uma função transição em uma MT leva em consideração seu estado atual e símbolo lido efetuando apenas uma leitura e escrita, já em um autômato de duas pilhas a transição é baseada no estado, símbolo lido da palavra de entrada e suas pilhas, e podendo efetuar leitura na fita e suas pilhas e escrita em suas pilhas, sendo então necessário mais de uma função transição de uma MT para simular uma função de um autômato de duas pilhas. Além de que é necessário transições para fazer o controle das pilhas, e leitura da fita. (ALVES, 2007).

A figura 15 representa as funções transições necessárias para que uma MT simule uma função transição de um Autômato finito que ao ler da fita o elemento X grava X na primeira pilha, sem levar em consideração os elementos da pilha, ou seja $\Pi = (Q, x, \varepsilon, \varepsilon) = (\{Q, x, \varepsilon\})$, sendo que ao final é colocado o cabeçote no início da palavra de entrada, lembrando que como em uma autômato de duas pilhas a leitura é destrutiva no momento que é feito a leitura do símbolo na MT esse é substituído por vazio (β).

Figura 15 – Simulação de Empilha X na Máquina de Turing



Fonte: Autômato... (2003)

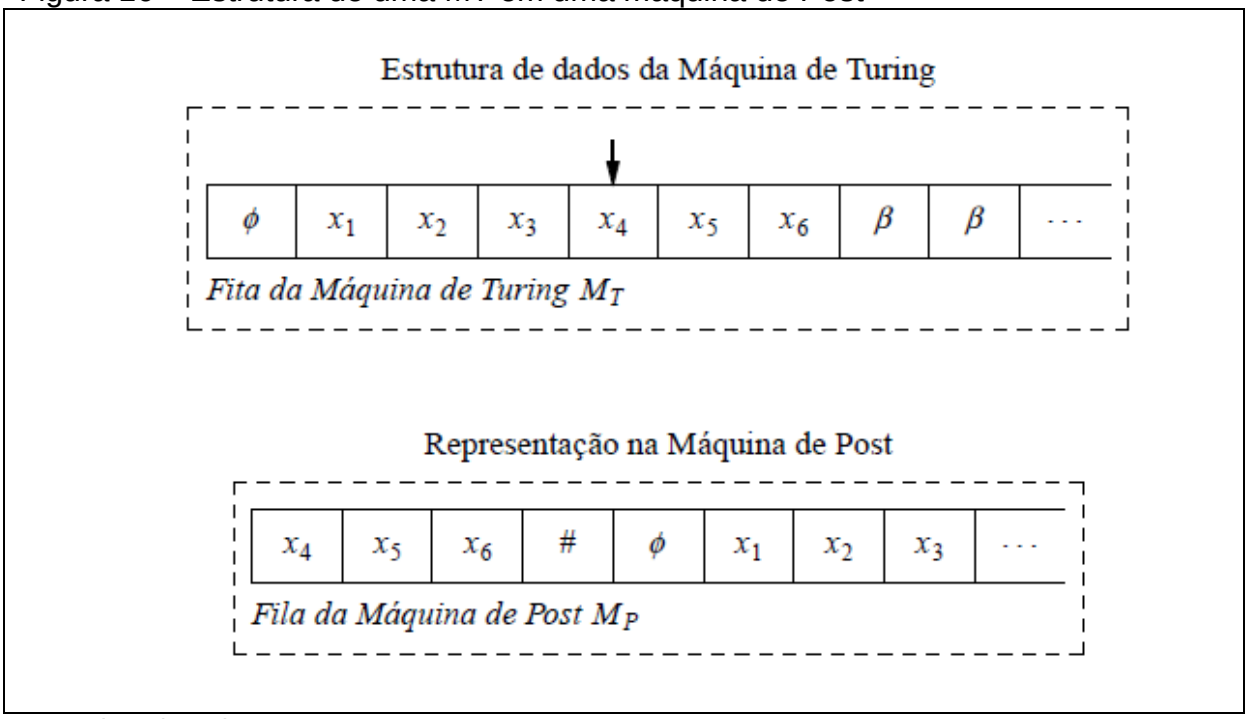
5.2 MÁQUINA DE POST VS MÁQUINA DE TURING

Será provado a equivalência entre uma Máquina de Post e uma Máquina de Turing, demonstrando que ambas possuem o mesmo poder computacional, ou seja, uma pode simular a outra.

5.2.1 Máquina de Post → Máquina de Turing

Para que uma máquina de Post possa reproduzir(simular) uma máquina de Turing, é necessário que sua estrutura em pilha substitua a fita presente na MT. O processo consiste na adição de um símbolo (#) que representará o cabeçote de leitura e escrita da MT, sendo que os símbolos que estão à direita do # são os símbolos que estão à esquerda do cabeçote da máquina de Turing (primeiro símbolo da fita até o penúltimo antes do cabeçote), e o que está à esquerda do # são os símbolos à direita do cabeçote de leitura, conforme é representado pela imagem abaixo. (ALVES, 2007).

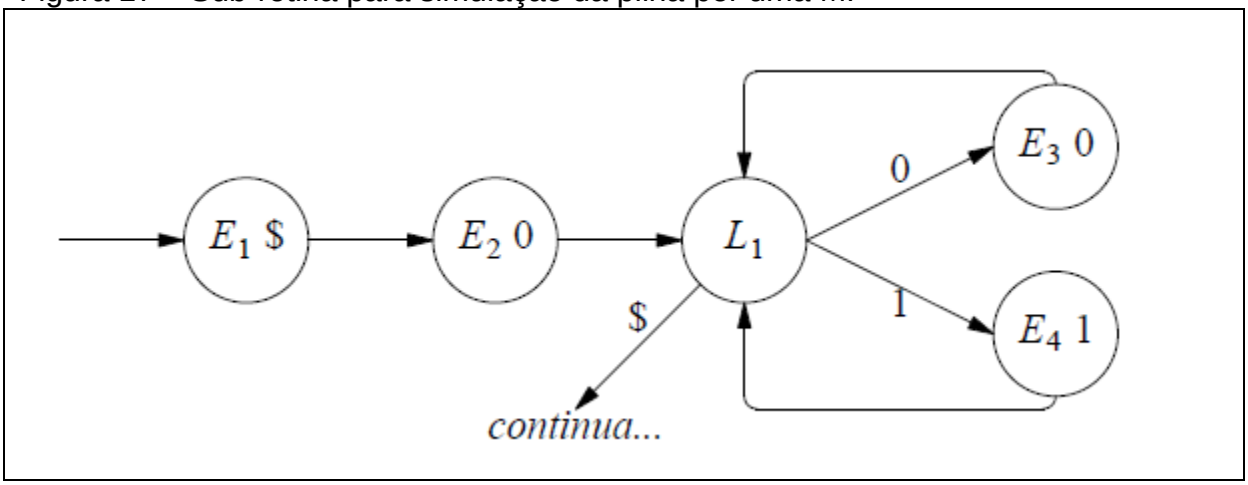
Figura 16 – Estrutura de uma MT em uma máquina de Post



Fonte: Alves (2007).

A partir da simulação da fita apresentada acima é possível definir como ocorrerá a leitura e gravação da máquina de post para simular a máquina de Turing, sendo essa baseada na movimentação do cabeçote de leitura, a movimentação para a direita é relativamente simples, o símbolo é desempilhado lido e gravado na última posição da pilha, seguindo o funcionamento normal da pilha, somente deve ser efetuado testes para caso de leitura do símbolo # pois no contexto para a direita ele representa o final da palavra de entrada, sendo então sua posição ocupada por um símbolo em branco (ε), caso não seja efetuado esse tratamento, a representação da fita ficaria inconsistente. A movimentação para a esquerda exige que seja efetuado uma sub-rotina na pilha, pois a mesma consegue trabalhar em um único sentido, essa sub-rotina é responsável por adicionar um elemento no começo da pilha, reposicionando os outros elementos, a figura 17 apresenta como pode ser desenvolvido uma sub-rotina para adicionar um elemento a frente da pilha utilizando um carácter auxiliar ($) para demarcar o momento que deve ser retornado para o fluxo normal do programa, no exemplo é 0 deve ser adicionado à frente da pilha, sendo o alfabeto da máquina ∑ = (1,0). (ALVES, 2007).

Figura 17 – Sub-rotina para simulação da pilha por uma MP



Fonte: Alves (2007).

5.2.2 Máquina de Turing → Máquina de Post

O formalismo de Post pode ser simulado por uma Máquina de Turing, colocando na fita da MT a variável de entrada (X) da máquina de post, onde a primeira posição de X consequentemente a primeira posição da pilha de post seria o primeiro símbolo da fita da MT após o símbolo de início de fita. É necessário então que seja tratado na MT as instruções de desvio/teste, aceitação, rejeição, partida, leitura e escrita;

a) partida: A instrução de partida será simulada pela instrução inicial da máquina de Turing.
b) leitura: A leitura de um símbolo pode ser feita substituindo o símbolo lido por um símbolo vazio, após a escrita do símbolo é re-alocado todos os símbolos lançando-os uma células para a esquerda retirando o símbolo vazio previamente informado, mantendo assim a integridade da fita com a pilha da máquina de post;
c) escrita/atribuição: A escrita sempre deve ser feita sempre ao final da palavra, ou seja, o símbolo será gravado ao final da palavra contida na fita da máquina, após a atribuição o cabeçote da MT será posicionado no início da fita;

d) desvio/teste: é efetuado a leitura e a escrita conforme descrito acima, mantendo a integridade da palavra contida na fita com a pilha da máquina de post;
e) aceitação: Representada por um estado final;
f) rejeição: Representada por um movimento inválido, parando a máquina.

5.3 MÁQUINA DE NORMA VS MÁQUINA DE TURING

Será provada a equivalência entre uma máquina de norma e uma Máquina de Turing, demonstrando que ambas possuem o mesmo poder computacional, ou seja, uma pode simular a outra.

5.3.1 Máquina de Norma → Máquina de Turing

A simulação da máquina de Turing por uma máquina de norma é efetuada de foram simples, devido que a máquina de norma possui uma gama maior de componentes de memória (registradores). A fita da máquina de Turing é simulada pelo registrador X em formato de arranjo unidimensional, ou seja, cada célula da fita corresponde a uma posição do arranjo. Em relação as instruções da MT, a estrutura da máquina de norma já possui instruções rótulas, então para cada instrução da máquina de Turing é atribuído a uma instrução rotulada, onde a primeira instrução da MT (q0) será atribuída ao rótulo 0 e as demais serão atribuídas nos demais rótulos no formato de rótulo final. Para o estado corrente e a cabeça de fita são utilizando para cada um registrador onde para o estado corrente o registrador assume o valor de um rótulo (rótulo corrente), e para o registrador responsável por efetuar a simulação da cabeça de fita, esse será utilizado para guardar a posição corrente do arranjo de X (registrador responsável por armazenar a fita). (DIVERIO; MENEZES, 2000).

5.3.2 Máquina de Turing → Máquina de Norma

Para efetuar a equivalência da Máquina de Turing com a máquina de norma, ou seja, uma máquina de norma ser simulada por uma máquina de Turing é

importante analisarmos que para uma máquina de Norma efetuar qualquer processo são necessários somente dois registradores sendo o X (registrador de entrada) e Y (registrador de saída), ambos em formado de arranjo unidimensional direta, onde cada posição de X = (X_1, $X_{2,}$ $X_{n)}$ seria um registrador. (DIVERIO; MENEZES, 2000).

Assim a máquina de Turing simula os registradores X e Y no seguinte formato. (DIVERIO; MENEZES, 2000):

a) o conteúdo de cada registrador é inserido na fita de norma unitária, o símbolo será repetido quantas vezes forem necessário durante a fita.
b) o registrador X ocupa as células impares da fita e o Y as pares;
c) para cada rotulo da Máquina de Norma é associado um estado da Máquina de Turing.

6 TRABALHOS CORRELATOS

6.1 TEORIA DA COMPUTAÇÃO: MÁQUINAS, PROGRAMAS E SUAS EQUIVALÊNCIAS

O artigo de Pinheiro et al. (2017), intitulado de Teoria da computação: Máquina, Programas e suas Equivalências, tem como foco demonstrar a equivalência entre máquinas e programas, entre os modelos de máquinas universais (Máquina de Turing, Máquina de Post, Autômato de Duas Pilhas e Máquina de Norma).

No artigo os autores apresentam as máquinas universais, sua definição formal e forma de execução, além de ser apresentado as conversões entre as máquinas: Máquina de Turing e Máquina de Post; Máquina de Post e Máquina Norma; Máquina de Norma e Autômato de Duas Pilhas; Autômato de Duas Pilhas e Máquina de Turing.

Como resultado foi provado que todos os modelos são equivalentes. Sendo todas as traduções efetuadas corretamente preservando a linguagem reconhecida pela máquina.

6.2 ANÁLISE DA MÁQUINA DE TURING PERSISTENTE COM MÚLTIPLAS FITAS DE TRABALHO

A dissertação de mestrado de Monica Xavier PY – Análise da Máquina de Turing Persistente com Múltiplas Fitas de Trabalho foi desenvolvida em 2003, tem como foco explorar o paralelismo na Máquina de Turing Persistente (modelo da máquina de Turing capaz de fundamenta a Teoria da Computação interativa), através da formalização de uma extensão paralela da máquina e analisar os efeitos dessa extensão, variando o número de fitas utilizadas no modelo.

A autora durante o trabalho contextualiza conceitos e definições da teoria da computação clássica e da interativa, ressaltando as relações existente entre ambas. Além de descrever a Máquina de Turing Persistente, que serviu de inspiração para a Máquina de Turing Persistente com Múltiplas Fitas, está que é formalizada e definida. Após a definição das máquinas é demonstrado a equivalências entre os modelos.

Como resultado obtido foi observado que nem sempre acrescentar recursos em uma máquina implica no aumento do seu poder computacional, acreditando que as principais contribuições do trabalho são a exploração efetuada pelo mesmo no paralelismo interno de uma Máquina de Turing Persistente, e a demonstração da equivalência entre os modelos demonstrados no trabalho.

6.3 CONSTRUÇÃO DE SIMULADORES GRÁFICOS PARA TEORIA DA COMPUTAÇÃO: UMA PROPOSTA PARA O ENSINO DO CONCEITO DE MÁQUINAS DE TURING

O artigo de Oliveira e Silva (2007), intitulado Construção de Simuladores para Teoria da Computação: Uma proposta para o ensino do Conceito de Máquina de Turing, foi publicado no IV Simpósio de Excelência em Gestão e Tecnologia (SEGT), 2007, o trabalho relata a construção de um sistema que utiliza uma interface gráfica para simular o conceito de uma Máquina de Turing.

Os autores durante o artigo efetuam a fundamentação teórica da Máquina de Turing, além de abordar as vantagens que se tem ao utilizar um simulador durante o ensino, enfatizando ainda o ganho existente caso o simulador seja aplicado para a aprendizagem de conceitos complexos. O protótipo foi desenvolvido em Java e exemplifica os componentes e como foram desenvolvidos.

Para validar o uso do protótipo, esse foi submetido ao uso por alunos do curso de ciência da computação da Faculdade Politécnica de Jundiaí, além de utilizado em outros casos, durante a coleta dos dados validou-se que o protótipo pode auxiliar no ensino da máquina de Turing. Os autores concluíram seus pensamentos analisando que o protótipo deve ser passado por uma gama maior de testes, mas que o mesmo se provou satisfatório nos que foram realizados, além de que o mesmo serve como referência para a criação de diversos outros simuladores com o mesmo intuito de auxiliar no ensino.

7 METODOLOGIA

A partir da fundamentação dos conceitos que englobam as máquinas universais, como complexidade, computabilidade, programa, máquinas e a própria computação, iniciou o estudo das máquinas universais, detalhando os componentes existentes na mesma e a sua execução.

O projeto foi desenvolvido utilizando Play Framework para Java e Angular JS, ambas foram escolhidas pelo fato de serem ferramentas para o desenvolvimento web, com uma curva de aprendizado baixo, robustas e com alto nível de produtividade, sendo o Play Framework responsável pelo back-end da aplicação e o Angular JS pelo front-end. A comunicação entre as duas ferramentas de desenvolvimento acontece via REST. (PIERIN, 2013).

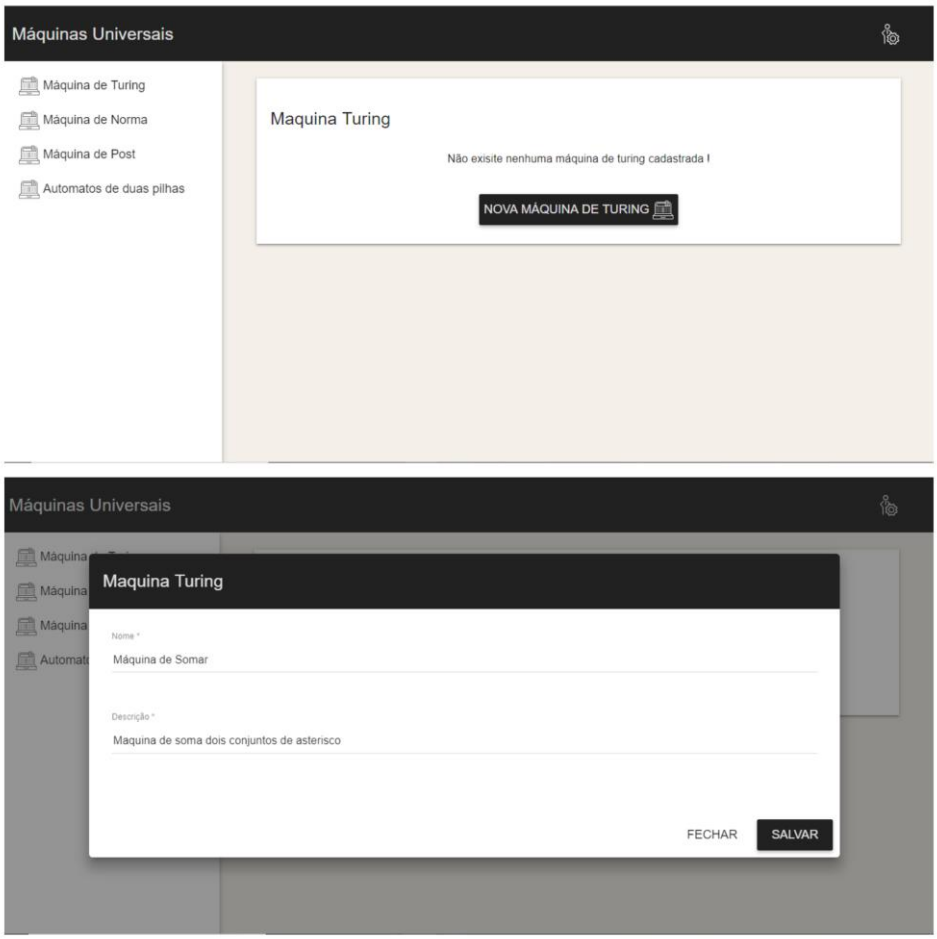
Foi utilizado como ambiente de desenvolvimento e teste, um computador com 16GB de RAM, processador AMD FX™-8350 Eight-Core 4.00 GHz, placa de vídeo Sapphire Dua-X™ R9 270X AMD RADEON R9 200 Series, 120 GB SSD, 1 terabyte de HD. E como Idea de desenvolvimento foi utilizado o IntelliJ 2017.3.4.

O projeto foi dividido em quatro seções principais, sendo cada seção uma máquina universal, para a máquina de Turing e Norma foi desenvolvido telas e formulários responsáveis por cadastrar as informações de seus componentes e transições com o intuído de simular a execução da máquina, para a máquina de Post e Autômato de Duas Pilhas além das telas de cadastro de informação e execução foi validado a conversão dessas para a máquina de Turing, validando assim a teoria de Church sobre as máquinas universais.

7.1 MODELAGEM E EXECUÇÃO DAS MÁQUINAS UNIVERSAIS

Durante a fundamentação do trabalho foram apresentados os componentes de cada máquina universal e sua execução, ou seja, como cada elemento/componente funcionava a partir de uma palavra inicial gerando uma saída após as regras/transições definidas. Devido a isso o protótipo apresenta uma interface onde é possível selecionar a máquina universal e através dela criar modelos para as máquinas trabalhas no protótipo, definido para ela, nome, descrição e seus componentes.

Figura 18 – Tela inicial e diálogo de criação de Máquina Turing



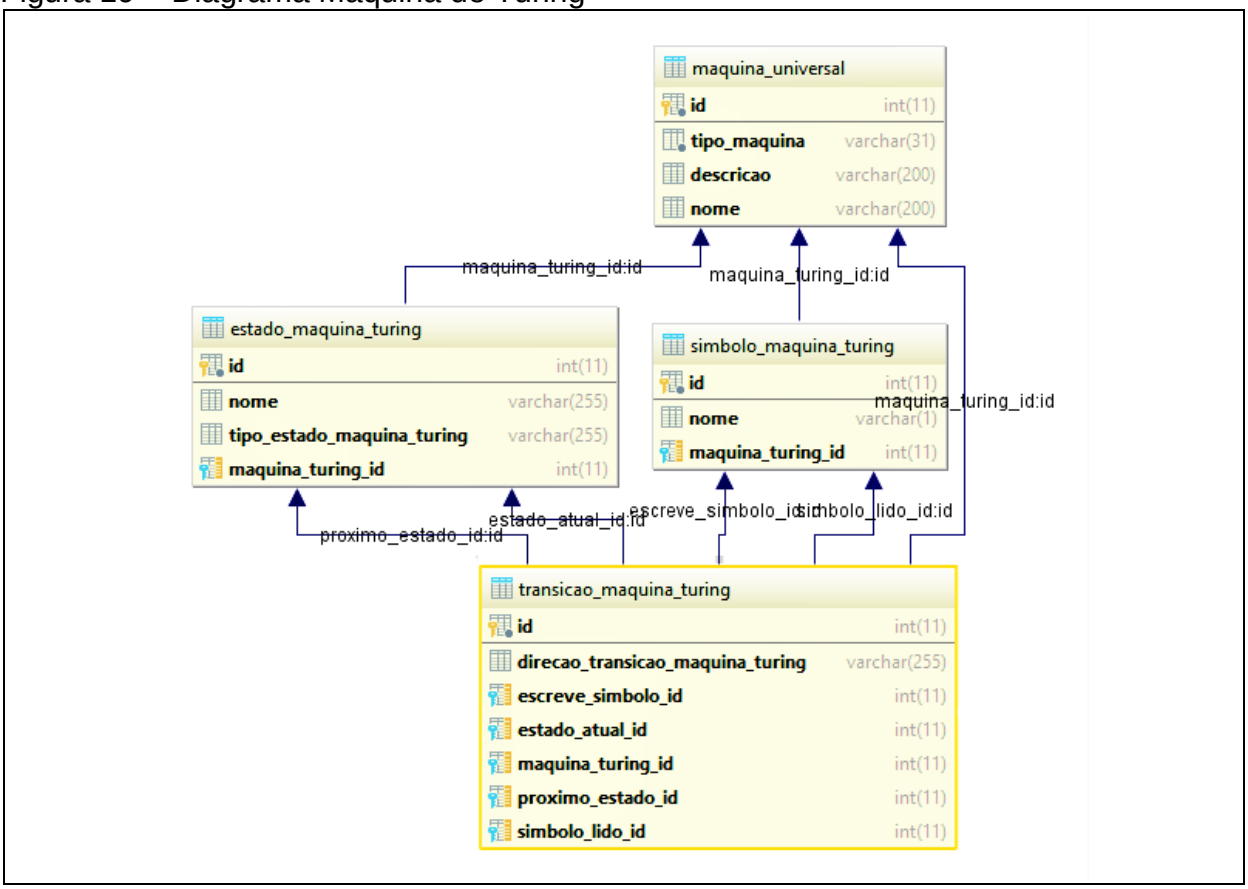
Fonte: Do autor.

A figura 18 apresenta a tela inicial da Máquina de Turing, onde é possível acessar máquinas de Turing existentes e criar novas através do botão “Nova Máquina de Turing”, todas as máquinas universais possuem o mesmo layout e fluxo de dados, a diferença está nos seus componentes e logica de execução, sendo esses vistos nos próximos capítulos.

7.2 MÁQUINA DE TURING

Para o desenvolvimento da Máquina de Turing foram criados as entidades, símbolos (para representar o alfabeto da máquina de Turing), estados (podendo ser do tipo normal, final e inicial) e transições (define como será o fluxo da MT, a partir de um estado e um símbolo lido na fita), sendo que as informações adicionais da máquina como nome e sua descrição é atributo da entidade maquina universal, o tipo desta máquina é informado como Máquina de Turing.

Figura 19 – Diagrama Máquina de Turing

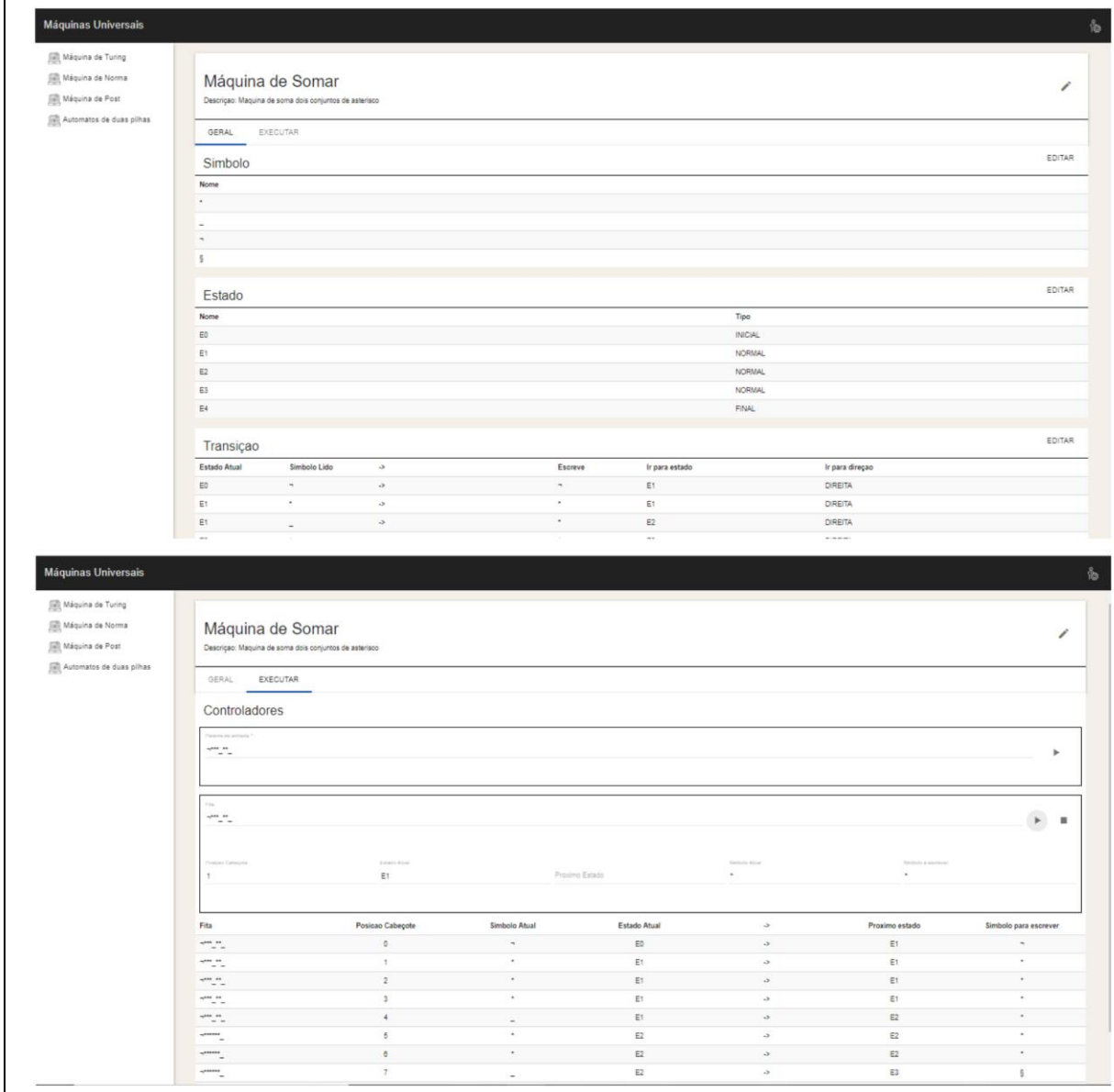


Fonte: Do autor.

Na figura 19 é apresentado o diagrama criado para suprir os componentes da MT, esses que são salvos na aplicação após passarem por uma validação com o intuito de manter os dados da MT íntegros, é controlado para existir um único estado inicial, o alfabeto não possuir símbolos repetidos e as transições não serem ambíguas.

Cada máquina universal é dívida em seções, na figura 20 é apresentado a seção geral de uma MT onde é localizado os formulários de seus componentes, sendo apresentados eles em listas, com a possibilidade de edição e adição de novos componentes (símbolos, estados e transições), tanto a edição como adição são efetuadas em modais, formulários com os campos necessários para que o componente funcione adequadamente, a próxima seção apresentada na figura 20 é a seção de execução, nela é informada uma palavra, caso essa seja aceita na máquina será gerado uma tabela com o histórico da execução da máquina, e um ferramenta para que seja possível verificar o funcionamento da mesma. Para as palavras que não forem aceitas será sinalizado através de uma mensagem qual o erro que aconteceu na execução.

Figura 20 – Seção geral e executar da Máquina de Turing



Fonte: Do autor.

Na figura 21 é apresentado o trecho de código responsável por efetuar a execução da máquina de Turing. Ele inicia configurando as dados da máquina, ou seja, acola em uma variável responsável por representar a fita a palavra informada, seta a posição atual do cabeçote como zero, além de buscar todos os símbolos, estados e transições existentes na máquina. Após isso é buscado pelo estado inicial da máquina, esse que é alocado em uma variável chamada estado atual, a variável estado atual juntamente com a posição atual, são responsáveis pela unidade de controle da MT, enquanto o estado atual não for final a máquina busca na lista de transição, uma transição do estado atual com leitura para o símbolo encontrado na

fita na posição atual do cabeçote, caso a transição seja encontrada, essa informa o próximo estado atual, escreve na fita o símbolo informado na posição do cabeçote, além de que possuir a direção do cabeçote, caso a direção seja direita será acrescentado 1 a posição atual, caso for esquerda será decrescendo 1 da posição atual. Se não for encontrado a transição ocorrerá um indeterminismo na máquina, encerrando a execução da mesma.

Ao final da execução caso esse seja completado com sucesso é informado para o usuário do protótipo os passos que a MT vez até chegar no resultado final, caso aconteça algum indeterminismo, será informado qual símbolo e qual estado gerou o mesmo, demais erros também são lançados para o usuário.

Figura 21 – Trecho código execução Máquina de Turing

```
List<EstadoMaquinaTuring> estadosMaquinasTuring = maquinaTuring.getEstadosMaquinasTuring();
List<SimboloMaquinaTuring> simbolosMaquinasTuring = maquinaTuring.getSimbolosMaquinasTuring();
List<TransicaoMaquinaTuring> transicoesMaquinasTuring = maquinaTuring.getTransicoesMaquinasTuring();

StringBuilder fita = new StringBuilder(palavra);
Integer posicaoAtual = 0;

EstadoMaquinaTuring estadoAtual = estadosMaquinasTuring.stream().filter(EstadoMaquinaTuring::isInicial).findFirst().orElse(null);

if (estadoAtual == null) {
    throw new ApplicationException("Não foi definido o estado inicial da máquina de turing!");
}

while (!estadoAtual.isFinal()) {

if (fita.length() < (posicaoAtual + 1)) {
    fita.append(SimboloEspecialMaquinaUniversal.SIMBOLO_VAZIO.getValor());
}

String simboloFita = String.valueOf(fita.charAt(posicaoAtual));
EstadoMaquinaTuring finalEstadoAtual = estadoAtual;
TransicaoMaquinaTuring transicaoMaquinaTuring = transicoesMaquinasTuring.stream().filter(transicao ->
        transicao.getEstadoAtual().equals(finalEstadoAtual) && transicao.getSimboloLido().getNome().equals(simboloFita))
        .findFirst().orElse(null);

if (transicaoMaquinaTuring == null) {
    throw new ApplicationException("Não foi encontrado uma transição para " + estadoAtual.getNome() +" - " + simboloFita);
}

EstadoMaquinaTuring proximoEstado = transicaoMaquinaTuring.getProximoEstado();
Integer changePosition = transicaoMaquinaTuring.getDirecaoTransicaoMaquinaTuring().getChangePosition();

fita.setCharAt(posicaoAtual, transicaoMaquinaTuring.getEscreveSimbolo().getNome().charAt(0));
estadoAtual = proximoEstado;
posicaoAtual = posicaoAtual + changePosition;
}
```

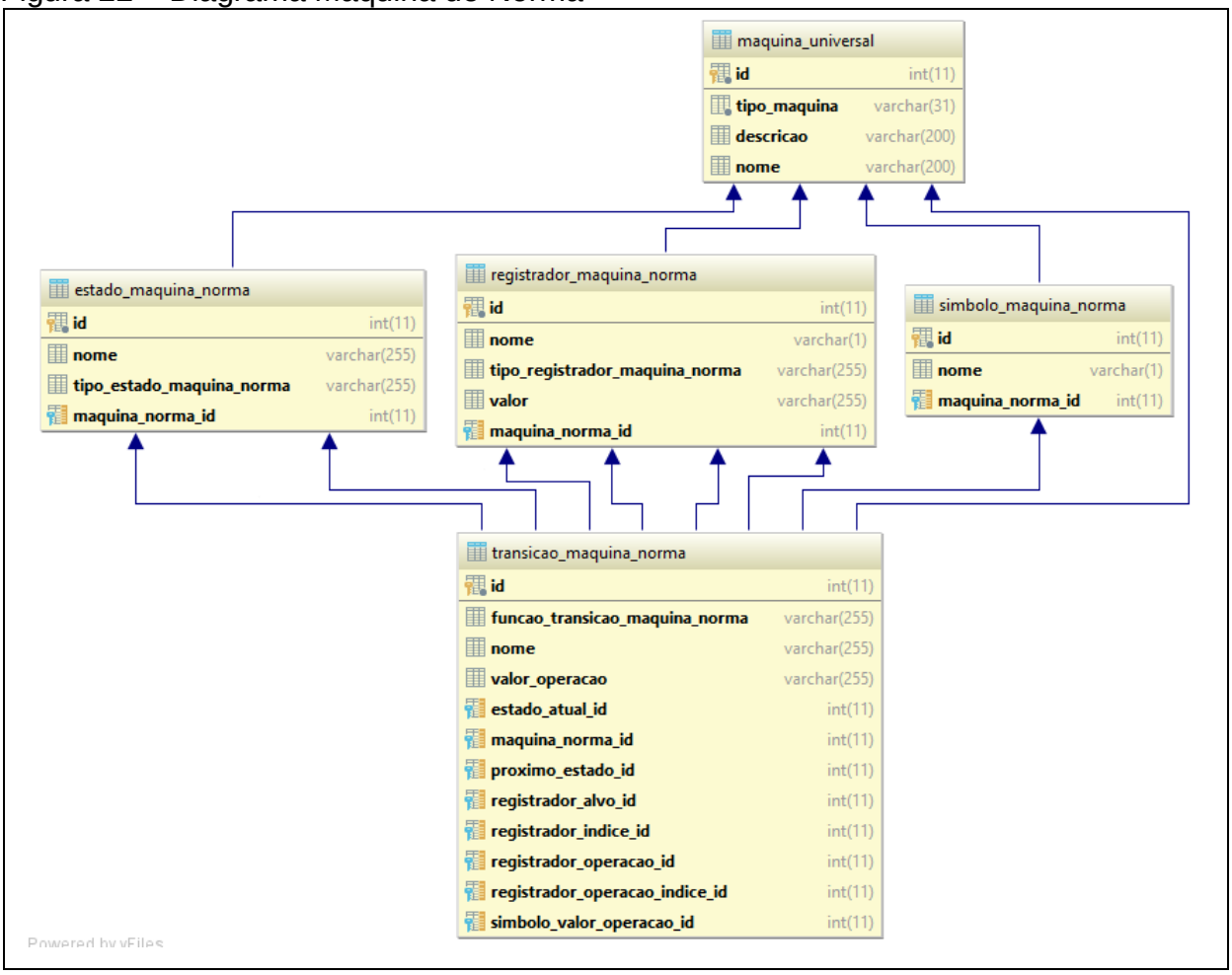
Fonte: Do autor.

7.3 MÁQUINA DE NORMA

De forma semelhante a Máquina de Turing foi criado entidades relacionais para suprir as necessidades e simbolizar os componentes da Máquina de Norma, sendo esses, símbolos (para representar o alfabeto da máquina), estados (representando os rótulos, podendo ser do tipo normal, final e inicial), transições (instruções que definem como será o fluxo da Máquina de Norma, a partir de um

estados (rótulo)) e registradores (responsáveis por armazenar os valores), sendo que as informações adicionais como nome e sua descrição é atributo da entidade maquina universal, o tipo desta máquina é Máquina de Norma.

Figura 22 – Diagrama Máquina de Norma



Fonte: Do autor.

Na figura 22 é apresentado o diagrama criado para suprir os componentes da máquina. Da mesma forma que a MT a Máquina de Norma foi dividida em duas seções, geral e execução, respectivamente a primeira é responsável por apresentar os componentes e efetuar a criação e edição dos mesmos, sendo a segunda responsável por executar a máquina previamente configurada na primeira seção a partir do valor informado no registrador X.

Foi mapeado para uma transição/instrução Norma as funções de adição, subtração, atribuição de valor e teste, sendo q um estado pode possuir somente um único tipo de instrução e somente é aceito que um estado tenha mais de uma transição se suas transições forem do tipo teste. Para os registradores esses podem

armazenar um único valor ou um conjunto de valores (registradores do tipo array), os elementos armazenados em um registrador são os pertencentes em seu alfabeto ou pertencentes ao conjunto dos números naturais.

Na seção de execução da máquina de norma, caso a máquina chegue a um estado final é retornado o histórico de execução da máquina, mostrando todos as instruções que foram executas na máquina e os registradores que foram modificados, além de que é possível na ferramenta que executa o histórico da máquina visualizar os valores de cada registrador no momento de execução de cada instrução.

Figura 23 – Algoritmo de execução da Máquina de Norma

```
List<EstadoMaquinaNorma> estadosMaquinaNorma = maquinaNorma.getEstadosMaquinaNorma();
List<RegistradorMaquinaNorma> registradoresMaquinaNorma = maquinaNorma.getRegistradoresMaquinaNorma();
List<TransicaoMaquinaNorma> transicoesMaquinaNorma = maquinaNorma.getTransicoesMaquinaNorma();
Boolean executando = Boolean.TRUE;
RegistradorMaquinaNorma registradorX; // busca pelo registrador x, caso esse não exista retorna um erro
List<EstadoMaquinaNorma> estadosIniciais; // busca pelo estado inicial, caso ele não exista, ou seja maior que 1 retorna erro
List<TransicaoMaquinaNorma> transicoesInciais; // busca pelas transições do estado
TransicaoMaquinaNorma transicaoAtual = transicoesInciais.stream().findAny().orElse(null);

while (executando) {

FuncaoTransicaoMaquinaNorma funcao = transicaoAtual.getFuncaoTransicaoMaquinaNorma();
RegistradorMaquinaNorma registradorAlvo = transicaoAtual.getRegistradorAlvo();
switch (funcao) {
    case ARMAZENAR_VALOR:
        registradorAlvo.setValor(execucaoAcaoArmazenaValores(transicaoAtual));
        break;
    case TESTE:
        transicaoAtual = execucaoAcaoTeste(transicaoAtual, maquinaNorma); // Executa os testes do estado e encontra o teste verdadeiro
        break;
    case SUBTRACAO:
        // Logica que subtrai valor do registrador da operação
        break;
    case ADICAO:
        // Logica que adiciona valor do registrador da operação
        break;
}
TransicaoMaquinaNorma finalTransicaoAtual = transicaoAtual;
if (finalTransicaoAtual == null) {
    executando = false;
} else {
    } else {
        if (finalTransicaoAtual.getProximoEstado().getTipoEstadoMaquinaNorma() == TipoEstadoMaquinaNorma.FINAL) {
            executando = false;
        }
    }
}
transicaoAtual = transicoesMaquinaNorma.stream().filter(t -> t.getEstadoAtual().equals(finalTransicaoAtual.getProximoEstado()))
.findAny().orElse(null);
```

Fonte: Do autor.

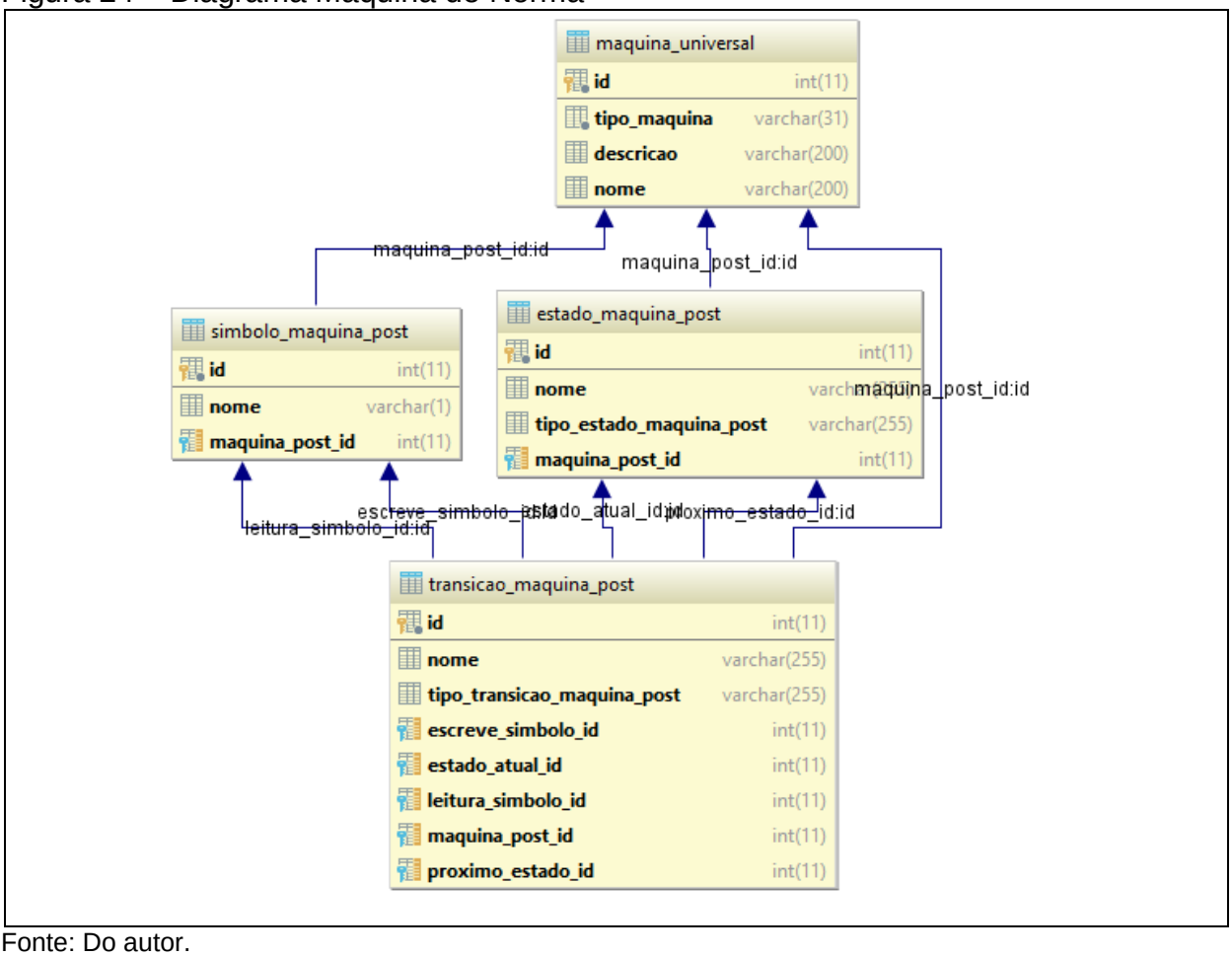
O algoritmo responsável pela execução da Máquina de Norma apresentado na figura 23, inicialmente verifica se todas as condições para a execução são satisfeitas, sendo essas, a existência de um registrador X, um único estado inicial, se o estado inicial possuí uma transição, após isso é executado as instruções, alterando os registradores alvos caso a função seja do tipo de atribuição de valor, subtração e adição, e efetuando testes na máquina para a tomada de decisão de qual a instrução a mesma deve ir, o algoritmo executa até que seja

encontrado um estado/rotulo final ou aconteça um indeterminismo, sendo esse apresentado em formato de erro.

7.4 MÁQUINA DE POST

A máquina de Post foi desenvolvida em três seções, geral, executar e converter, as seções, geral e executar são similares a da Máquina de Turing e Máquina de Norma, entretanto adaptado para os componentes da Máquina de Post, sendo esses estados (podem ser partida/inicial, normal, aceita/final, rejeita/final), símbolos e transições (podem ser escrita e leitura/teste).

Figura 24 – Diagrama Máquina de Norma



Fonte: Do autor.

Na figura 24 é possível verificar a estrutura dos componentes da máquina de Post. Durante a fundamentação teórica da Máquina de Post, foi apresentado a máquina a partir de um diagrama de fluxo, onde a própria transição seria o rótulo/ estado, ou seja, programa composto somente de símbolos e transições (diagrama

de fluxo), durante o desenvolvimento da máquina e sua conversão optou-se por fragmentar o diagrama em estado e transição, para aproximar o modelo de Post com o de Turing, além de simplificar a busca pela próxima transição da máquina, as mudanças não afetam o funcionamento da máquina nem sua execução.

A execução da máquina de Post segue os princípios de uma fila (FIFO), como citado acima as transições podem ser, escrita e leitura/teste, o algoritmo que executa a máquina previamente cadastrada na seção geral, busca a transição do estado inicial, caso seja encontrada a máquina irá executar removendo o primeiro elemento da palavra e realocando os demais para transição de leitura/teste, e adicionando símbolos pertencentes ao alfabeto da máquina ao final da palavra quando a transição por de escrita, até o momento em que essa encontra um estado final (aceitação/rejeição). De forma semelhante a máquina de norma, os estados devem possuir somente um único tipo de transição, sendo que somente estados que possuem transição de leitura/teste podem possuir mais de uma transição, isso se dá ao fato que a máquina pode seguir diferentes caminhos de acordo com o símbolo lido na fila. Na figura 25 é apresentado trecho de código da execução da Máquina de Post.

Figura 25 – Trecho de código da Máquina de Post

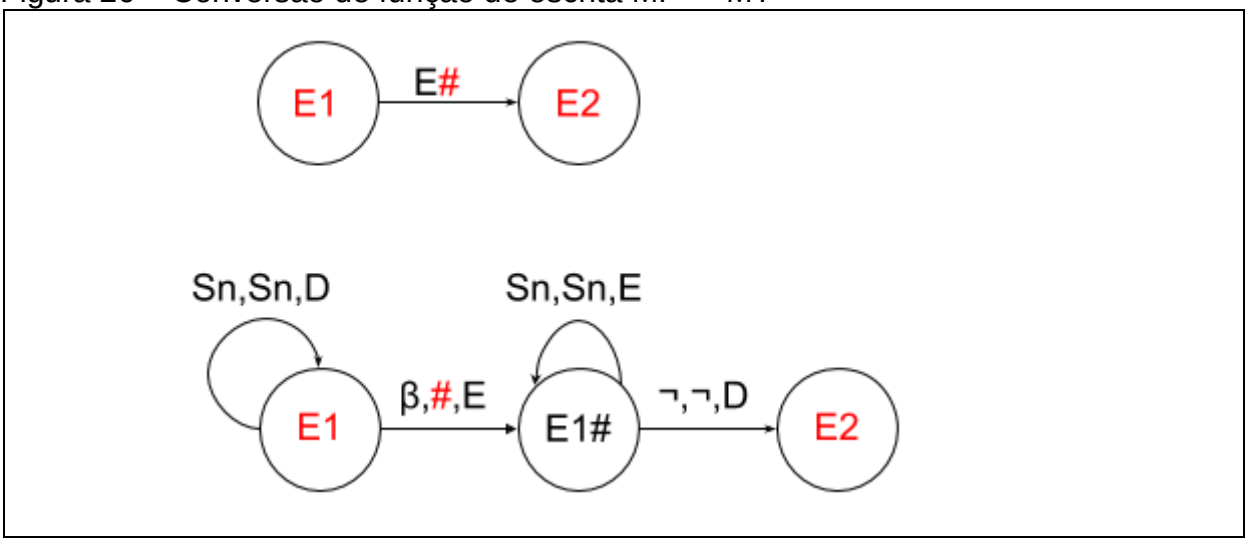
```
StringBuilder fila = new StringBuilder(palavra);
Boolean executar = Boolean.TRUE;
TransicaoMaquinaPost transicaoAtual = null;

List<EstadoMaquinaPost> estadosIniciais = maquinaPost.getEstadosMaquinaPost().stream()
    .filter(EstadoMaquinaPost::isInicial).collect(Collectors.toList());

// verifica se estado inicial não é facil e se existe somente um unico estado inicial

EstadoMaquinaPost estadoInicial = estadosIniciais.stream().findFirst().orElse(null);
transicaoAtual = getProximaTransicao(estadoInicial, maquinaPost, fila);

while (executar) {

    switch (transicaoAtual.getTipoTransicaoMaquinaPost()) {

        case LEITURA_TESTE:
            fila = new StringBuilder(fila.substring(1));
            break;

        case ESCRITA:
            SimboloMaquinaPost escreveSimbolo = transicaoAtual.getEscreveSimbolo();
            if (escreveSimbolo == null) {
                throw new ApplicationException("Deve ser informado o simbolo a ser escrito em uma transiçao de escrita,
                 erro na transiçao: " + transicaoAtual.getNome());
            }
            fila.append(escreveSimbolo.getNome());
            break;
    }

    if (transicaoAtual.getProximoEstado().isFinal()) {
        transicaoAtual = null;
        executar = false;
    } else {
        // defini a proxima transição, verifica as transições do estado atual, e encontra a correta
        transicaoAtual = getProximaTransicao(transicaoAtual.getProximoEstado(), maquinaPost, fila);
    }
}
.findAny().orElse(null);
```

Fonte: Do autor.

7.4.1 Conversão Máquina de Post ⇾ Máquina de Turing

Para validar a Tese de Church previamente fundamentada, foi desenvolvido a conversão para as máquinas de Post, essa baseia-se em desenvolver estruturas que comportem cada transição de Post em Turing, efetuando o funcionamento da máquina em sua máquina alvo, ou seja, além da MT chegar ao mesmo resultado que a MP, essa deve simular o funcionamento em fila da MP.

Figura 26 – Conversão de função de escrita MP ⇾ MT

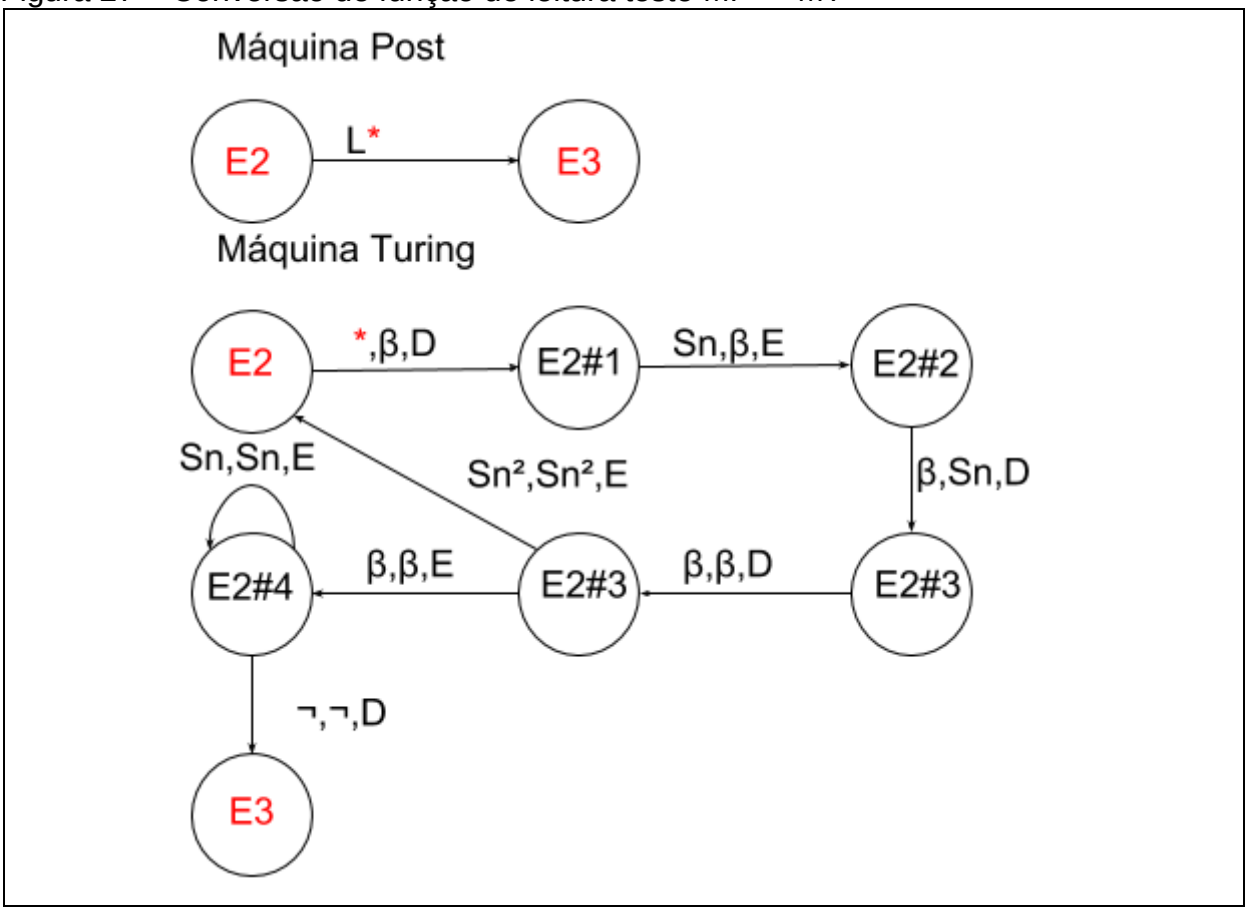


Fonte: Do autor.

Na figura 26 são demonstrados os procedimentos para a conversão de uma transição de escrita desenvolvida em uma Máquina de Post para uma Máquina de Turing, o primeiro grafo representa a transição em Post e o segundo em Turing. Como citado anteriormente é necessário que a máquina alvo além de produzir o mesmo resultado simule a execução da máquina a ser simulada (convertida), os dados na imagem em vermelho representam os símbolos que geram a escrita na máquina, ou seja, E1 é o estado inicial que escreve # no final da fita (MT) ou fila (MP) indo para E2, as demais transições existentes na transição da máquina de Turing são para simular o funcionamento de fila de Post, sendo Sn usado para representar que deve ter uma transição para todos os símbolos do alfabeto, ou seja, para que a máquina de Turing simule uma ação de escrita em Post ela deve percorrer toda a fita até encontrar um símbolo vazio (β) onde irá escrever na fita o símbolo indicado na transição de escrita, após isso deve voltar até encontrar o símbolo inicial da MT (representando por ¬), endereçando assim o próximo estado,

com o cabeçote localizado no primeiro elemento da palavra após o símbolo inicial da máquina.

Figura 27 – Conversão de função de leitura teste MP ⇒ MT



Fonte: Do autor.

Na figura 27 são demonstrados os procedimentos para a conversão de uma transição do tipo leitura/teste, o primeiro grafo representa uma transição do tipo leitura/teste em uma MP já o segundo grafo representa a mesma transição simulando o comportamento da MP em uma MT. A transição a ser simula efetua o seguinte procedimento, no estado E2 caso o símbolo do topo da pilha seja * retira-o do topo e vai para o estado E3, os símbolos e estados envolvidos estão demarcados de vermelho, os demais estados e transições são auxiliares para efetuar a simulação da execução da MP pela MT.

Para a MT simular a execução de uma transição citada acima, essa deve remover o símbolo lido, avançar para a direita da fita remover novamente o símbolo e escrevê-lo na posição anterior, até encontrar um símbolo vazio a direita, afim de realocar todos os símbolos existentes na fita (para simular o comportamento de fila),

após isso deve voltar até encontrar o símbolo inicial da MT (representando por ¬), endereçando assim o próximo estado, com o cabeçote localizado no primeiro elemento da palavra após o símbolo inicial da máquina.

Na conversão da MP para MT além de cada transição passar pelos processos descritos é adicionado no alfabeto de símbolos da máquina o símbolo de vazio caso essa não possua (representando na aplicação pelo símbolo §), e o símbolo inicial da MT (representado na aplicação pelo símbolo ¬). O tipo de estado é mantido na conversão, entretanto um estado pode possuir estados auxiliares como foi descrito acima, esses que são estados do tipo normal.

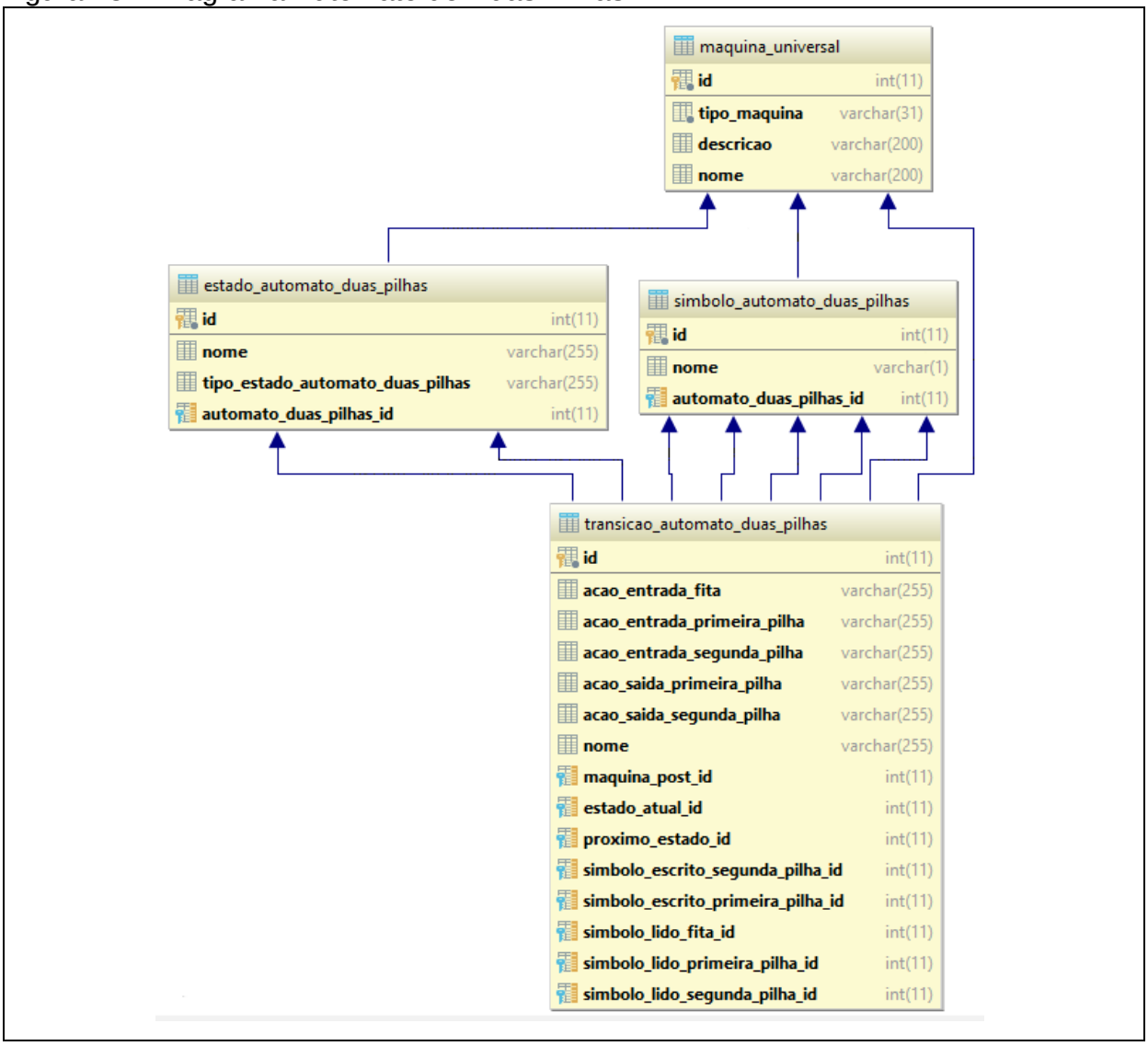
A máquina convertida será encontrada na listagem das máquina de Turing.

7.5 AUTÔMATO DE DUAS PILHAS

Da mesma forma que a Máquina de Post foi desenvolvida o Autômato de Duas Pilhas com as seções gerais, executar e converter, afim de validar a teoria de Church, efetuando a simulação da máquina universal Autômato de Duas Pilhas em uma Máquina de Turing. Seguindo o mesmo princípio que as demais máquinas, na seção geral é fundamentado os componentes da máquina, afim de fornecer ferramentas para criar um modelo de um autômato de duas pilhas e posteriormente ser executado na seção de mesmo nome a partir de uma palavra definida pelo usuário, demostrando os resultados e passos que a máquina passou caso encontre um final em sua execução, ou mostrando o erro ocorrido durante o processo.

Os componentes criados foram estados (normal, final e inicial), símbolos (alfabeto da máquina) e transições (responsável por definir o fluxo da máquina). A transição desenvolvida pode ser dividia em duas etapa, ações de entrada/teste e de saída, sendo que essas podem ser, retirar símbolo especifico (ação de entrada), retirar qualquer símbolo (ação de entrada), não efetua leitura (ação de entrada), empilha símbolo especifico (ação de saída) e não empilha símbolo (ação de saída). Existe três ações de entrada e duas ações de saída, conforme é demonstrado na figura 28 onde é possível ver os componentes desenvolvidos para o Autômato de Duas pilhas.

Figura 28 – Diagrama Autômato de Duas Pilhas



Fonte: Do autor.

Após a definição da máquina foi desenvolvido a execução da máquina, lembrando que a estrutura do Autômato de Duas Pilhas baseia-se em pilhas responsáveis por armazenar as informações da máquina, podendo exercer tanto leitura como escritas na mesma, caso a leitura seja executada o símbolo lido é destruído, ou seja, removido da pilha, a palavra a ser executada na máquina é armazena em uma fita, a leitura da fita é destrutiva como na pilha entretanto não é possível escrever na fita, sendo então utilizada somente para a tomada de decisão de qual transição deve ser utilizada estando em um determinado estado, lembrando que a decisão é tomada após a leitura de um símbolo especifico, qualquer símbolo ou não leitura da fita e das duas pilhas, podendo a partir da decisão ser empilhado um símbolo ou não nas duas pilhas. Visando isso foi desenvolvido um algoritmo que

a partir do estado inicial da máquina encontra uma transição que satisfaça a situação dos componentes da máquina, passando entre os estados modificando as pilhas até chegar em um estado final ou gerar um não determinismo ou erro de execução. Na figura 29 é possível visualizar algumas partes desse código e o fluxo que o mesmo segue.

Figura 29 – Trecho do algoritmo responsável pela execução dos Autômatos de duas pilhas

```
PilhaDinamica primeiraPilha = new PilhaDinamica(); PilhaDinamica segundaPilha = new PilhaDinamica();
EstadoAutomatoDuasPilhas estadoAtual =  //busca pelo estado inicial da fita
while (!estadoAtual.isFinal()) {
    String simboloAtual = // pego o símbolo atual da fita ou vazio caso esse não exista
    TransicaoAutomatoDuasPilhas transicaoAutomatoDuasPilhas = null; // transição atual
    List<TransicaoAutomatoDuasPilhas> transicoesPossiveis = new ArrayList<>(); // lista usada para trazer as transições possíveis
    transicoesPossiveis = // pega todas as transições possiveis a partir do símbolo da fita
    if (transicoesPossiveis.isEmpty()) {
        // apresenta erro caso não seja encontrado uma transição
    }
    if (transicoesPossiveis.size() > 1) {
        String primeiraPilha = // pega o símbolo atual da primeira pilha ou vazio caso esse não exista
        String segundaPilha = // pega o símbolo atual da segunda pilha ou vazio caso esse não exista
        transicoesPossiveis = // refina as transições encontradas validando com os dados das pilhas
    }
    // caso não exista transições validas ou exista mais que uma transição a execução para por não determinismo
    transicaoAutomatoDuasPilhas = transicoesPossiveis.get(0);

    // remove da fita/palavra o símbolo lido caso
    if (transicaoAutomatoDuasPilhas.getAcaoEntradaFita().equals(AcaoComponenteAutomatoDuasPilhas.QUALQUER_SIMBOLO) ||
            transicaoAutomatoDuasPilhas.getAcaoEntradaFita().equals(AcaoComponenteAutomatoDuasPilhas.RETIRAR_SIMBOLO_ESPECIFICO)) {
        if (palavra != null && !palavra.isEmpty()) {
            palavra = palavra.substring(1);
        }
    }
    if (transicaoAutomatoDuasPilhas.getAcaoEntradaPrimeiraPilha().equals(AcaoComponenteAutomatoDuasPilhas.QUALQUER_SIMBOLO) ||
            transicaoAutomatoDuasPilhas.getAcaoEntradaPrimeiraPilha().equals(AcaoComponenteAutomatoDuasPilhas.RETIRAR_SIMBOLO_ESPECIFICO)) {
        if (primeiraPilha.toString() != null && !primeiraPilha.toString().isEmpty()) {
            primeiraPilha.pop();
        }
    }
    // empilha símbolo especifico nas pilhas caso tipo de transição de saída faça isso
    if (transicaoAutomatoDuasPilhas.getAcaoSaidaPrimeiraPilha().equals(AcaoComponenteAutomatoDuasPilhas.EMPILHA_SIMBOLO_ESPECIFICO)) {
        primeiraPilha.push(transicaoAutomatoDuasPilhas.getSimboloEscritoPrimeiraPilha().getNome());
    }

    estadoAtual = transicaoAutomatoDuasPilhas.getProximoEstado();
}
```

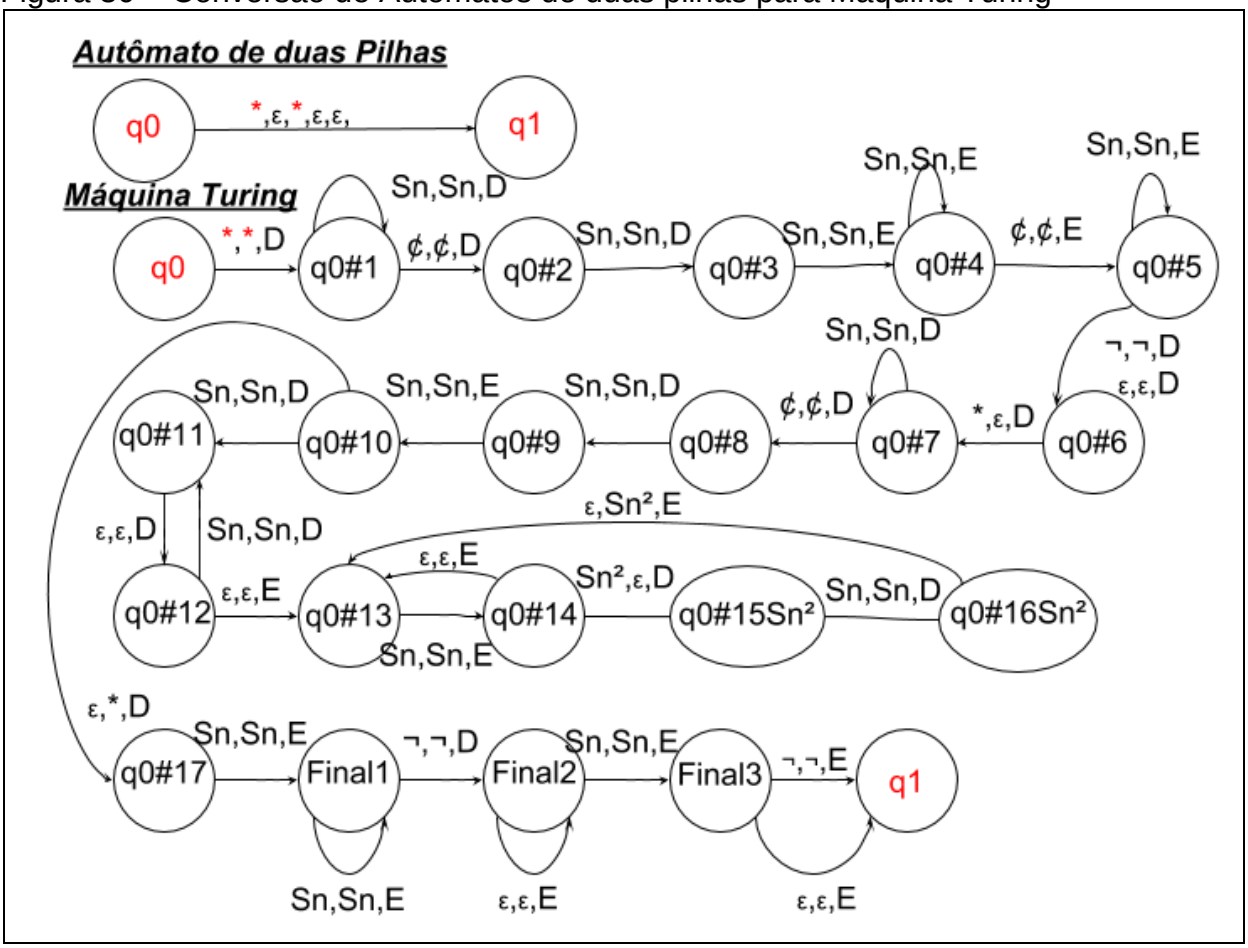
Fonte: Do autor.

7.5.1 Conversão Máquina de Post → Máquina de Turing

Além da conversão de Post para Turing, foi desenvolvida a conversão de Autômato de Duas para Máquina de Turing, com o mesmo intuito de validar a tese de Church, como foi visto na conversão de Post, é necessário que a máquina simule tanto seus componentes como seu comportamento gerando ao final o mesmo resultado. Entretanto diferente da Máquina de Post, que possui os mesmo componentes da Máquina de Turing (fita, estado e Símbolo) diferenciando somente seu comportamento, autômatos de duas pilhas possui tanto um comportamento diferente como seus componentes também, trazendo assim uma complexidade maior para a simulação, como foi visto durante a fundamentação teórica para uma Máquina de Turing simular uma Máquina Universal de Autômatos de Duas Pilhas, a

fita da máquina será responsável por representar tanto a palavra informada, como as duas pilhas, para que isso ocorra é adicionado ao alfabeto da máquina um símbolo auxiliar ao final da palavra essa alocada na fita, símbolo esse representando na aplicação por ¢, esse é responsável por demarcar o final da palavra e o início das pilhas, a separação das pilhas é feita de uma forma que a primeira pilha corresponde às células ímpares da fita após o símbolo de auxiliar, e a segunda pilha os símbolos pares após o símbolo auxiliar.

Figura 30 – Conversão de Autômatos de duas pilhas para Máquina Turing



Fonte: Do autor.

Na figura 30 é possível encontrar um trecho da simulação de uma transição de Autômato de duas pilhas em uma máquina de Turing, essa tem o papel de ao ler o símbolo “*” independente dos dados alocados nas pilhas, empilhas “*” na primeira pilha. Para simular isso foi desenvolvido para a máquina de Turing uma serie de estados para efetuar as funções existentes nas transições, lembrando que como é possível as transições executarem diferentes ações, teve-se que desenvolver algo genérico e que se adapte à realidade de cada transição, na figura

30, dos estados q0#1 a q0#4, a máquina verifica se a máquina atende as configurações da transição, q0#4 a q0#6 é buscado o primeiro símbolo existe na fita, q0#6 a q0#7 remove o símbolo da fita, q0#7 a q0#8 é encontrada a primeira fila, q0#8 a q0#9 não remove símbolo da pilha, q0#9 a q0#10 não remove símbolo da segunda pilha, q0#10 a q0#17 escreve símbolo na primeira pilha, sendo que se a primeira posição já estiver sendo ocupada será realocado os demais e o símbolo indicado na transição será alocado na primeira posição, final1 a final3 encontra o início para o próximo estado da transição. Além de que algumas funções mapeadas não foram exemplificadas pela figura 30, como o controle e escrita da segunda pilha e a remoções de símbolos da pilha.

Na conversão da máquina de Autômatos finitos para MT além de cada transição passar pelos processos descritos é adicionado no alfabeto de símbolos da máquina o símbolo de vazio caso essa não possua (representando na aplicação pelo símbolo §), símbolo inicial da MT (representado na aplicação pelo símbolo ¬) e o símbolo auxiliar que marca o final da palavra e o início das pilhas (representado na aplicação pelo símbolo ¢). O tipo de estado é mantido na conversão, entretanto um estado pode possuir estados auxiliares como foi descrito acima, esses que são estados do tipo normal.

A máquina convertida será encontrada na listagem das máquina de Turing.

8 RESULTADOS OBTIDOS

Após o desenvolvimento do protótipo, foi cadastrado para cada máquina universal, transições, estados e símbolos responsáveis por unir dois conjuntos de ‘*’ separados por ‘_’.

Para a Máquina de Turing foram cadastrados cinco estados (q0, q1, q2, q3 e q4), sendo q0 estado inicial e q4 estado final, seu alfabeto possui os símbolos (*, _, §, ¬), lembrando que § corresponde ao símbolo vazio e ¬ ao símbolo de início da máquina, necessário para o funcionamento da MT, o conjunto de transições cadastrado foi ((q0, ¬, *, q1, Direita), (q1, *, *, q1, Direita), (q1, _, *, q2, Direita), (q2, *, *, q2, Direita), (q2, _, _, q3, Esquerda), (q2, §, §, q3, Esquerda), (q3, * _, q4, Esquerda)), contabilizando um total de sete transições, sendo a definição da transição (estado atual, símbolo lido, símbolo gravado na fita, próximo estado, direção do cabeçote).

Para a Máquina de Post foram cadastrados cinco estados (q0, q1, q2, q3 e q4), sendo q0 o estado de partida e q4 estado de aceitação, seu alfabeto possui os símbolos (*, _, §), lembrando que § corresponde ao símbolo vazio, o conjunto de transições cadastrado foi ((q0, *, leitura, q1), (q1, * escrita, q0), (q0, _, leitura, q2), (q2, *, leitura, q3), (q3, *, escrita, q2), (q2, _, leitura, q4)), contabilizando um total de seis transições, sendo a definição da transição (estado atual, símbolo operação, operação máquina, próximo estado).

Para Autômato de Duas Pilhas foram cadastrados três estados (q1, q2 e q3), sendo q1 o estado inicial e q3 o estado final, seu alfabeto possui os símbolos (*, _), o conjunto de transições cadastrado foi ((q1, *, ε, *, ε, ε, q1), (q1, _, ε, ε, ε, ε, ε, q2), (q2, *, ε, *, ε, ε, ε, q2), (q2, _, ε, ε, ε, ε, ε, q3)), contabilizando um total de quatro transições, sendo a definição da transição (estado lido, símbolo lido fita, símbolo lido da pilha 1, símbolo gravado da pilha 1, símbolo lido da pilha 2, símbolo gravado da pilha 2, próximo estado).

Para Máquina de Norma forma cadastrados dez estados (q#, q0, q1, q2, q3, q4, q5, q6, q7, q8 e q9) sendo o q# estado inicial e q9 estado final, seu alfabeto possui os símbolos (*, _), possuindo quatro registradores (X, Y, C e Z), com X e Y sendo do tipo Array e C e Z podendo armazenar somente um valor, o conjunto de transições cadastrado foi ((q#, Armazenar, C, 0, q1), (q0, Teste, X[C], *, q1), (q1, Armazenar, Y[Z], X[C], q2), (q2, Adição, C, (1,C), q3), (q3, Adição, Z, (1,Z), q0), (q0,

Teste, X[C], *, q1), (q1, Armazenar, Y[Z], X[C], q2), (q2, Adição, C, (C,1), q3), (q3, Adição, Z, (Z,1), q0), (q0, Teste, X[C], _, q4), (q4, Adição, C, (C,1), q5), (q5, Teste, X[C], *, q6), (q6, Armazenar, Y[Z], X[C], q7), (q7, Adição, C, (C,1), q8), (q8, Adição, Z, (Z,1), q5), (q5, Teste, X[C], _, q9)), contabilizando um total de doze transições, sendo a definição da transição (estado atual, operação, operação máquina, próximo estado).

As máquinas construídas executaram as seguintes palavras **_**, **_*, _**, _*, **__, podendo assim efetuar a união de dois conjuntos de *, separados por _, lembrando que na Máquina de Turing foi adicionado ‘¬’ a palavra antes de sua execução. Após as execuções foi efetuado as conversões sendo que essas aceitaram as mesmas palavras das máquinas simuladas, ou seja, simuladas corretamente por outro modelo.

9 CONCLUSÃO

Esse trabalho teve como objetivo apresentar conceitos e definições existentes na área da teoria da computação relacionados as máquina universais e validar através de simulação/conversão a tese de Church, desenvolvendo quatro máquinas universais (Máquina de Turing, Máquina de Post, Máquina de Norma e Autômato de Duas Pilhas) e efetuando conversões de Post para Turing e Autômato de Duas Pilha para Turing.

O principal questionamento/objetivo foi atendido ao ser desenvolvido um protótipo responsável por apresentar componentes responsáveis pela criação das máquina e as conversões/simulações efetuadas.

A validação da tese de Church foi devidamente realizada através simulação da Máquina de Post e Autômato de Duas Pilhas, pelo fato que a primeira máquina possui um comportamento diferente da Máquina de Turing e a segunda possui uma maior diversidade de componentes (estrutura da máquina) e ambas foram simuladas de forma correta, além de que durante o trabalho foi demonstrado formas de efetuar conversão indiretamente entre todas as máquinas, validando assim que uma máquina universal pode simular outra máquina universal.

É importante analisarmos que a simulação/conversão de modelo não deve ser vista como uma comparação de desempenho, pelo fato de que além da resolução do problema é simulado o comportamento da máquina sendo que esse impacta na criação de rotinas que não seriam desenvolvidas caso fosse criado uma máquina única para a resolução do problema, o que deve ser analisado é que elas possuem o mesmo poder computacional.

Durante o desenvolvimento deve ser ressaltado que existiu um grande dificuldade em encontrar maneiras de efetuar as conversões/simulação, pelo fato de que mesmo de posse dos conceitos e informações de como deveria ser efetuada, não foi encontrado exemplo ou códigos que efetuassem as rotinas, sendo então essas criadas com base nos conceitos e informações levantas na fundamentação.

Dentro as possibilidades futuras, está a utilização da ferramenta em ambiente educacional a fim de mediar o impacto causado em alunos ao serem apresentados a esses conceitos. Aprimoração das técnicas de simulação/conversão utilizadas, ampliação dos componentes das máquina, exemplo, adicionar mais fitas a Máquina de Turing. Utilização do protótipo no auxílio de problemas que utilizam

máquinas universais como soluções, buscando auxiliar e validas a efetividade do mesmo.

REFERÊNCIAS

ALVES, Débora Pandolfi. **Equivalência de Máquinas Universais:** Demonstração, Análise e Simulação. 2007. 199 f. Monografia (Especialização) - Curso de Ciência da Computação, Universidade do Vale do Rio dos Sinos, São Leopoldo, 2007.

AUTÔMATO com Duas Pilhas. 2003. Disponível em: <https://www.yumpu.com/pt/document/view/12987052/automato-com-duas-pilhas-podre>. Acesso em: 28 nov. 2017.

CINTRA, Caroline Carbonell; BRONFMAN, Paula Nemetz; DIVERIO, Tiaraju Asmuz. **A importância da teoria da computação na ciência da computação**. Salão de Iniciação Científica. Livro de resumos. Porto Alegre: UFRGS, 1996.

DE OLIVEIRA, Gerson Pastre; DA SILVA, Marco Aurélio Seraphim. Construção de Simuladores Gráficos para Teoria da Computação: Uma Proposta para o Ensino do Conceito de Máquinas de Turing. **IV Simpósio de Excelência em Gestão e Tecnologia (SEGET)**, 2007. Disponível em: <http://www.cpge.aedb.br/seget/artigos07/1038_1038_Artigo%20turing.pdf>. Acesso em: 28 jun. 2018.

DIVERIO, Tiarajú Asmuz; MENEZES, Paulo Blauth. **Teoria da Computação:** Máquinas Universais e Computabilidade. 3. ed. Porto Alegre: Bookman, 2011. 288 p. (Série Livros Didáticos Informática UFRGS).

_____. **Teoria da Computação:** Máquinas Universais e Computabilidade. 2. ed. Porto Alegre: Sagraluzzatto, 2000. 224 p. (Série Livros Didáticos Informática UFRGS).

EXATAS. **A Máquina de Post.** 2015. Elaborada por: Estudando Ciências da Computação. Disponível em: <http://deexatas.blogspot.com.br/2015/07/a-maquina-de-post.html>. Acesso em: 27 nov. 2017.

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

_____. **Introduction to Automata Theory, Languages, and Computation.** 2. ed. Boston: Addison Wesley, 2001.

LEAL, Liara Aparecida dos Santos. **Uma Fundamentação Teórica para a Complexidade Estrutural de Problemas de Otimização**. 2002. 116 f. Tese (Doutorado) - Curso de Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

LEWIS, Harry R.; PAPADIMITRIOU, Christos H.. **Elementos de Teoria da Computação.** 2. ed. Eua: Bookman, 2004. 340 p.

MENEZES, Paulo Blauth. **Linguagens formais e autômatos.** 5. ed. Porto Alegre: Bookman, 2005. 215 p.

_____. **Linguagens Formais e Autômatos.** 3. ed. Porto Alegre: Sagra Luzzatto, 2000. 165 p.

PINHEIRO, Thielyon et al. Teoria da Computação: Máquinas, Programas e suas Equivalências. I **Seminário de Pesquisa Científica e Tecnológica**, v. 1, n. 1, 2017.

PY, Mônica Xavier. **Análise da Máquina de Turing Persistente com Múltiplas Fitas de Trabalho.** 2003. 74 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2003.

SILVA, Flávio Soares Corrêa da; MELO, Ana Cristina Vieira de. **Modelos Clássicos de Computação.** São Paulo: Thomson Learning, 2006. 67 p.

SIMEONE, Fernando. **Hierarquia de Chomsky.** 2015. Disponível em: <https://pt.slideshare.net/fernandosimeone/hierarquia-de-chomsky>. Acesso em: 27 nov. 2017.

SIPSER, Michael. **Introduction to the Theory of Computation.** 3. ed. Boston: Cengage Learning, 2013. 482 p.

_____. **Introdução À Teoria Da Computação.** 2. ed. São Paulo: Cengage Learning, 2007. 488 p.

TORRES, Delfim Fernando Marado. O Computador Matemático de Post. **Boletim da Sociedade Portuguesa de Matemática,** [s. L.], v. 46, p.81-94, abr. 2002.

VIEIRA, Luiz Filipe Menezes; VIEIRA, Marcos Augusto Menezes; VIEIRA, Newton José. Language Emulator, uma ferramenta de auxílio no ensino de Teoria da Computação. In: **XIII Workshop sobre Educação em Computação–XXV Congresso da Sociedade Brasileira de Computação**. 2003.

VIEIRA, Newton José. **Introdução aos Fundamentos da Computação:** Linguagens e Máquinas. São Paulo: Pioneira Thomson, 2006. 334p.

APÊNDICE(S)

APÊNDICE A - ARTIGO CIENTÍFICO

Máquinas Universais: Demonstração, Ánalise e Equivalência

Leandro Justin Vieira[1]

[1]Curso de Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma, SC – Brazil

Leandrojv83@gmail.com

Abstract. *Computer science is based in computational models built in the first half of the 20th century. In 1963, the Turing Machine was published, a model known and accepted as the formalization of an algorithm, that which belongs to a class known as the universal machines, possessing the greatest computational power to this day. The computation theory has an important role in the construction of knowledge and provides development of logic and formal thinking, that being more and more necessary for computation. Four models of universal machines are presented (Turing's Machine, Post's Machine, Two-Cell Automaton and Norma's Machine), those which are contextualized and from that, simulators were developed, other than also demonstrating how a model can be simulated by another universal machine, in order to prove Church's Thesis. At the end of the research, a web prototype was developed, responsible for the simulators and that effectuates a conversion from Post's Machine to Turing's Machine and from the Two-Cell Automaton also to Turing's Machine.*

Resumo. *A ciência da computação é fundamentada em modelos computacionais construídos na primeira metade do século XX, em 1936 foi publicado a Máquina de Turing modelo conhecido e aceito como a formalização de um algoritmo, esse que pertence à uma classe chamada de máquinas universais, detentora do maior poder computacional até hoje. A teoria da computação tem um papel importante na construção do conhecimento e proporciona desenvolvimento do raciocínio logico e formal, sendo que esse é cada vez mais necessário para a computação. É apresentado quatro modelos de máquinas universais (Máquina de Turing, Máquina de Post, Autômato de duas pilhas e Máquina de Norma), esses que são contextualizados e a partir disso desenvolvido simuladores para cada, além de demostrar também como um modelo pode ser simulado por outra máquina universal, afim de comprovar a Tese de Church. Ao final da pesquisa, foi desenvolvido um protótipo web responsável pelos simuladores e que efetua a conversão de Máquina de Post para Máquina de Turing e Autômato de Duas Pilhas para Máquina de Turing.*

1. Introdução

A ciência da computação é fundamentada em modelos computacionais que foram construídos na primeira metade do século XX, entretanto sua origem é milenar, sendo desenvolvida em diversas regiões e ao longo da história. A ciência da computação atual possui duas ênfases, a teórica baseada em fundamentos e modelos computacionais, e a ênfase prática que aplica a teoria na construção de projetos de sistemas computacionais. É importante ressaltar que a ênfase teórica não depende de instrumentos computacionais ou máquinas tecnologias, ou seja,

não necessita de computadores para ser formulada (hardwares e softwares) (DIVERIO; MENEZES, 2011).

Como dito os modelos utilizados atualmente foram construídos na primeira metade do século XX, aparentemente nesse momento histórico específico sentiu-se uma grande necessidade de caracterizar o que significaria ser computável, definir um modelo de computação suficientemente genérico para especificar qualquer função computável. É possível notar que existiu essa necessidade pois diversos pesquisadores trabalharam simultaneamente nesse problema, entretanto, mesmo os modelos sendo construídos independentemente, eles se mostraram matematicamente equivalentes, ou seja, parecia já existir de fato um conceito característicos de computabilidade, que podia ser formulado de diversas maneiras, contudo cada modelo nasceu de um contexto de pesquisa distinto, sendo portanto único, tendo papéis diferentes na evolução da ciência da computação, e cada um evidência com maior clareza aspectos específicos da noção da computabilidade (SILVA; MELO, 2006).

Entre os modelos desenvolvidos podemos citar o Cálculo Lambda (Alonzo Church, 1936), funções Recursivas (Stephen Cole Kleene, 1936), Máquina de Turing (Alan Turing, 1936), Máquina Norma (Richard Bird, 1976), Sistema Canônico de Post (Emil Leon Post, 1936), entre outros.

Entretanto existe uma dificuldade no que diz respeito ao compreendimento desses conteúdos, pois os mesmos possuem uma complexidade maior devido que para seu entendimento é necessário compreender modelos matemáticos, diferente de outras disciplinas e áreas da computação (VIEIRA; VIEIRA; VIEIRA, 2003).

2. Justificativa

A teoria da computação caracteriza-se pelo estudo formal dos processos de computação, suas capacidades e suas limitações (CINTRA; BRONFMAN; DIVERIO, 1996), sendo de grande importância para a ciência da computação, pois fundamenta e conceitua o embasamento teórico para um correto e amplo entendimento da ciência envolvida na computação, além de proporcionar um desenvolvimento do raciocínio logico e formal, sendo que esse é cada vez mais necessário para a computação (DIVERIO; MENEZES, 2000). Além de que diferentes modelos de computação, acabam se mostrando mais adequados para explicar diferente paradigmas de programação, portanto, é importante que se tenha o conhecimento de diversos modelos (SILVA; MELO, 2006).

Ou seja, todos esses fundamentos estão ligados com o que os cientistas da computação desenvolvem hoje, por exemplo, os conceitos da máquina de Turing podem auxiliar na percepção de como se dá o fluxo do software, conhecer e entender teorias como a de problemas intratáveis também é de suma importância, pois permite ao acadêmico que se deparar com um problema, conseguir identificar se é possível desenvolver uma solução para o mesmo, pois esse pode pertencer a uma classe de problemas intratáveis, sendo então necessário descobrir um modo diferente do habitual para contornar o problema (HOPCROFT; ULLMAN; MOTWANI, 2002).

3. Desenvolvimento

O trabalho desenvolvido tem como objetivo desenvolver uma aplicação que simule e demonstre as máquinas universais e suas equivalências, para isso é necessário compreender as máquinas universais a serem simuladas sendo essas Máquina de Turing, Máquina de Post, Máquina de Norma e Autômato de Duas Pilhas, além de entender conceitos e teorias como a tese de Church, computabilidade, problema de parada, problema de decisão, além de encontrar a equivalência entre as máquinas universais já citadas.

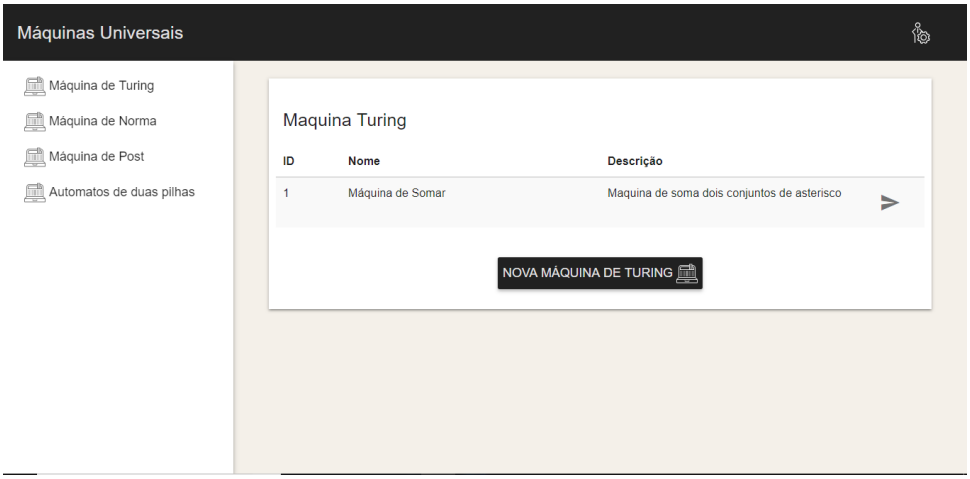
4. Metodologia

O projeto foi desenvolvido utilizando Play Framework para Java e Angular JS, ambas foram escolhidas pelo fato de serem ferramentas para o desenvolvimento web, com uma curva de aprendizado baixo, robustas e com alto nível de produtividade, sendo o Play Framework responsável pelo back-end da aplicação e o Angular JS pelo front-end. A comunicação entre as duas ferramentas de desenvolvimento acontece via REST. (PIERIN, 2013).

O projeto foi dividido em quatro seções principais, sendo cada seção uma máquina universal, para a máquina de Turing e Norma foi desenvolvido telas e formulários responsáveis por cadastrar as informações de seus componentes e transições com o intuído de simular a execução da máquina, para a máquina de Post e Autômato de Duas Pilhas além das telas de cadastro de informação e execução foi validado a conversão dessas para a máquina de Turing, validando assim a teoria de Church sobre as máquinas universais.

4.1 modelagem e execução das máquinas universais

Durante a fundamentação do trabalho foram apresentados os componentes de cada máquina universal e sua execução, ou seja, como cada elemento/componente funcionava a partir de uma palavra inicial gerando uma saída após as regras/transições definidas. Devido a isso o protótipo apresenta uma interface onde é possível selecionar a máquina universal e através dela criar modelos para as máquinas trabalhas no protótipo, definido para ela, nome, descrição e seus componentes.



2 Figura 1 – Tela inicial e diálogo de criação de Máquina Turing

4.2 Máquina de Turing

Para o desenvolvimento da Máquina de Turing foram criados as entidades, símbolos (para representar o alfabeto da máquina de Turing), estados (podendo ser do tipo normal, final e inicial) e transições (define como será o fluxo da MT, a partir de um estado e um símbolo lido na fita), sendo que as informações adicionais da máquina como nome e sua descrição é atributo da entidade maquina universal, o tipo desta máquina é informado como Máquina de Turing.

Cada máquina universal é dívida em seções, a seção geral de uma MT onde é localizado os formulários de seus componentes, sendo apresentados eles em listas, com a possibilidade de edição e adição de novos componentes (símbolos, estados e transições), tanto a edição como adição são efetuadas em modais, formulários com os campos necessários para que o componente funcione adequadamente, a próxima é a seção de execução, nela é informada uma palavra, caso essa seja aceita na máquina será gerado uma tabela com o histórico da

execução da máquina, e um ferramenta para que seja possível verificar o funcionamento da mesma. Para as palavras que não forem aceitas será sinalizado através de uma mensagem qual o erro que aconteceu na execução.

A execução da máquina de Turing inicia configurando as dados da máquina, ou seja, acola em uma variável responsável por representar a fita a palavra informada, seta a posição atual do cabeçote como zero, além de buscar todos os símbolos, estados e transições existentes na máquina. Após isso é buscado pelo estado inicial da máquina, esse que é alocado em uma variável chamada estado atual, a variável estado atual juntamente com a posição atual, são responsáveis pela unidade de controle da MT, enquanto o estado atual não for final a máquina busca na lista de transição, uma transição do estado atual com leitura para o símbolo encontrado na fita na posição atual do cabeçote, caso a transição seja encontrada, essa informa o próximo estado atual, escreve na fita o símbolo informado na posição do cabeçote, além de que possuir a direção do cabeçote, caso a direção seja direita será acrescentado 1 a posição atual, caso for esquerda será decrescendo 1 da posição atual. Se não for encontrado a transição ocorrerá um indeterminismo na máquina, encerrando a execução da mesma.

Ao final da execução caso esse seja completado com sucesso é informado para o usuário do protótipo os passos que a MT vez até chegar no resultado final, caso aconteça algum indeterminismo, será informado qual símbolo e qual estado gerou o mesmo, demais erros também são lançados para o usuário.

4.3 Máquina de Norma

De forma semelhante a Máquina de Turing foi criado entidades relacionais para suprir as necessidades e simbolizar os componentes da Máquina de Norma, sendo esses, símbolos (para representar o alfabeto da máquina), estados (representando os rótulos, podendo ser do tipo normal, final e inicial), transições (instruções que definem como será o fluxo da Máquina de Norma, a partir de um estados (rótulo)) e registradores (responsáveis por armazenar os valores), sendo que as informações adicionais como nome e sua descrição é atributo da entidade maquina universal, o tipo desta máquina é Máquina de Norma.

Foi mapeado para uma transição/instrução Norma as funções de adição, subtração, atribuição de valor e teste, sendo q um estado pode possuir somente um único tipo de instrução e somente é aceito que um estado tenha mais de uma transição se suas transições forem do tipo teste. Para os registradores esses podem armazenar um único valor ou um conjunto de valores (registradores do tipo array), os elementos armazenados em um registrador são os pertencentes em seu alfabeto ou pertencentes ao conjunto dos números naturais.

Na seção de execução da máquina de norma, caso a máquina chegue a um estado final é retornado o histórico de execução da máquina, mostrando todos as instruções que foram executas na máquina e os registradores que foram modificados, além de que é possível na ferramenta que executa o histórico da máquina visualizar os valores de cada registrador no momento de execução de cada instrução.

O algoritmo responsável pela execução da Máquina de Norma, inicialmente verifica se todas as condições para a execução são satisfeitas, sendo essas, a existência de um registrador X, um único estado inicial, se o estado inicial possuí uma transição, após isso é executado as instruções, alterando os registradores alvos caso a função seja do tipo de atribuição de valor, subtração e adição, e efetuando testes na máquina para a tomada de decisão de qual a instrução a mesma deve ir, o algoritmo executa até que seja encontrado um estado/rotulo final ou aconteça um indeterminismo, sendo esse apresentado em formato de erro.

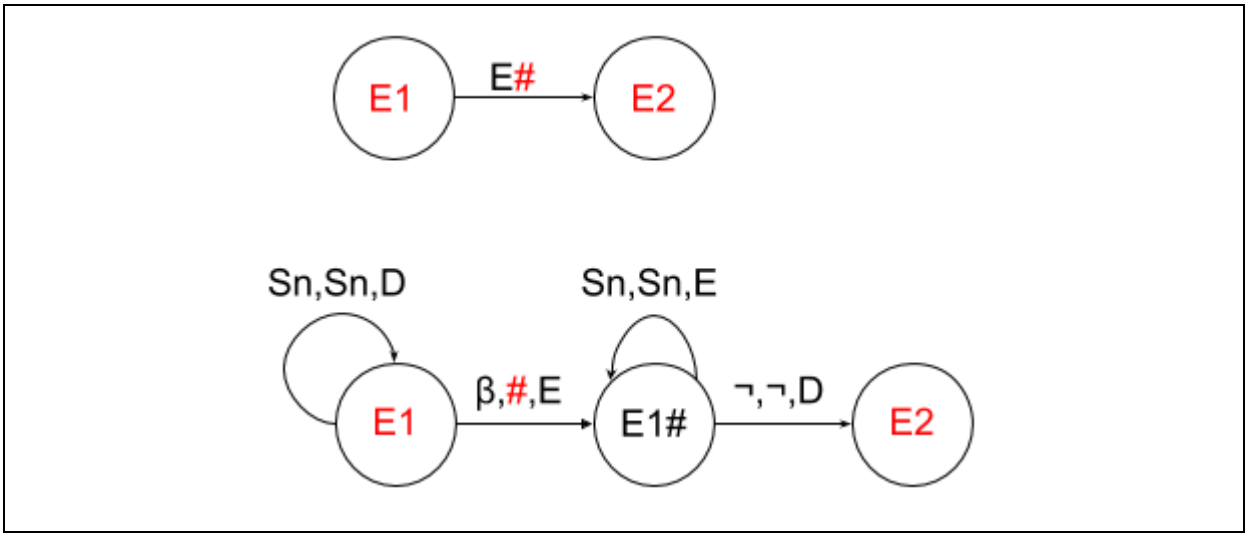
4.4 Máquina de Post

A máquina de Post foi desenvolvida em três seções, geral, executar e converter, as seções, geral e executar são similares a da Máquina de Turing e Máquina de Norma, entretanto adaptado para os componentes da Máquina de Post, sendo esses estados (podem ser partida/inicial, normal, aceita/final, rejeita/final), símbolos e transições (podem ser escrita e leitura/teste).

A execução da máquina de Post segue os princípios de uma fila (FIFO), como citado acima as transições podem ser, escrita e leitura/teste, o algoritmo que executa a máquina previamente cadastrada na seção geral, busca a transição do estado inicial, caso seja encontrada a máquina irá executar removendo o primeiro elemento da palavra e realocando os demais para transição de leitura/teste, e adicionando símbolos pertencentes ao alfabeto da máquina ao final da palavra quando a transição por de escrita, até o momento em que essa encontra um estado final (aceitação/rejeição). De forma semelhante a máquina de norma, os estados devem possuir somente um único tipo de transição, sendo que somente estados que possuem transição de leitura/teste podem possuir mais de uma transição, isso se dá ao fato que a máquina pode seguir diferentes caminhos de acordo com o símbolo lido na fila.

4.4.1 Conversão Máquina de Post ⇾ Máquina de Turing

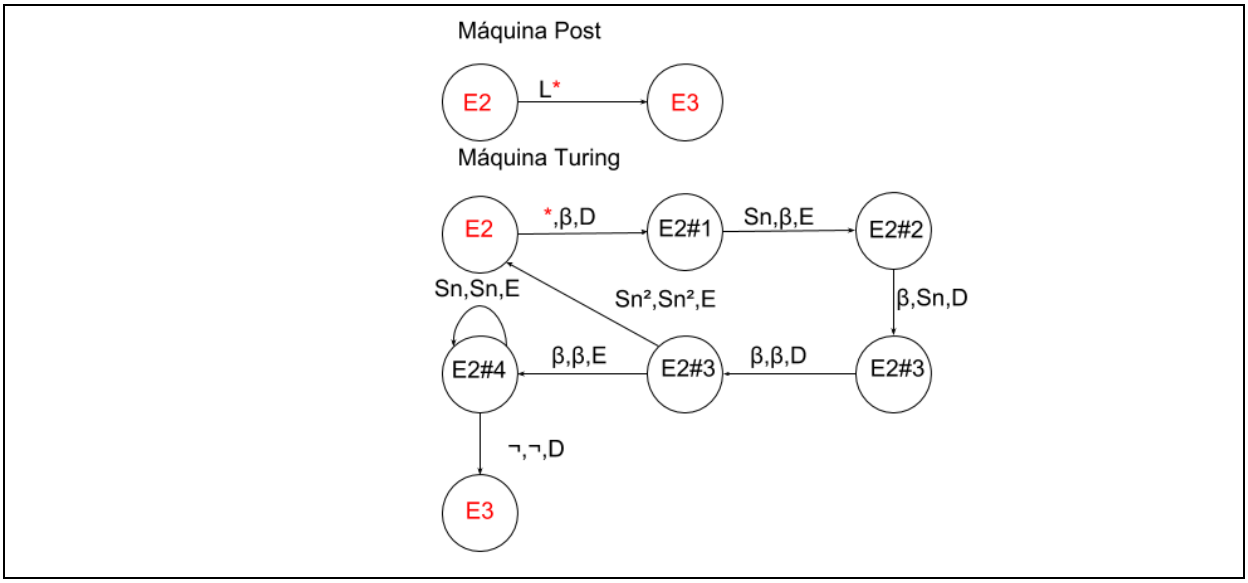
Para validar a Tese de Church previamente fundamentada, foi desenvolvido a conversão para as máquinas de Post, essa baseia-se em desenvolver estruturas que comportem cada transição de Post em Turing, efetuando o funcionamento da máquina em sua máquina alvo, ou seja, além da MT chegar ao mesmo resultado que a MP, essa deve simular o funcionamento em fila da MP.



3 Figura 2 – Conversão de função de escrita MP ⇾ MT

Na figura 2 são demonstrados os procedimentos para a conversão de uma transição de escrita desenvolvida em uma Máquina de Post para uma Máquina de Turing, o primeiro grafo representa a transição em Post e o segundo em Turing. Como citado anteriormente é necessário que a máquina alvo além de produzir o mesmo resultado simule a execução da máquina a ser simulada (convertida), os dados na imagem em vermelho representam os símbolos que geram a escrita na máquina, ou seja, E1 é o estado inicial que escreve # no final da fita (MT) ou fila (MP) indo para E2, as demais transições existentes na transição da máquina de Turing são para simular o funcionamento de fila de Post, sendo Sn usado para representar que deve ter uma transição para todos os símbolos do alfabeto, ou seja, para que a máquina de Turing simule uma ação de escrita em Post ela deve percorrer toda a fita até

encontrar um símbolo vazio (β) onde irá escrever na fita o símbolo indicado na transição de escrita, após isso deve voltar até encontrar o símbolo inicial da MT (representando por ¬), endereçando assim o próximo estado, com o cabeçote localizado no primeiro elemento da palavra após o símbolo inicial da máquina.



4 Figura 3 – Conversão de função de leitura teste MP → MT

Na figura 3 são demonstrados os procedimentos para a conversão de uma transição do tipo leitura/teste, o primeiro grafo representa uma transição do tipo leitura/teste em uma MP já o segundo grafo representa a mesma transição simulando o comportamento da MP em uma MT. A transição a ser simula efetua o seguinte procedimento, no estado E2 caso o símbolo do topo da pilha seja * retira-o do topo e vai para o estado E3, os símbolos e estados envolvidos estão demarcados de vermelho, os demais estados e transições são auxiliares para efetuar a simulação da execução da MP pela MT.

Para a MT simular a execução de uma transição citada acima, essa deve remover o símbolo lido, avançar para a direita da fita remover novamente o símbolo e escrevê-lo na posição anterior, até encontrar um símbolo vazio a direita, afim de realocar todos os símbolos existentes na fita (para simular o comportamento de fila), após isso deve voltar até encontrar o símbolo inicial da MT (representando por ¬), endereçando assim o próximo estado, com o cabeçote localizado no primeiro elemento da palavra após o símbolo inicial da máquina.

Na conversão da MP para MT além de cada transição passar pelos processos descritos é adicionado no alfabeto de símbolos da máquina o símbolo de vazio caso essa não possua (representando na aplicação pelo símbolo §), e o símbolo inicial da MT (representado na aplicação pelo símbolo ¬). O tipo de estado é mantido na conversão, entretanto um estado pode possuir estados auxiliares como foi descrito acima, esses que são estados do tipo normal.

7.5 Autômato de Duas Pilhas

Da mesma forma que a Máquina de Post foi desenvolvida o Autômato de Duas Pilhas com as seções gerais, executar e converter, afim de validar a teoria de Church, efetuando a simulação da máquina universal Autômato de Duas Pilhas em uma Máquina de Turing. Seguindo o mesmo princípio que as demais máquinas, na seção geral é fundamentado os componentes da máquina, afim de fornecer ferramentas para criar um modelo de um autômato de duas pilhas e posteriormente ser executado na seção de mesmo nome a partir de uma palavra definida pelo

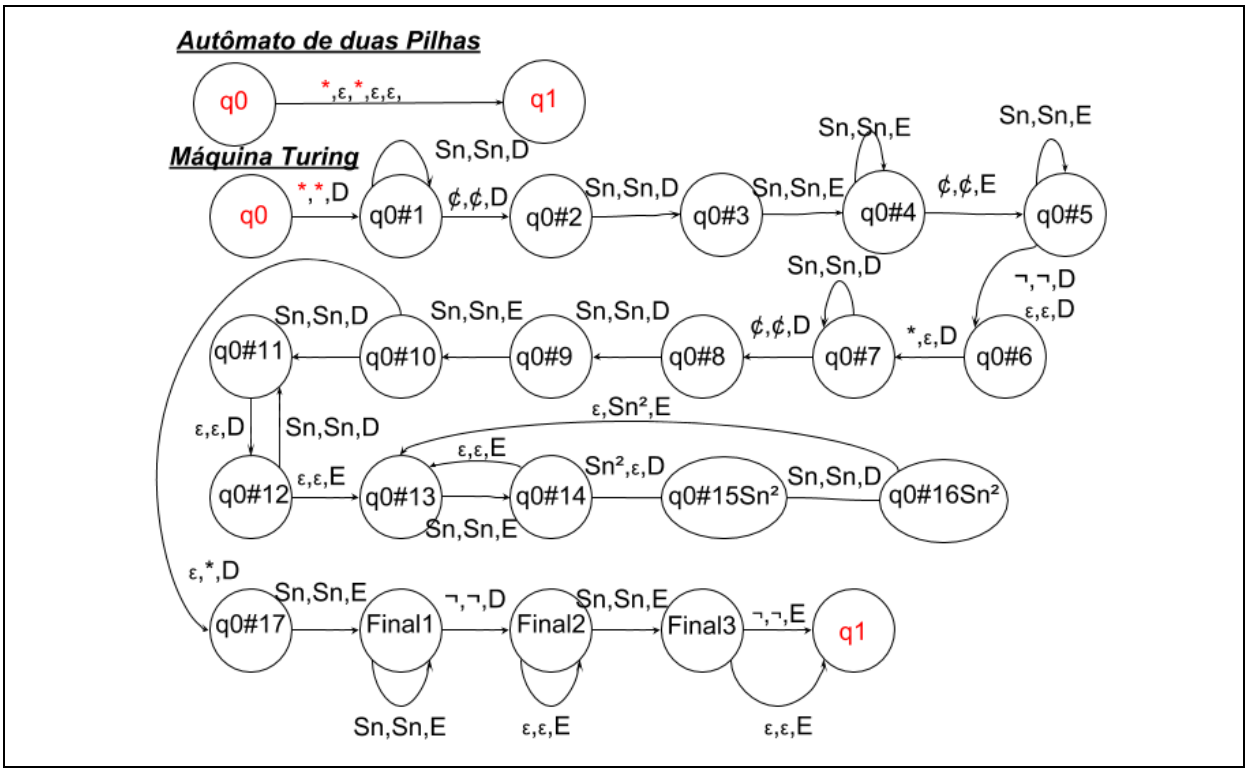
usuário, demostrando os resultados e passos que a máquina passou caso encontre um final em sua execução, ou mostrando o erro ocorrido durante o processo.

Os componentes criados foram estados (normal, final e inicial), símbolos (alfabeto da máquina) e transições (responsável por definir o fluxo da máquina). A transição desenvolvida pode ser dividia em duas etapa, ações de entrada/teste e de saída, sendo que essas podem ser, retirar símbolo especifico (ação de entrada), retirar qualquer símbolo (ação de entrada), não efetua leitura (ação de entrada), empilha símbolo especifico (ação de saída) e não empilha símbolo (ação de saída). Existe três ações de entrada e duas ações de saída.

Após a definição da máquina foi desenvolvido a execução da máquina, lembrando que a estrutura do Autômato de Duas Pilhas baseia-se em pilhas responsáveis por armazenar as informações da máquina, podendo exercer tanto leitura como escritas na mesma, caso a leitura seja executada o símbolo lido é destruído, ou seja, removido da pilha, a palavra a ser executada na máquina é armazena em uma fita, a leitura da fita é destrutiva como na pilha entretanto não é possível escrever na fita, sendo então utilizada somente para a tomada de decisão de qual transição deve ser utilizada estando em um determinado estado, lembrando que a decisão é tomada após a leitura de um símbolo especifico, qualquer símbolo ou não leitura da fita e das duas pilhas, podendo a partir da decisão ser empilhado um símbolo ou não nas duas pilhas. Visando isso foi desenvolvido um algoritmo que a partir do estado inicial da máquina encontra uma transição que satisfaça a situação dos componentes da máquina, passando entre os estados modificando as pilhas até chegar em um estado final ou gerar um não determinismo ou erro de execução.

7.5.1 Conversão Máquina de Post ⇾ Máquina de Turing

Além da conversão de Post para Turing, foi desenvolvida a conversão de Autômato de Duas para Máquina de Turing, com o mesmo intuito de validar a tese de Church, como foi visto na conversão de Post, é necessário que a máquina simule tanto seus componentes como seu comportamento gerando ao final o mesmo resultado. Entretanto diferente da Máquina de Post, que possui os mesmo componentes da Máquina de Turing (fita, estado e Símbolo) diferenciando somente seu comportamento, autômatos de duas pilhas possui tanto um comportamento diferente como seus componentes também, trazendo assim uma complexidade maior para a simulação, como foi visto durante a fundamentação teórica para uma Máquina de Turing simular uma Máquina Universal de Autômatos de Duas Pilhas, a fita da máquina será responsável por representar tanto a palavra informada, como as duas pilhas, para que isso ocorra é adicionado ao alfabeto da máquina um símbolo auxiliar ao final da palavra essa alocada na fita, símbolo esse representando na aplicação por ¢, esse é responsável por demarcar o final da palavra e o início das pilhas, a separação das pilhas é feita de uma forma que a primeira pilha corresponde às células ímpares da fita após o símbolo de auxiliar, e a segunda pilha os símbolos pares após o símbolo auxiliar.



5 Figura 4 – Conversão de Autômatos de duas pilhas para Máquina Turing

Na figura 4 é possível encontrar um trecho da simulação de uma transição de Autômato de duas pilhas em uma máquina de Turing, essa tem o papel de ao ler o símbolo "*" independente dos dados alocados nas pilhas, empilhas "*" na primeira pilha. Para simular isso foi desenvolvido para a máquina de Turing uma serie de estados para efetuar as funções existentes nas transições, lembrando que como é possível as transições executarem diferentes ações, teve-se que desenvolver algo genérico e que se adapte à realidade de cada transição, na figura 4, dos estados q0#1 a q0#4, a máquina verifica se a máquina atende as configurações da transição, q0#4 a q0#6 é buscado o primeiro símbolo existe na fita, q0#6 a q0#7 remove o símbolo da fita, q0#7 a q0#8 é encontrada a primeira fila, q0#8 a q0#9 não remove símbolo da pilha, q0#9 a q0#10 não remove símbolo da segunda pilha, q0#10 a q0#17 escreve símbolo na primeira pilha, sendo que se a primeira posição já estiver sendo ocupada será realocado os demais e o símbolo indicado na transição será alocado na primeira posição, final1 a final3 encontra o início para o próximo estado da transição. Além de que algumas funções mapeadas não foram exemplificadas pela figura 4, como o controle e escrita da segunda pilha e a remoções de símbolos da pilha.

Na conversão da máquina de Autômatos finitos para MT além de cada transição passar pelos processos descritos é adicionado no alfabeto de símbolos da máquina o símbolo de vazio caso essa não possua (representando na aplicação pelo símbolo §), símbolo inicial da MT (representado na aplicação pelo símbolo ¬) e o símbolo auxiliar que marca o final da palavra e o início das pilhas (representado na aplicação pelo símbolo ¢). O tipo de estado é mantido na conversão, entretanto um estado pode possuir estados auxiliares como foi descrito acima, esses que são estados do tipo normal.

4. Conclusão

Esse trabalho teve como objetivo apresentar conceitos e definições existentes na área da teoria da computação relacionados as máquina universais e validar através de simulação/conversão a tese de Church, desenvolvendo quatro máquinas universais (Máquina de Turing, Máquina de

Post, Máquina de Norma e Autômato de Duas Pilhas) e efetuando conversões de Post para Turing e Autômato de Duas Pilha para Turing.

O principal questionamento/objetivo foi atendido ao ser desenvolvido um protótipo responsável por apresentar componentes responsáveis pela criação das máquina e as conversões/simulações efetuadas.

A validação da tese de Church foi devidamente realizada através simulação da Máquina de Post e Autômato de Duas Pilhas, pelo fato que a primeira máquina possui um comportamento diferente da Máquina de Turing e a segunda possui uma maior diversidade de componentes (estrutura da máquina) e ambas foram simuladas de forma correta, além de que durante o trabalho foi demonstrado formas de efetuar conversão indiretamente entre todas as máquinas, validando assim que uma máquina universal pode simular outra máquina universal.

É importante analisarmos que a simulação/conversão de modelo não deve ser vista como uma comparação de desempenho, pelo fato de que além da resolução do problema é simulado o comportamento da máquina sendo que esse impacta na criação de rotinas que não seriam desenvolvidas caso fosse criado uma máquina única para a resolução do problema, o que deve ser analisado é que elas possuem o mesmo poder computacional.

Durante o desenvolvimento deve ser ressaltado que existiu um grande dificuldade em encontrar maneiras de efetuar as conversões/simulação, pelo fato de que mesmo de posse dos conceitos e informações de como deveria ser efetuada, não foi encontrado exemplo ou códigos que efetuassem as rotinas, sendo então essas criadas com base nos conceitos e informações levantas na fundamentação.

Dentro as possibilidades futuras, está a utilização da ferramenta em ambiente educacional a fim de mediar o impacto causado em alunos ao serem apresentados a esses conceitos. Aprimoração das técnicas de simulação/conversão utilizadas, ampliação dos componentes das máquina, exemplo, adicionar mais fitas a Máquina de Turing. Utilização do protótipo no auxílio de problemas que utilizam máquinas universais como soluções, buscando auxiliar e validas a efetividade do mesmo.

Referências

DIVERIO, Tiarajú Asmuz; MENEZES, Paulo Blauth. **Teoria da Computação:** Máquinas Universais e Computabilidade. 3. ed. Porto Alegre: Bookman, 2011. 288 p. (Série Livros Didáticos Informática UFRGS).

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

MENEZES, Paulo Blauth. **Linguagens formais e autômatos.** 5. ed. Porto Alegre: Bookman, 2005. 215 p.

SILVA, Flávio Soares Corrêa da; MELO, Ana Cristina Vieira de. **Modelos Clássicos de Computação.** São Paulo: Thomson Learning, 2006. 67 p.

VIEIRA, Luiz Filipe Menezes; VIEIRA, Marcos Augusto Menezes; VIEIRA, Newton José. Language Emulator, uma ferramenta de auxílio no ensino de Teoria da Computação. In: **XIII Workshop sobre Educação em Computação–XXV Congresso da Sociedade Brasileira de Computação**. 2003.