UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS VII PATOS - PB
CENTRO DE CIÊNCIAS EXATAS E SOCIAIS APLICADAS - CCEA
CURSO DE BACHARELADO EM COMPUTAÇÃO

THALES MYLLER DE OLIVEIRA ALMEIDA

A GENERALIZAÇÃO DAS MÁQUINAS DE TURING POR MEIO DOS GRAFOS CAYLEY NA SOLUÇÃO DE PROBLEMAS NÃO RECURSIVAMENTE ENUMERÁVEIS

PATOS
2017

THALES MYLLER DE OLIVEIRA ALMEIDA

A GENERALIZAÇÃO DAS MÁQUINAS DE TURING POR MEIO DOS GRAFOS CAYLEY NA SOLUÇÃO DE PROBLEMAS NÃO RECURSIVAMENTE ENUMERÁVEIS

Trabalho de Conclusão de Curso apresentado ao Curso de Computação, da Universidade Estadual da Paraíba como requisito parcial à obtenção do título de Bacharelado em Computação.

Área de concentração: Ciência da Computação

Orientador: Prof. Dr. Rodrigo Alves Costa

PATOS

2017

A447g

Almeida, Thales Myller de Oliveira.

A generalização das Máquinas de Turing por meio dos Grafos Cayley na solução de problemas não recursivamente enumeráveis [manuscrito] / Thales Myller de Oliveira Almeida. - 2017.

32 p.

Digitado. Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências Exatas e Sociais Aplicadas, 2024. "Orientação : Prof. Dr. Rodrigo Alves Costa, Coordenação do Curso de Computação - CCEA. "

1. Grafos Cayley. 2. Problema indecidível. 3. Máquina de Turing. I. Título

21 ed. CDD 004

Elaborada por Esdras R. F. Dantas - CRB - 15/670 BSC7/UEPB

Thales Myller de Oliveira Almeida

A GENERALIZAÇÃO DAS MÁQUINAS DE TURING POR MEIO DE GRAFOS CAYLEY NA SOLUÇÃO DE PROBLEMAS NÃO RECURSIVAMENTE ENUMERÁVEIS

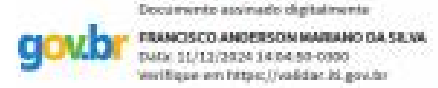
Trabalho de Conclusão de Curso apresentado ao Curso de Computação da Universidade Estadual da Paraíba, em cumprimento à exigência para obtenção do grau de Licenciado em Ciência da Computação

Aprovado em 05 de dezembro de 2017

BANCA EXAMINADORA

Prof. Dr. Rodrigo Alves Costa
(Orientador)

Prof. Dr. Wellington Candeia de Araujo
(Examinador)



Prof. Esp. Francisco Anderson Mariano da Silva
(Examinador)

RESUMO

Muito antes de os computadores digitais serem desenvolvidos, Alan M. Turing propôs uma máquina com o mesmo poder computacional das máquinas atuais: as máquinas de Turing, capazes de representar qualquer função computável através de fitas unidimensionais. Apesar de toda eficiência apresentada, as máquinas tradicionais não conseguiam computar alguns problemas, que na verdade não podem ser computados por nenhum algoritmo existente, os problemas indecidíveis. Dado o poder restritivo das máquinas tradicionais, novos modelos foram gradativamente propostos, até que surgiu o interesse com geometria de fitas alternativas. Também surgiram máquinas modeladas com grafos bidimensionais, os chamados grafos de Cayley, que, associados com as fitas alternativas, podem contribuir para um enorme avanço no poder computacional. Esse novo modelo apresentou um avanço relevante na computação de problemas indecidíveis, visto que, as máquinas de Turing modelados através de grafos de Cayley se mostram estritamente mais poderosas do que as máquinas tradicionais.

Palavras-chave: Máquinas de Turing; Problemas indecidíveis; Grafos de Cayley.

ABSTRACT

Long before digital computers are developed, Alan M. Turing proposed a machine with the same computational power of current machines, Turing machines, a model able to represent any computable function. This model consisted of one-dimensional tapes. However, despite all presented efficiency, traditional machines can not compute some problems that call undecidable. Undecidable problems are problems that can not be computed by any existing algorithm. So, since the restraining power of traditional machines, new models have been proposed. Then came the geometry of interest with backup tapes, which improved the outcome of these machines. It also emerged machines modeled with two-dimensional graphs, Cayley graphs considered, representing a major advance in its computational power. This new model showed significant advance in computing problems undecidable, since Turing machines with Cayley graphs show is strictly more powerful than traditional machines.

Keywords: Turing machines; Undecidable problems; Cayley's graphs.

SUMÁRIO

1. INTRODUÇÃO

No inicio do século XX diversas pesquisas foram realizadas a fim de definir um modelo computacional genérico, capaz de programar qualquer função computável (MENEZES, 2008).

Em 1936, foi proposto pelo matemático britânico Alan M. Turing um modelo conhecido, em sua homenagem, como Máquina de Turing, que posteriormente se mostrou muito avançado em termos científicos para a época, tendo em vista a sua capacidade computacional e a semelhança com os computadores eletrônicos.

Atualmente, a máquina de Turing é aceita como uma formalização do conceito de procedimento, isto é, uma sequência finita de instruções que podem ser realizadas mecanicamente em um tempo finito.

De acordo com Palazzo (2007), em 1936, Alonzo Church apresentou uma hipótese que ficou conhecida como a Hipótese de Church, segundo a qual qualquer função computável pode ser representada por uma Máquina de Turing.

Como o conceito de procedimento é matematicamente impreciso, não seria possível provar que a Máquina de Turing é realmente o dispositivo computacional mais genérico possível, na qual ao se provar, teríamos o limite máximo do que pode ser atingido por qualquer dispositivo de computação. Todavia, nenhum modelo computacional desenvolvido até o presente conseguiu apresentar capacidade computacional superior ao modelo de Turing, o que reforça a Hipótese de Church. Em resumo, mesmo sem se poder considerar a máquina de Turing o mais genérico dispositivo de computação, o fato é que nenhum outro modelo, até os dias de hoje, foi elaborado com maior poder de computabilidade.

De acordo com a tese de Church, todos os modelos razoáveis de procedimento são semelhantes, e a MT se mostrou simples e adaptável o suficiente para possibilitar todas as demonstrações dos principais resultados. Assim, podemos considerar a máquina de Turing como o mais importante modelo usado para a observação do que é ou não computável.

Pode-se usar uma MT como aceitador ou reconhecedor (RANGEL, 1997), ou ainda como a implantação de um procedimento mais geral, que transforma uma cadeia de entrada em uma cadeia de saída.

As máquinas de Turing tradicionais são constituídas de fitas unidimensionais, porém ao observar o pensamento restritivo desta composição, estudos foram realizados com intuito de adicionar fitas e dimensões no seu funcionamento (CUNHA, 2012). Logo, surgiu o interesse sobre geometria de fitas alternativas, e uma série de resultados sobre a eficiência de máquinas com essas características foram aparecendo.

Cunha (2012) com a obtenção desses resultados relata também que qualquer máquina de Turing pode ser modelada como um grafo bidimensional, com nós correspondentes a células de fita e as setas correspondentes as transições permitidas. Porém, para ser considerado um grafo, é necessário que critérios sejam satisfeitos como: singularidade das cores de saída, homogeneidade, fita infinita, grafo ligado, função de transição finita e o retorno a célula que veio.

Portanto, com o poder restrito das máquinas de Turing tradicionais e a busca de maior poder computacional por meio da inserção de fitas e dimensões no funcionamento dessas máquinas, como também na resolução de problemas insolúveis, pode-se mostrar os grafos, mais precisamente os grafos de Cayley, como dispositivos de potencialidade na constituição dessas máquinas.

Logo, se propõe a introdução dos grafos de Cayley no funcionamento conjunto com as máquinas de Turing, como meio de obter com maior eficácia os resultados citados anteriormente.

Tendo em vista os critérios mostrados, Cunha (2012) afirma que é o necessário para que o grafo seja considerado um grafo de Cayley de um grupo infinito gerado finitamente, pois o mesmo atende todos os critérios, os quais comprovam sua afirmação. Assim, esta representação não é apenas mais um modelo de máquina equivalente para a classe de funções computáveis, a qual depende do grupo para o qual é gerado.

Para grupos de problemas solúveis, isto leva a máquinas que computam exatamente a classe de funções computáveis, mas o mesmo não é verdade para problemas insolúveis, para os quais estas máquinas são muito mais poderosas que as máquinas de Turing tradicionais.

2. OBJETIVOS

Os objetivos deste estudo estão divididos em objetivo geral e objetivos específicos. No objetivo geral, será abordado o tema de modo a englobar o assunto foco deste trabalho, e nos objetivos específicos, os tópicos dispostos no transcorrer do artigo.

2.1 OBJETIVO GERAL

Apresentar os grafos de Cayley como um aprimoramento mais genérico no funcionamento conjunto com as máquinas de Turing para a resolução de problemas insolúveis.

2.2 OBJETIVOS ESPECÍFICOS

- Demonstrar a capacidade de computabilidade das máquinas de Turing;
- Apresentar algumas das limitações existentes nesse modelo de dispositivo de computação;
- Mostrar os grafos de Cayley como um método de otimização dos resultados obtidos pelas máquinas de Turing para problemas insolúveis.

3. REFERENCIAL TEÓRICO

3.1 Máquinas de Estado Finito

Ao falar sobre as Máquinas de Estado Finito, Gersting (2001) indica que as mesmas constituem um modelo capaz de assimilar características de computadores atuais. Em termos gerais, o que se pode entender sobre tais máquinas é que as mesmas possuem, assim como os computadores digitais, operações sincronizadas por pulsos discretos. Elas possuem ações previsíveis, ou seja, funcionam de forma determinística, respondendo aos dados de entrada e produzindo dados de saída, mediante os processos aplicados aos primeiros e de acordo com uma sequência de estados, de ordem finita, que a máquina pode assumir.

3.2 Autômatos Finitos

Hopcroft (2002) expõe que, nas décadas de 1940 e 1950, tipos de maquinas mais simples, que hoje chamamos de “autômatos finitos”, foram estudados por diversos pesquisadores. Esses autômatos, propostos originalmente para modelar a função do cérebro, se mostraram extremamente úteis para uma grande variedade de outros propósitos.

Alguns conceitos, como autômatos finitos e certos tipos de gramáticas formais, são usados no projeto e na construção de importantes componentes de software. Outros conceitos como a máquina de Turing, ajudam a entender o que podemos esperar do nosso software.

Todavia, apesar dos autômatos finitos serem considerados desde antigamente máquinas mais simples, eles despertaram o interesse de diversos pesquisadores e têm grande importância na construção de componentes de softwares.

Menezes (2008) relata sobre autômatos finitos da seguinte maneira: “é um sistema de estados finitos, portanto possui um número finito e predefinido de estados, o qual constitui um modelo computacional do tipo sequencial”.

Desta forma, o autômato finito possui um conjunto de estados, que ao receber “entradas”, de forma sequencial movimenta sua “unidade de controle” entre os estados.

No que diz respeito ao conceito de autômatos, Hopcroft (2002) dispõe que:

> Um autômato finito tem um conjunto de estados, e seu “controle” se desloca de estado para estado em resposta a “entradas” externas. Uma das distinções cruciais entre classes de autômatos finitos é o fato desse controle ser “determinístico”, significando que o autômato não pode estar em mais de um estado em qualquer instante, ou “não-determinístico”, significando que ele pode estar em diversos estados ao mesmo tempo.

Contudo, o sistema com autômatos determinísticos assume um único estado bem determinado em um tempo *t*. Ao mesmo tempo, o sistema com autômatos não-determinísticos pode assumir um conjunto de estados no mesmo tempo *t*.

Todavia, os autômatos finitos por apresentarem um sistema de estados finitos trazem algumas limitações na potencialidade do seu armazenamento. Na qual a máquina de Turing em suas características, traz uma evolução na composição desse modelo.

Portanto, as máquinas de Turing constituem um aprimoramento em relação às máquinas de estado finito tradicionais, pois superam as limitações existentes em tais modelos, principalmente devido a uma característica que aumenta de forma elevada suas perspectivas: a máquina de Turing possui memória auxiliar ilimitada (MENEZES, 1998).

Desta forma, mesmo sendo as máquinas de estado finito capazes de assimilar características dos computadores atuais, as máquinas de Turing se mostram capazes de superar restrições existentes nesse modelo.

3.3 Autômatos de Pilha

Autômatos de pilha são formalismos (máquinas) capazes de reconhecer as Linguagens Livres de Contexto (COSTA, 2008).

Na definição informal Delamaro (1998) expõe que:

> Um autômato de pilha (AP) pode ser visto como uma máquina composta de uma fita de leitura unidirecional, um controle de estado com um registrador de estado atual e uma pilha. No AP a fita e a pilha são divididas em células que comportam apenas um símbolo cada uma. O cabeçote de leitura da pilha é posicionado na célula do topo da pilha. No início, o registrador guarda o estado inicial do AP, a fita armazena a palavra de entrada a partir da sua primeira célula, o cabeçote da fita é posicionado na primeira célula da fita, e a pilha e definida como vazia.

Já na definição formal, Costa (2008) relata que AP é uma sêxtupla <Σ,Γ,S,S0,δ,B>, onde:

- Σ é o alfabeto de entrada do AP;
- Γ é o alfabeto da pilha;
- S é o conjunto finito não vazio de estados do AP;
- S0 é o estado inicial, $S0 \in S$;
- δ é a função de transição de estados, $\delta: S \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow$ conjunto de subconjuntos finitos de $S \times \Gamma^*$
- B é o símbolo da base da pilha, $B \in \Gamma$.

Os autômatos de pilha em suas características apresentam maior poder em relação aos autômatos finitos, uma vez que apresentam um “espaço de armazenamento” extra, utilizado durante o processamento de uma cadeia, definido como uma pilha que caracteriza uma memória auxiliar onde pode inserir e remover informações.

Essa pilha é fundamental para o diferencial entre o autômato finito e o autômato de pilha, na qual adiciona poder aos autômatos finitos. Logo, os AP possuem o mesmo poder de reconhecimento das GLC’s.

O autômato finito mostra-se incapaz de reconhecer certas linguagens devido a sua incapacidade de memorizar as informações da cadeia analisada, já o autômato de pilha é capaz desse reconhecimento.

Costa (2008) dá a representação do AP da seguinte forma:

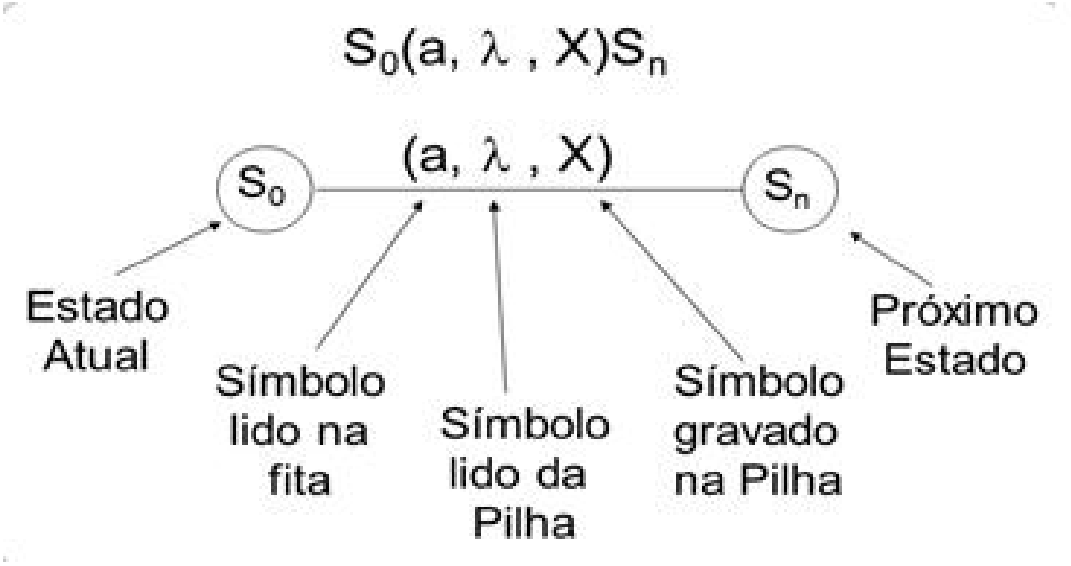


Figura 1: Representação do autômato de pilha
Fonte: Costa, 2008.

4. Máquinas de Turing

Ao falar sobre máquinas de Turing (MT), Rangel (1997) as retrata como um modelo matemático do processo de computação de composição simples, uma vez que Turing buscava a universalidade da aceitação do seu modelo.

Para Gersting (2001), ela é uma máquina conceitual, a qual recebe em seu funcionamento dados de entrada, alterando o estado da máquina e disponibilizando uma saída compatível. Trata-se, portanto, de uma máquina de estado finito que agrega possibilidades adicionais, como a de reler os dados de entrada, além de escrever e apagar por cima dos mesmos.

Ainda sobre o conceito de máquina de Turing, Menezes (2008) afirma que se trata de um autômato cuja fita não possui tamanho máximo e que pode ser usado simultaneamente como dispositivo de entrada, de saída e de memória de trabalho.

Assim, apesar de sua simplicidade, uma máquina de Turing possui, no mínimo, o mesmo poder computacional de qualquer computador de propósito geral. O modelo de máquina de Turing é constituído de três partes:

1) Fita – usada simultaneamente como dispositivo de entrada, de saída e de memória de trabalho;

2) Unidade de Controle – reflete o estado corrente da máquina. Possui uma unidade de leitura e gravação, a qual acessa a célula da fita de cada vez e se movimenta para a esquerda ou para a direita;

3) Programa, Função Programa ou Função de Transição – função que define o estado da máquina e comanda as leituras, as gravações e o sentido de movimento da cabeça.

Mesmo a máquina de Turing sendo um modelo computacional de composição simples, ele apresenta variadas características no seu funcionamento e sua composição, como também o mesmo poder computacional das máquinas atuais.

Delamaro (1998) expõe o seguinte: “As MT’s podem ser vistas como as mais simples “máquinas de computação”, servindo para determinar quais funções são computáveis e quais não são”.

Desta forma, pode-se afirmar que a MT é um modelo computacional suficientemente genérico, capaz de determinar a computabilidade das funções.

Por sua vez, Chaer e Rocha (2009) tratam as máquinas de Turing como elementos definidos por regras e mostram que assim como os autômatos de estados finitos são modelos determinados por conjuntos finitos de configurações e de regras, possibilitados de, quando uma cadeia de símbolos, apresentar (ou não) uma cadeia de saída e um resultado de aceitação ou rejeição.

Desta forma, podemos observar características similares das máquinas de Turing com os autômatos finitos, uma vez que ambos são determinados por regras e julgamentos de aceitação ou não de cadeias.

Mas a máquina de Turing apresenta uma fita unidimensional que contém um número ilimitado de células cada uma das quais pode conter um único símbolo. Logo, essa é a sua característica distintiva em relação aos autómatos finitos (que não têm dispositivo de armazenamento) e aos autómatos de pilha (que armazenam numa pilha).

De acordo com Zilio (2009):

> A hipotética máquina de Turing seria constituída por uma fita de dados de tamanho infinito, mas de estados finitos (“*finite state machine*”); por um processador de informações; e por um cabeçote

> capaz de ler, apagar e escrever informações na fita, além de poder movimentá-la. A máquina seria capaz de processar informações serialmente, com “memória” capaz de recordar qual a função do símbolo que está inscrito na fita e qual o estado da máquina no momento da leitura, podendo, assim, determinar a próxima ação (que é efetivada pelo cabeçote) e, conseqüentemente, o próximo estado finito da máquina (que está inscrito na fita). A universalidade da máquina de Turing consiste na possibilidade de imputar nela qualquer algoritmo, não havendo, assim, ao menos em princípio, limites para os tipos de processos que ela poderia instanciar. A máquina de Turing acabou por instituir o padrão de funcionamento de todas as máquinas digitais que conhecemos.

Logo, a composição da MT possibilitou o processamento serial de informações, como também, a principio a atribuição de qualquer algoritmo. Portanto, o padrão de funcionamento das máquinas atuais parte do pressuposto da constituição das máquinas de Turing.

Dentre outros também é interessante analisar o conceito de Hopcroft (1939) sobre Máquinas de Turing:

> Essas máquinas são autômatos que modelam o poder dos computadores reais. Elas nos permitem estudar a decidibilidade, a questão do que pode ou não pode ser feito por um computador. Elas também tornam possível distinguir problemas tratáveis – aqueles que podem ser resolvidos em tempo polinomial – dos problemas intratáveis – aqueles que não o podem.

Tendo em vista que as máquinas de Turing nos permitem analisar a decidibilidade dos problemas, como também os problemas tratáveis ou intratáveis, podemos concluir que elas ajudam tanto na obtenção de soluções para problemas indecidíveis, como na distinção de problemas intratáveis. Em outras palavras, mostrando o que o computador é capaz de realizar.

4.1 Constituição da Máquina de Turing

Rangel (1997) afirma que uma máquina de Turing M é uma tupla M = < K, , , , i, F >, onde K é um conjunto (finito, não vazio) de estados, é o alfabeto (finito) de símbolos da fita, é o alfabeto de símbolos de entrada, é a função de transição, i ϵ K é o estado inicial, e F K é o conjunto de estados finais.

Já Gersting, em outra definição formal da máquina de Turing, expõe que:

> Sejam S um conjunto finito de estados e I um conjunto finito de símbolos para a fita (o alfabeto da fita), incluindo um símbolo especial b. Uma máquina de Turing é um conjunto de quíntuplas da forma (s, i, i´, s´,d), onde s, s´ ϵ S; i, i´ ϵ I; e d ϵ {D,E}, tais que duas quíntuplas distintas nunca começam com os mesmos símbolos s e i (GERSTING, 2001, p.420).

Gersting (2001) representa essa definição com a Figura 2:

s	***i***	***i'***	***s'***	***d***
estado atual	*símbolo atual*	*símbolo impresso*	*próximo estado*	*direção do movimento*

Figura 2. Quíntupla para máquina de Turing

Fonte: Gersting, 2001.

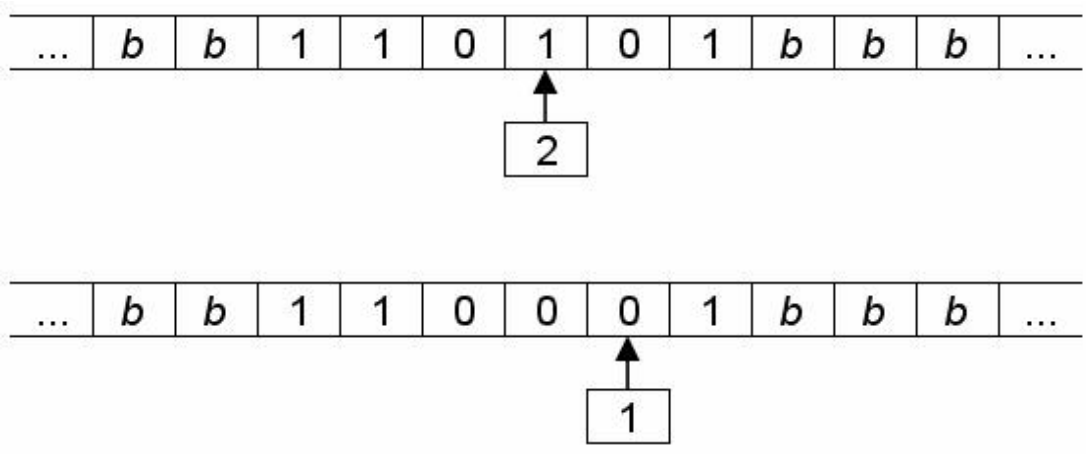


Figura 3. Exemplo de fita para máquina de Turing e comportamento da mesma para a quíntupla (2,1,0,1,D)
Fonte: Gersting, 2001.

Costa (2008) dispõe sobre o funcionamento da máquina de Turing por meio da Figura 4:

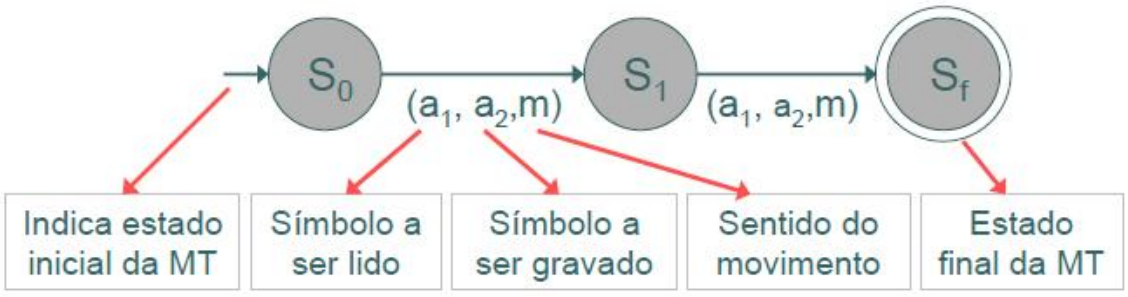


Figura 4. Funcionamento da máquina de Turing
Fonte: Costa, 2008.

O funcionamento da máquina, de modo geral, parte do estado inicial S_o, lendo a cabeça de leitura/gravação o símbolo contido em uma célula da fita de cada vez e gravando um novo símbolo. No caso, lê a_1 e grava a_2 respectivamente, e após estas ações, a cabeça se move em uma célula, de acordo com o sentido do movimento. Por fim, ela chega ao seu estado final.

4.2 Gramáticas Irrestritas

Gramática em sua definição consiste em um sistema desenvolvido para reescrever cadeias (ROSA, 2010).

Por sua vez, uma Gramática Irrestrita é uma gramática sem qualquer restrição nas suas produções. Portanto, o termo “irrestrita” serve somente para destacar tal fato (MENEZES, 2008, p.187).

Logo, se tomarmos a forma geral não impondo restrições, obteremos uma gramática irrestrita, porém várias restrições foram feitas, depois, para se obter gramáticas específicas.

Finalmente, uma linguagem L é recursivamente enumerável se, e somente se, L é gerada por uma gramática irrestrita.

Rosa (2008) mostra que os outros tipos de gramáticas consideradas (livre de contexto, sensível ao contexto) restringem a forma das produções, já a gramática irrestrita, não. Tentaremos mostrar que as gramáticas irrestritas são equivalentes às Máquinas de Turing.

Assim, nota-se, como o próprio nome já diz que a gramática irrestrita não apresenta restrição na produção e também que a classe da linguagem recursivamente enumerável existe, se for gerada por uma gramática irrestrita.

5. Classes de Linguagens

As restrições impostas nas diferentes gramáticas, fez com que surgissem diferentes linguagens geradas por cada gramática específica.

5.1 Linguagens Regulares

De acordo com a hierarquia de Chomsky, a classe das Linguagens Livres de Contexto contém propriamente a classe das Linguagens Regulares (Menezes, 2008). Entretanto constitui uma classe de linguagens relativamente restrita, sendo fácil definir linguagens que não pertencem a esta classe.

A classe das Linguagens Regulares é, portanto, extremamente restrita. Uma vez que está na composição de outra classe tratada pelo autor como “relativamente restrita”.

5.2 Linguagem Livre de Contexto

Uma gramática livre de contexto consiste em um conjunto de variáveis, um conjunto de símbolos terminais e uma variável de inicio, bem como as produções (Hopcroft, 1939).

Portanto, começando com o símbolo de inicio, podemos derivar sequências de caracteres (*strings*) substituindo repetidamente uma variável pelo corpo de alguma produção com essa variável na cabeça.

A linguagem da gramática livre de contexto é o conjunto de *strings* de terminais que podemos derivar dessa forma; chamada de linguagem livre de contexto (Hopcroft, 1939).

O estudo dessas classes é de fundamental importância na computação, pois compreende um universo mais amplo de linguagens (comparando com o das regulares), que possuem algoritmos reconhecedores e geradores que implementam a linguagem e que possuem eficiência razoável.

Para mostrar que uma determinada linguagem é livre de contexto, é suficiente expressá-la usando os formalismos Gramática Livre de Contexto ou Autômatos com Pilha (Menezes, 2008). Entretanto, a demonstração de que uma linguagem é livre de contexto necessita ser realizada caso a caso.

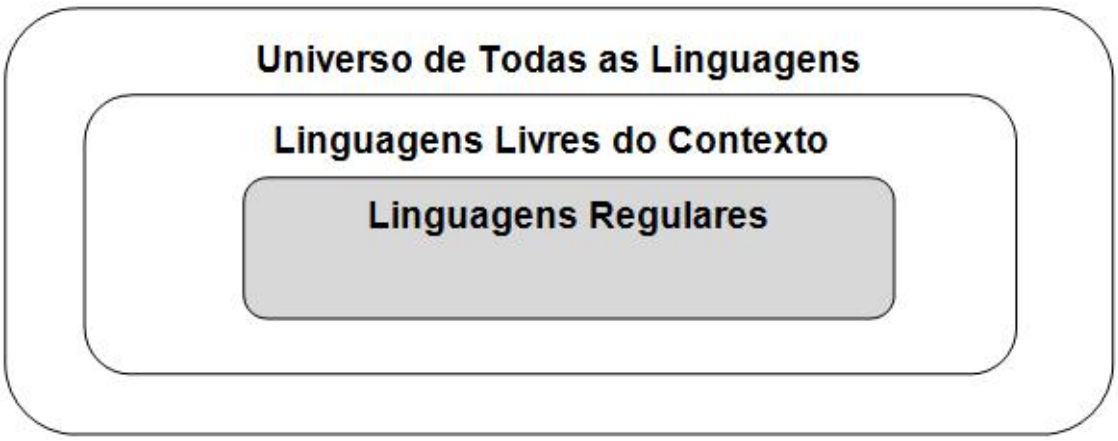


Figura 5. Relação entre as classes de linguagens

Fonte: Menezes, 2008

5.3 Linguagens Sensíveis ao Contexto

Segundo Menezes (2008), Linguagens Sensíveis ao Contexto são aquelas que podem ser aceitas por uma Máquina de Turing com Fita Limitada, ou seja, uma máquina de Turing finita. O correspondente formalismo axiomático é a Gramática Sensível ao Contexto, em oposição ao termo “livre do contexto”.

Esta classe de linguagem é gerada por gramáticas com tipos de produções mais gerais que as produções das gramáticas livres de contexto, mas com algum tipo de restrição ou, do ponto de vista de autômatos, como certo tipo de máquinas de Turing que seja mais poderosa que os autômatos a pilha não determinísticos.

A classe das Linguagens Sensíveis de Contexto está contida propriamente na classe das Linguagens Recursivas. Trata-se de uma classe especialmente importante, pois inclui a grande maioria das linguagens aplicadas.

As linguagens recursivas, assim como as linguagens sensíveis de contexto têm a possibilidade de aceitação por máquinas de Turing finitas. A relação entre as diversas classes é ilustrada na Figura 6.

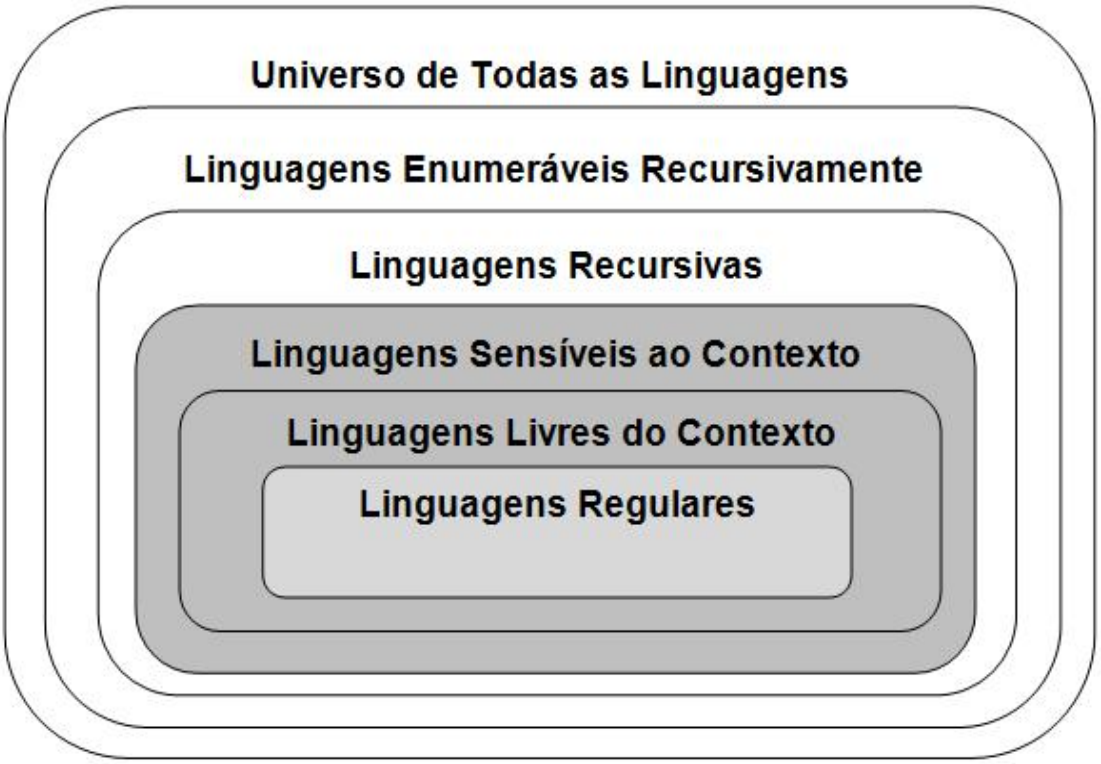


Figura 6. Relação entre as classes de linguagens

Fonte: Menezes, 2008

5.4 Linguagem Recursivamente Enumerável

Uma linguagem é recursivamente enumerável (RE) se existe uma máquina de Turing que aceita toda cadeia da linguagem, e não aceita cadeias que não pertencem à linguagem. Porém, “não aceita” não é o mesmo que “rejeita”. A máquina de Turing poderia entrar num *loop* infinito e nunca parar para aceitar ou rejeitar a cadeia (ROSA, 2010).

Portanto, mesmo que a máquina entre em processo de repetição de passos (*loop*) infinito, pode ser considerada uma linguagem recursivamente enumerável, pois, esse processo não corresponde à rejeição de uma determinada cadeia da linguagem.

Hopcroft (2002) trata as Linguagens Recursivamente Enumeráveis como um conjunto de linguagens que podemos aceitar usando uma máquina de Turing. Assim, podemos entender que se tratam de linguagens que podem ser reconhecidas ou aceitas por qualquer tipo de dispositivo de computação. Considerando que, segundo a Hipótese de Church, a máquina de Turing é o dispositivo mais geral de computação, então a classe das linguagens recursivamente enumeráveis representa o conjunto de todas as linguagens que podem ser reconhecidas mecanicamente e em um tempo finito.

A classe das Linguagens Recursivamente Enumeráveis inclui algumas para as quais é impossível determinar mecanicamente se uma palavra não pertence à linguagem. Portanto é conveniente definir uma subclasse da classe das Linguagens Enumeráveis Recursivamente, composta pelas Linguagens Recursivas, para as quais existe pelo menos uma máquina de Turing que pára para qualquer entrada, aceitando ou rejeitando.

O processo de inserção de uma subclasse da linguagem recursivamente enumerável que aceite ou rejeite uma determinada cadeia foi proposto como forma de solução. Tendo em vista que a classe das linguagens recursivamente enumeráveis possui palavras não pertencentes à linguagem e que isso provocaria a não aceitação de uma determinada cadeia, acarretando em *loop* infinito.

5.5 Linguagens Recursivas

Uma linguagem é designada por recursivas ou decidível se para ela existe uma máquina de Turing que a reconhece, isto é, que pára para todos os dados e que pára num estado final se a palavra dada pertence à linguagem e pára num estado não final se a palavra dada não pertence à linguagem (MOREIRA E MATOS, 1996).

A linguagem recursiva é uma subclasse da linguagem recursivamente enumerável, na qual, dada a linguagem, existe o reconhecimento (aceitação ou rejeição), por uma máquina de Turing.

Uma linguagem L é recursiva de L=L(M) para alguma máquina de Turing M tal que (HOPCROFT, 2002):

1) “Se w está em L, então M aceita (e, portanto pára)”.
2) “Se w não está em L, então M pára eventualmente, embora nunca entre em um estado de aceitação”.

Uma MT desse tipo corresponde à noção informal de “algoritmo”, que é uma sequência bem definida de etapas que sempre termina e produz uma resposta. Dada uma linguagem L como um “problema”, então o problema L será decidível se for uma linguagem recursiva e será chamado indecidível se não for uma linguagem recursiva.

6. METODOLOGIA

Este artigo trata-se de uma pesquisa bibliográfica, a fim de mostrar o conceito de máquina de Turing, a sua potencialidade como dispositivo geral de computação e também as suas limitações para problemas insolúveis. Logo, também mostra a inserção de grafos de Cayley nas máquinas de Turing como aprimoramento desse dispositivo para melhor resolução desses problemas.

No capítulo 7, é mostrada a limitação foco do trabalho, a indecidibilidade, como também exemplos de problemas indecidíveis, é o caso do problema da parada.

No capítulo 8, mostram-se os grafos de Cayley discorrendo o seu funcionamento, sua composição e o seu poder diante das máquinas de Turing padrão para problemas de palavras insolúveis.

7. Limitações das Máquinas de Turnig

Tendo demonstrado a potência e flexibilidade de computação efetiva, como capturadas nas máquinas de Turing, mostraremos agora que mesmo esta potência e flexibilidade têm limites. Turing mostrou que certos problemas naturais em computação não podem ser computados por nenhum algoritmo, real ou imaginado (ROSA, 2010). Ou seja, o que nós chamamos de problemas indecidíveis são aqueles para os quais as máquinas de Turing são incapazes de mostrar como saída uma solução, ou seja, resolvê-los em tempo hábil.

7.1 Indecidibilidade

Há conjuntos que não são recursivamente enumeráveis, ou seja, linguagens para as quais não é possível desenvolver uma máquina de Turing que as reconheça (MENEZES, 2008).

Logo, linguagem indecidível é uma linguagem em que não existe uma máquina de Turing que ao receber uma cadeia de entrada libera uma saída, seja aceitando-a ou rejeitando-a, dessa forma a máquina de Turing não consegue decidir tal linguagem.

7.2 Linguagem de diagonalização

De acordo com Hopcroft (2002), é possível encontrar no processo de enumeração de máquinas de Turing a máquina Mi, a i-ésima máquina de Turing. Uma máquina de Turing M cujo código é Wi, o i-ésimo string binário. Muitos inteiros não correspondem em absoluto a qualquer máquina de Turing. Se Wi não for um código de máquina de Turing válido, tomaremos Mi como a MT com um estado e nenhuma transição, isto é, para esses valores de i, Mi é uma máquina de Turing que pára imediatamente quando aplicada sobre qualquer entrada. Desse modo, L(Mi) é Ø se Wi deixa de ser um código de MT válido.

A linguagem Ld, a Linguagem de diagonalização, é o conjunto de *strings* Wi tais que Wi não está em L(Mi). Isto é, Ld consiste em todos os strings w tais que MT M cujo código é w não aceita quando recebe w como entrada.

A razão para o nome linguagem de “diagonalização” pode ser entendida através da Figura 7:

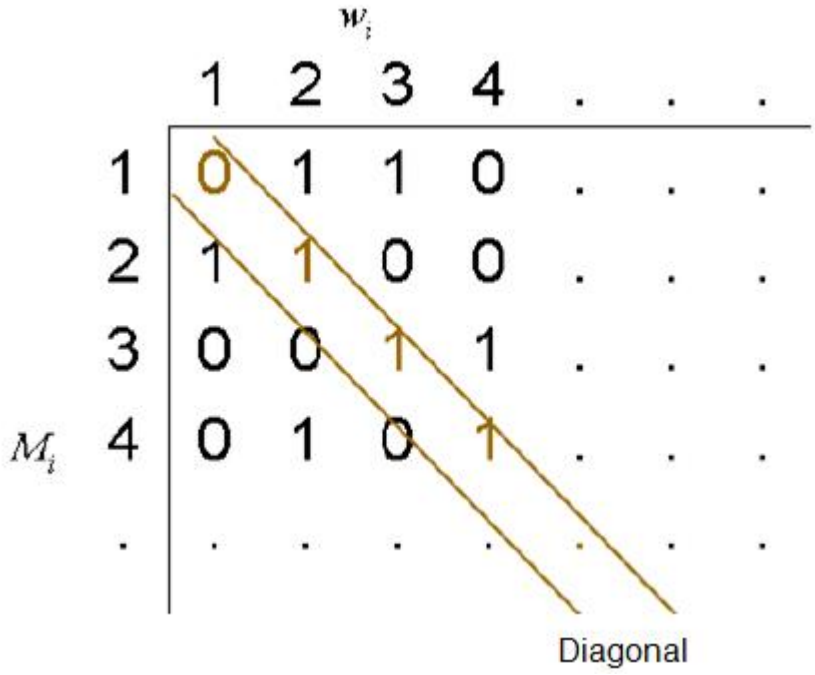


Figura 7: Tabela que representa a aceitação de cadeias por TM.

Fonte: Rosa, 2008.

A tabela informa que para todo i (linha) e j (coluna), se a máquina de Turing Mi aceita a cadeia de entrada, wj: 1 significa “sim” e 0 significa “não”.

Os valores na diagonal dizem se Mi aceita wi, e que para construir a Ld, complementa-se a diagonal. O truque de complementar a diagonal para construir o vetor característico de uma linguagem que não pode ser a linguagem que aparece em nenhuma linha é chamado de diagonalização.

Isto funciona porque o complemento da diagonal é ele próprio um vetor característico que descreve a pertinência em alguma linguagem, a Ld (HOPCROFT, 1939). Este vetor característico discorda em alguma coluna com toda linha da tabela. Portanto, o complemento da diagonal não pode ser o vetor característico de nenhuma máquina de Turing.

Com isso é apresentado o Teorema, onde Ld não é uma linguagem recursivamente enumerável (RE). Ou seja, não há nenhuma máquina de Turing que aceite Ld.

Para a prova do Teorema (Hopcroft, 1939):

> Suponha que Ld seja T(M) para alguma máquina de Turing M. Como Ld é uma linguagem sobre o alfabeto {0; 1}, M estaria na lista das máquinas de Turing, já que esta lista inclui todas as máquinas de Turing com alfabeto de entrada {0; 1}. Portanto, há pelo menos um código para M, por exemplo, M = Mi.

Se wi está em Ld, então Mi aceita wi. Mas então, pela definição de Ld, wi não está em Ld, porque Ld contém apenas aqueles wj tal que Mj não aceita wj. Similarmente, se wi não está em Ld, então Mi não aceita wi. Portanto, por definição de Ld, wi está em Ld.

Como wi não pode estar e não estar em Ld ao mesmo tempo, há uma contradição na suposição de que M existe. Ou seja, Ld não é uma linguagem recursivamente enumerável.

7.3 Problema da Parada

O problema da parada foi um dos primeiros problemas a serem provados como indecidíveis. Em 1936, Alan Turing, provou que não pode existir um procedimento geral para resolver tal problema (VIZOTTO, 2011).

A prova da indecidibilidade do problema da parada usa uma técnica chamada diagonalização, descoberta pelo matemático George Cantor em 1873. O problema de determinar se uma máquina de Turing aceita uma dada cadeia de entrada (VIZOTTO, 2011):

AMT = {(M, w) |M é uma MT e M aceita w}.

Teorema: AMT é indecidível, ou seja, não conseguimos construir uma máquina de Turing que sempre pára aceitando ou rejeitando a entrada.

AMT é Turing - reconhecível, na qual reconhecedores são mais poderosos que decisores.

A máquina de Turing U a seguir reconhece AMT, em que: U = Sobre a entrada (M, w), onde M é uma MT e w é uma cadeia:

1) Simule M sobre a entrada w.

2) Se M em algum momento entra no seu estado de aceitação, aceite, se M em algum momento entra no seu estado de rejeição, rejeite.

Então se conclui que a máquina entra em processo de repetição de passos *(loop)* sobre a entrada (M, w) se M entra em *loop* sobre w, é por isso que a máquina não decide AMT. Se o algoritmo tivesse alguma forma de determinar que M não iria parar sobe w, ele poderia dizer rejeite, porém o algoritmo não tem como fazer essa determinação. Então esse é denominado o problema da parada.

Hopcroft (1939) relata a definição do problema da parada da seguinte maneira: Definindo H(M) para a MT M como conjunto de entradas w tais que M pára a partir de uma entrada dada w, independente se M aceitar ou não w. Então o problema da parada é o conjunto de pares (M, w) tais que w está em H(M). Este problema é um exemplo de algo que é recursivamente enumerável (RE), mas não recursivo.

7.4 Indecidibilidade do problema da parada

Um dos mais importantes problemas conhecidos como indecidíveis é o problema da parada. Turing utilizou o método da redução ao absurdo para conseguir provar que o problema da parada é indecidível, ou seja, ele assumiu que ele era decidível e obteve uma contradição.

Vizzotto (2011) mostra a prova da indecidibilidade do problema da para com o seguinte Teorema:

> Dada a linguagem AMT = {(M, w) |M é uma MT e M aceita w}. Em sua prova supomos que AMT seja decidível e assim obtemos uma contradição, ou seja, suponha que H seja um decisor para AMT. Sobre a entrada (M, w), na qual M é uma MT, e w é uma cadeia, H pára e aceita se M aceita w. Além disso, H pára e rejeita se M falha em aceitar w.

Assim, H é uma MT tal que: H(M, w) = aceite, se M aceita w ou rejeite, se M não aceita w.

Agora devemos construir uma nova MT D com H como sub-rotina. Essa nova MT chama H para determinar o que M faz quando a entrada para M é sua própria descrição (M). Então D faz o oposto, ou seja, ele rejeita se M aceita e aceita se M rejeita.

Assim, D = “Sobre a entrada (M), onde M é uma MT:

1) Rode H sobre a entrada {M, (M)}.
2) Dê como saída o oposto de que H dá como saída, ou seja, se H aceita, rejeite e se H rejeita, aceite.

Esse processo consiste em rodar uma máquina sobre a própria descrição, ou seja, rodar um programa consigo próprio como entrada. Assim, rodando D{(M)} temos: aceite, se M rejeita (M), ou rejeite, se M aceita (M). Então, se rodarmos D{(D)} temos: aceite, se D não aceita (D), e rejeite, se D aceita (D), o que a mesma define como uma contradição.

Para finalizar a prova declaramos a suposição de que MT H decida AMT, então usa H para construir uma MT D que, quando recebe uma dada entrada (M), aceita exatamente quando M rejeita a entrada (M). Logo ao rodar D sobre si própria, as máquinas tomam as seguintes ações, sendo a última uma contradição (VIZOTTO, 2011):

1) H aceita (M, w) exatamente quando M aceita w;
2) D rejeita (M) exatamente quando M aceita (M);
3) D rejeita (D) exatamente quando D aceita (D).

De certo modo, é mostrado que, ao rodar uma máquina sobre sua própria descrição, encontraremos a prova por contradição do problema da parada, onde ao executar os passos, chegaremos ao absurdo.

8. Grafos de Cayley

Quando Turing definiu sua A-máquina, que viria a ser chamada máquina de Turing, ele imaginou uma máquina cuja memória fosse disposta ao longo de uma fita unidimensional. Isso parecia um pouco arbitrário e talvez excessivamente restritivo, tendo em vista que o foco no momento fosse o termo “computável” e a adição de fitas e dimensões de classe mais simples. Com isso, rapidamente máquinas com várias fitas e multidimensionais foram propostas (CUNHA, 2012).

O estudo da geometria de fitas alternativas caiu em desuso por algum tempo, então a comunidade da teoria da complexidade, em seguida, reacendeu o interesse nesse assunto, considerando não a classe de funções computáveis por MT’s, mas o tempo e a complexidade do espaço de funções e geometrias diferentes. Esses estudos levaram a uma série de resultados sobre a eficiência de máquinas com uma, muitas ou grandes dimensões de fitas.

No entanto, ao ver as restrições contidas em fitas unidimensionais, a comunidade da teoria da complexidade se viu na necessidade de buscar resultados sobre máquinas de grandes dimensões de fitas.

O problema é a questão de se permitir que grafos Cayley possam ser fitas que são apenas um modelo de máquina equivalente para a classe de funções computáveis. Na verdade, tal questão dependerá da estrutura do grupo de que o grafo de Cayley é gerado, ou seja, para grupos com problemas de palavras solucionáveis, este de fato, leva a máquinas que calculam a classe de funções computáveis. No entanto, para grupos com problemas de palavras indecidíveis, os

grafos de Cayley são estritamente mais poderosos do que máquinas padrão de Turing.

Contudo, os grafos Cayley se mostram mais eficientes para problemas relacionados às palavras indecidíveis, já para problemas de palavras decidíveis, este é levado a máquinas que calculam essas funções, por exemplo, máquinas de Turing padrão.

Cunha (2012) relata que:

> Gráficos Cayley são, em certo sentido, o grau “direito” de generalidade e leva à seguinte definição: Seja G um grupo infinito e S ⊂ G um grupo gerador finito de G que é fechada sob inversas. Em seguida, o gráfico de Cayley G gerado por S é chamado o grafo de fita, (G, S).

Desta forma, os grafos Cayley apresentam um elevado poder comparado às máquinas de Turing padrão, mostrando também a sua generalidade e a eficiência em problemas indecidíveis.

8.1 Máquinas de Turing em grafos de Cayley

Podemos ter a seguinte definição: seja G = (G1, . . . , Gn) sendo um grupo finitamente gerado com o conjunto {G1, . . . , Gn} fechada sob inversas. Em seguida, uma máquina de Turing sobre G com a geração de definir (G1. . . , Gn) é um 7-tupla, (Q, Γ, b, Σ δ, q0, F) onde (CUNHA, 2012):

- Q é o conjunto finito de estados;
- Γ é o conjunto finito de símbolos de fita;
- B ∈ Γ é um símbolo designado branco;
- Σ ⊂ Γ \ {b} é o conjunto de símbolos de entrada;
- Δ: Q × Γ → Q × Γ × {g1,. . . , Gn} é a função de transição;
- Q0 ∈ Q é o estado inicial;
- F ⊂ Q é o conjunto de estados terminais (normalmente um para aceitar e rejeitar um).

Em relação à classe de funções computáveis, as máquinas de Turing em grafos de Cayley dependem da sensibilidade das propriedades do grupo e da produção do grafo de fita. Na qual é mostrado o exemplo que: Seja (G, S) um grafo de fita. Há uma máquina de Turing sobre (G, S) que pode resolver o problema da palavra para G.

A prova deste processo é facilmente descrita com a entrada de uma fita unidimensional separada, na qual, também resolve o problema da palavra.

Formalmente representando uma dada sequência de geradores, a cabeça da fita no gráfico vai marcar primeiro a origem com um símbolo especial e seguir as bordas dada pela sequência na fita de entrada. Se, depois ela alcançar a extremidade da entrada, a cabeça da fita no grafo lê o símbolo especial, aceitar. Caso contrário, rejeitar. Claramente, esta máquina só aceita se a cabeça sobre a fita do grafo retorna à seu ponto de partida.

Contudo, se a cabeça da fita lê o simbolo especial que a própria marcou na origem no início da execução, ela aceitará a sequência, caso contrário ele rejeitará, ou seja, se a cabeça da fita voltar para o inicío ele aceitará, se não, rejeitará.

Existem grupos com problemas de palavras indecidíveis, o que nos leva a acreditar que máquinas de Turing em um determinado grupo, os grafos de Cayley, com problemas de palavras indecidíveis são estritamente mais poderosas do que o padrão de máquinas de Turing. No entanto, isto requer que as máquinas de Turing sobre o referido grupo seja capaz de calcular todas as funções computáveis. Felizmente, este é o caso (CUNHA, 2012).

Logo, fica evidente o poder das máquinas de Turing em grafos de Cayley, uma vez que essas máquinas se tornam capazes de calcular todas as funções computáveis, como também, o maior desempenho em problemas de palavras indecidíveis, aumentando assim o poder computacional das máquinas de Turing, e fazendo com que problemas, antes intratáveis, tenham solução.

9. CONCLUSÃO

Dentre todos os modelos computacionais propostos, a máquina de Turing é ainda hoje o modelo mais genérico e de maior eficiência na resolução de problemas solúveis. Considerado o mais importante modelo na definição do que é ou não computável.

Porém, mesmo com todo o poder computacional existente nessas máquinas, elas se mostram limitadas na resolução de problemas insolúveis.

Essas máquinas eram constituídas de fitas unidimensionais, logo ao perceber o poder restritivo desse modelo, diversos estudos foram feitos a fim de melhorar o desempenho nessas limitações, os quais propuseram a adição de fitas e dimensões, surgindo assim o estudo com fitas alternativas e grafos bidimensionais, mais precisamente os grafos de Cayley.

Os grafos de Cayley é um modelo genérico e eficiente, que se mostra equiparado às máquinas de Turing tradicionais quanto a problemas de palavras solúveis, uma vez que apresenta o mesmo poder computacional dessas máquinas, porém ao considerarmos os problemas de palavras insolúveis, eles se mostram estritamente mais poderosos que as máquinas tradicionais, trazendo assim solução para problemas que antes não tinham.

REFERÊNCIAS

CHAER I.; ROCHA R. L. A., **Um estudo sobre a Ação Elementar de Consulta no Formalismo Adaptativo.** 2009.

COSTA, Y. M. G. Máquinas de Turing. Slides. Ciências da Computação. Departamento de Informática. Universidade Estadual de Maringá, 2008.

CUNHA, A**. Turing Machines on Cayley Graphs.** University of Michigan, Ann Arbor, MI 48109, USA, 2012.

DELAMARO, M. E. **Linguagens Formais e Autômatos**. Maringá: UEM, 1998.

GERSTING, Judith L. **Fundamentos matemáticos para ciência da computação**. 4.ed. Rio de Janeiro: LTC, 2001.

HOPCROFT, J. E., 1939 – **Introdução à teoria dos autômatos, linguagem e computação**/ John E. Hopcroft, Rajeev Montwani, Jeffrey D. Ullman; tradução da 2.ed. original de Vandenberg D. de Souza. – Rio de Janeiro: Elsevier, 2002.
MENEZES, Paulo Blauth. **Linguagens Formais e Autômatos**. Porto Alegre: Editora Sagra-Luzzatto, 1998;

MOREIRA, N.; MATOS, A., **Máquinas de Turing: Uma introdução,** 1996, rev. 2001.

OLIVEIRA P. G.; SILVA S. A. M., **Construção de simuladores gráficos para Teoria da computação: Uma proposta para o Ensino do conceito de máquinas de Turing**, 2006.

PALAZZO, L. A. M., **Linguagens Formais e Autômatos: Máquina de Turing - Linguagens Sensíveis ao Contexto e Enumeráveis Recursivamente**, 2007.

RANGEL J. L., **Linguagens Formais**, 1997, p 7-15.

ROSA, J. L. G. **Linguagens Formais e Autômatos**. Editora LTC. Rio de Janeiro, 2010.

ROSA, J. L. G. SCC-0505 - **Teoria da Computação e Linguagens Formais.** Slides. Ciências de Computação. Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo, 2008.

VIZOTTO, J. K. **Problema da parada.** Slides. Teoria da Computação. Universidade Federal de Santa Maria.

ZILIO, D., **Inteligência artificial e pensamento: redefinindo os parâmetros da questão primordial de Turing**. Artigo publicado na revista: Ciências e cognição, 2009.