

LINGUAGENS FORMAIS E AUTÔMATOS MÁQUINA DE TURING

Autor(a): Me. Eng. Pedro Henrique Chagas Freitas

Revisor: Edilson José Rodrigues

Tempo de leitura do conteúdo estimado em 1 hora.

Introdução

Olá, caro(a) estudante, você sabia que o estudo sobre a máquina de turing é essencial no desenvolvimento da computação como conhecemos hoje? Para entender melhor, nesta unidade, vamos conhecer o estudo sobre as linguagens recursivamente enumeráveis e sensíveis ao contexto, visando explorar os limites do que pode ser reconhecido computacionalmente e a base que derivou os estudos sobre a máquina de turing. A partir disso, é possível demonstrar que existem infinitas, mas contáveis, linguagens computáveis, e infinitas e não contáveis linguagens não computáveis. Ou seja, existem mais linguagens não computáveis do que computáveis atualmente. Nesse contexto, o estudo dessa unidade será desenvolvido usando os formalismos da máquina de turing e gramáticas regulares.

Feita essa breve introdução, garanto que o assunto é extremamente interessante e meu papel será apresentar as temáticas sobre a máquina de turing, da forma mais simples e didática possível.

Boa leitura!

Definição, Modelo e Propriedades da Máquina de Turing

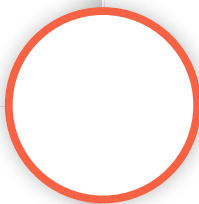
Para que você possa compreender a definição, o modelo e as propriedades da **máquina de Turing**, precisamos essencialmente entender a trajetória para chegar até aqui. A **ciência da computação** trata do conhecimento **matemático** desenvolvido ao longo da história humana, desde a Mesopotâmia, passando pela Grécia, até chegar aos vales do silício espalhados hoje pelo mundo, principalmente na China.

No início do século XX, ocorreu um grande impulso nas pesquisas com o objetivo de definir um modelo computacional suficientemente genérico, capaz de implementar qualquer função computável base para viabilizar a exploração dos limites do que pode ser computado (MENEZES, 2015, p. 204).

Em 1936, Alan Turing propôs um **formalismo** para a representação de **procedimentos efetivos**. O trabalho de Turing é particularmente significativo para a teoria da computação, porque foi o **primeiro** a identificar programas escritos para uma “máquina computacional”, como noções intuitivas do efetivamente computável.

A máquina de Turing é um formalismo muito simples, universalmente conhecido e provavelmente o mais usado como modelo teórico de computação. A intenção por trás do modelo é simular, na medida do possível, as atitudes **humanas** com relação à computação. O resultado foi a **fundamentação teórica** para o desenvolvimento do computador como conhecemos hoje. Atualmente, é aceita como uma formalização de um procedimento efetivo, ou seja, de um **algoritmo** ou **função** computável (DIVERIO, 2015).

REFLITA



Como teria evoluído a computação sem a fundamentação teórica da máquina de turing? Essa é uma boa questão, porque traz a discussão de um dos maiores passos na evolução humana, isto é, a computação como a conhecemos hoje. E, a partir disso, podemos nos perguntar como será a fundamentação teórica da aplicação da máquina de Turing na computação quântica? Quais novos desafios estão por vir no futuro? Reflita sobre isso.

Em 1936, Alonzo Church demonstrou a **tese de Church**, na qual afirmou que qualquer função computável poderia ser processada através de uma máquina de Turing. Dessa forma, existe um procedimento definido no qual uma máquina de Turing processará uma função computacional.

Porém, como a noção **intuitiva** de procedimentos não é matematicamente

precisa, é impossível apresentar formalmente se a máquina de Turing é, de fato, o modelo mais genérico de dispositivo computacional. Todavia, há **evidências** de que a tese é **verdadeira**, o que, por sua vez, reforçou que todos os demais modelos conhecidos propostos possuem, no máximo, a mesma capacidade computacional dessa máquina. Assim, a tese de Church foi assumida como uma hipótese para toda a teoria da computação, razão pela qual também é conhecida como hipótese de Church (MENEZES, 2015).

Resumindo, uma máquina de Turing é um **autômato** cuja fita não possui tamanho máximo e pode ser usada **simultaneamente** como dispositivo de entrada, de saída e de memória de trabalho.

As **linguagens** recursivamente **enumeráveis** ou linguagens **tipo 0**, por sua vez, são fundamentais na aplicabilidade da máquina de Turing, porque são elas que podem ser aceitas por uma máquina de Turing. Podemos considerar que, segundo a hipótese de Church, a máquina de Turing é o dispositivo computacional mais geral, logo, a classe das linguagens recursivamente enumeráveis representaria, a partir da máquina de turing, por definição, o conjunto de todas as linguagens que podem ser reconhecidas **mecanicamente** e em um tempo finito.

De maneira análoga às outras classes de linguagens, podem representar as linguagens recursivamente enumeráveis, usando um formalismo **axiomático** ou **gerador**, na forma de gramática, o qual denomina-se gramática irrestrita. Como o próprio nome indica, uma gramática irrestrita não possuirá qualquer restrição quanto à forma das produções. Portanto, o formalismo gramatical possui o mesmo poder computacional que o formalismo da máquina de Turing (DIVERIO, 2015).

Uma consequência importante do estudo das linguagens recursivamente enumeráveis é que, computacionalmente falando, existem mais problemas não computáveis (para os quais não existem máquinas de Turing capazes de processá-los) do que problemas computáveis (caso contrário). De fato, existem infinitos, mas contáveis, problemas computáveis, e infinitos e não contáveis problemas não computáveis.

Podemos definir, então, que a classe das linguagens recursivamente enumeráveis inclui algumas linguagens para as quais é impossível determinar mecanicamente se uma palavra não pertence à linguagem, ainda que seja possível determinar se pertence. Logo, um problema computável pode ser um problema **parcialmente** solucionável. Vejamos um exemplo: se L é uma destas linguagens com impossibilidade de determinação, então, por definição, para qualquer máquina de Turing M que aceita L , existe pelo menos uma palavra w não pertencente a L que, ao ser processada por M , a máquina entra em *loop* infinito. Assim, podemos afirmar que:

- se w pertence a L , M para e aceita a entrada;
- se w não pertence a L , M pode parar, rejeitando a palavra, ou permanecer processando indefinidamente.

Dessa forma, é conveniente definir uma **subclasse** da classe das linguagens enumeráveis recursivamente, composta pelas linguagens **recursivas**, dentre as quais existe, ao menos, uma máquina de Turing que, para qualquer **entrada**, aceitará ou rejeitará a mesma. Todo problema que se enquadra nesta classe de linguagens é denominado problema **solucionável** (MENEZES, 2015).

Indo, agora, mais fundo no funcionamento da máquina de Turing, temos que, intuitivamente, um algoritmo deve possuir as seguintes características, entre elas:

1. ter uma descrição finita;
2. consistir nos passos: discretos (em oposição ao contínuo); executáveis mecanicamente; executáveis em um tempo finito.

Partindo desse pressuposto, podemos derivar uma noção **intuitiva** sobre a máquina de Turing, proposta por Alan Turing, em 1936, que é um mecanismo simples que formaliza a ideia de uma pessoa que realiza cálculos.

Você se lembra que, há bastante tempo, os computadores tinham como razão

de existir a realização de cálculos? Pois bem, foi assim que tudo começou com a máquina de Turing. Apesar da sua simplicidade, o modelo máquina de Turing possui, no mínimo, a mesma capacidade, ou seja, o mesmo poder computacional de qualquer computador de propósito geral. Logo, o ponto de partida de Turing foi analisar a situação a partir da qual uma pessoa, com um instrumento de **escrita** e um **apagador**, poderia realizar cálculos em uma folha de papel e, a partir disso, derivar resultados lógicos (DIVERIO, 2015).

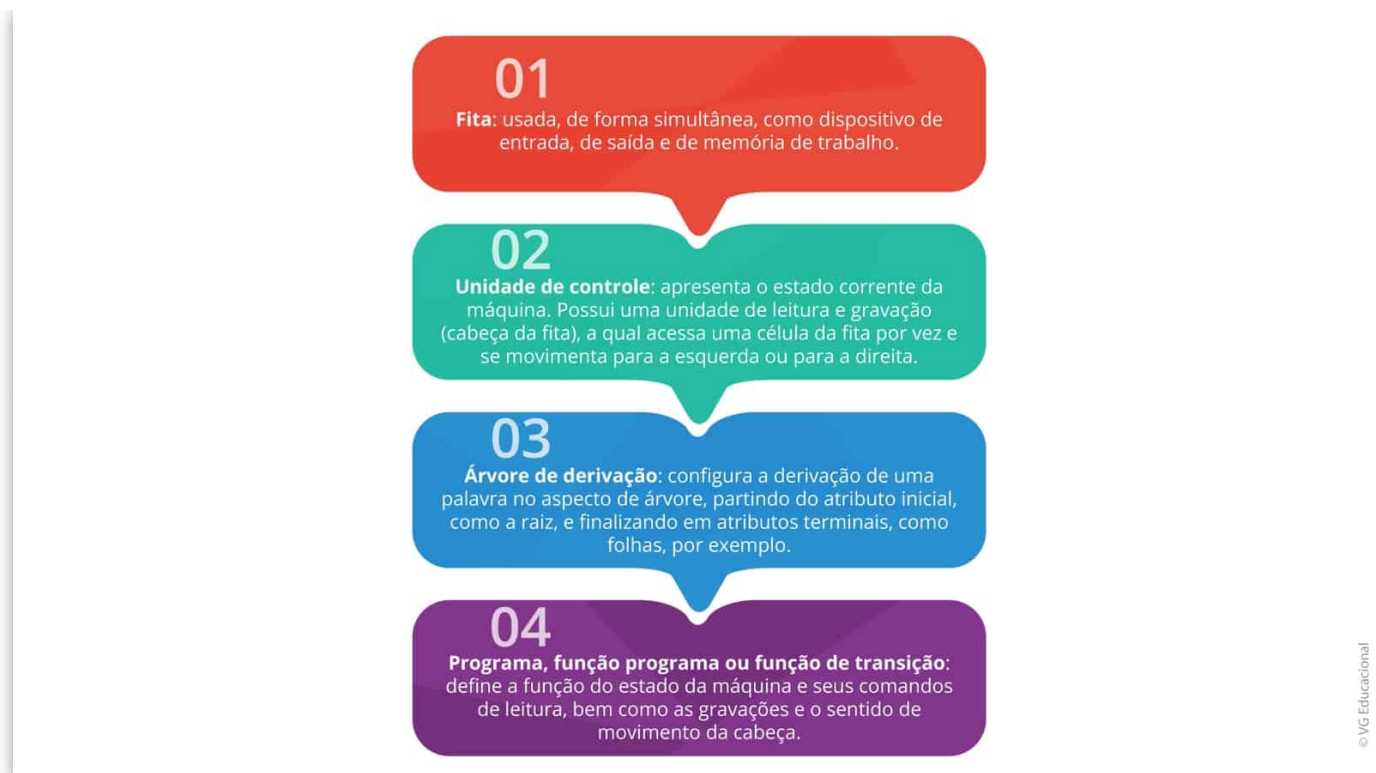
Inicialmente, a partir da suposição de que a folha de papel contém somente os dados iniciais para a apresentação do problema, o trabalho da pessoa se resume em sequências de operações simples, são elas:

1. ler um símbolo de um quadrado;
2. alterar um símbolo em um quadrado;
3. mover os olhos para outro quadrado.

Observe que, quando é encontrada alguma representação satisfatória no resultado, isto é, encontra-se a resposta desejada, a pessoa finaliza seus cálculos. Para viabilizar esse procedimento, as seguintes hipóteses são aceitáveis (MENEZES, 2015, p. 103):

1) a natureza bidimensional do papel não é um requisito essencial para os cálculos. Pode ser assumido que o papel consiste de uma fita infinita organizada em quadrados; 2) o conjunto de símbolos deve ser finito. Para representar informações mais complexas, é possível utilizar sequências de símbolos, analogamente à noção de alfabeto e linguagem natural como o português; 3) o conjunto de estados da mente da pessoa durante o processo de cálculo é finito. Mais ainda, entre esses estados, existem dois em particular: "estado inicial" e "estado final", correspondendo ao início e ao fim dos cálculos, respectivamente; 4) o comportamento da pessoa, a cada momento, é determinado somente pelo seu estado presente e pelo símbolo para o qual sua atenção está voltada; 5) a pessoa é capaz de observar e alterar o símbolo de apenas um quadrado de cada vez, bem como de transferir sua atenção para somente um dos quadrados adjacentes.

Dito isso, quanto ao modelo, podemos definir a noção de uma pessoa calculando do ponto de vista de uma máquina em três partes. No elemento a seguir, você pode conferir cada uma delas.



#PraCegoVer : o infográfico estático apresenta 4 (quatro) caixas de texto sequenciais nas cores vermelha, verde, azul e roxa. Na primeira caixa, a vermelha, de número 01, encontra-se o seguinte texto: “Fita: usada, de forma simultânea, como dispositivo de entrada, de saída e de memória de trabalho”. Na segunda caixa, de cor verde e número 02, encontra-se o seguinte texto: “Unidade de controle: apresenta o estado corrente da máquina. Possui uma unidade de leitura e gravação (cabeça da fita), a qual acessa uma célula da fita por vez e se movimenta para a esquerda ou para a direita”. Na terceira caixa, de cor azul e número 03, encontra-se o seguinte texto: “Árvore de derivação: configura a derivação de uma palavra no aspecto de árvore, partindo do atributo inicial, como a raiz, e finalizando em atributos terminais, como folhas, por exemplo”. Na quarta e última caixa, na cor roxa e número 04, encontra-se o seguinte texto: “Programa, função programa ou função de transição: define a função do estado da máquina e seus comandos de leitura, bem como as gravações e o sentido de movimento da cabeça”.

Medeiros (2015, p. 104), afirma que:

A fita é finita à esquerda e infinita (tão grande quanto necessário) à

direita, sendo dividida em células, cada uma armazenando um símbolo. Os símbolos podem:

- pertencer ao alfabeto de entrada;
- pertencer ao alfabeto auxiliar;
- ser “branco”;
- ser “marcador de início de fita”.

Inicialmente, a palavra a ser processada (ou seja, a informação de entrada para a máquina) ocupa as células mais à esquerda após o marcador de início de fita, ficando as demais com “branco”,

Você pode visualizar como isso funciona e entender melhor vendo a figura a seguir (os símbolos β e b representam “branco” e “marcador de início de fita”, respectivamente).

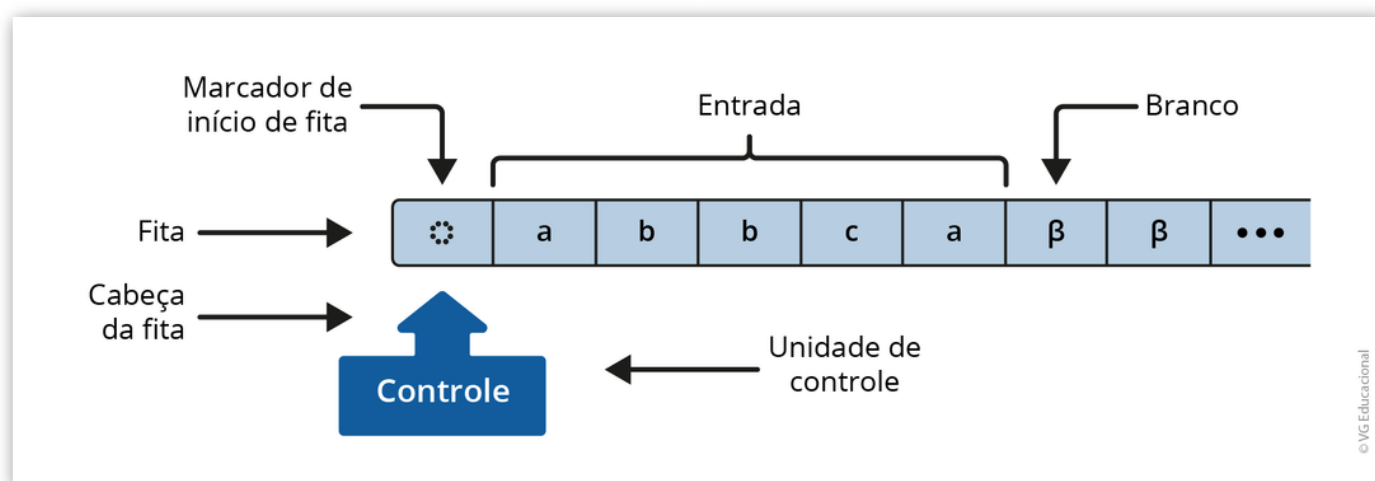


Figura 4.1 - Visão formal da máquina de Turing, composta pela fita e pela unidade de controle

Fonte: Medeiros (2015, p. 216).

#PraCegoVer : a imagem está dividida em dois blocos: o controle ou unidade de controle que aponta para a cabeça da fita; e a fita propriamente dita, com o marcador de início da fita, a entrada e o branco, com um número finito e predefinido de estados determinados pela unidade de controle.

A unidade de controle apresenta um número finito e predefinido de estados. A cabeça da fita tem a finalidade de ler o símbolo de uma célula, uma de cada vez, e gravar um novo símbolo por vez. Após a leitura/gravação (a gravação é

realizada na mesma célula de leitura), a cabeça moverá uma célula para a direita ou para a esquerda. O símbolo é, então, gravado e o sentido do movimento é definido pelo programa (DIVERIO, 2015).

Quanto à definição matemática das propriedades da máquina de Turing, temos que uma máquina de turing (M) é uma dupla com (MENEZES, 2015, p. 106):

$$M = (A, C, F, Q_0, S, L, M)$$

Respectivamente, as variáveis representam:

- 1) A - equivalente ao alfabeto com símbolos de entrada;
- 2) C - equivalente a um conjunto com estados finitos permitidos na máquina;
- 3) F - equivalente à função do programa que será executado na máquina, tal que a função é definida pelo conjunto C de estados finitos permitidos.

Logo, a transição de estados da máquina terá:

- 4) Q_0 - equivalente ao estado inicial da máquina;
- 5) S - equivalente ao conjunto dos estados finais da máquina;
- 6) L - equivalente ao alfabeto auxiliar, que pode estar vazio;
- 7) M - equivalente ao marcador de início da fita.

Segundo Menezes (2015),

*O símbolo que inicializa a fita sempre ocorrerá uma **vez** e será na **célula** mais à **esquerda** da fita, colaborando com a identificação de que a **cabeça da fita** está na célula mais à esquerda da fita. Por sua vez, a função programa tem por finalidade, considerar o estado corrente e o **símbolo** lido da fita com objetivo de apresentar um novo estado, o símbolo, por sua vez, a ser gravado e o sentido de movimento da cabeça, da esquerda e da direita será representados por E e D, respectivamente. A **função** programa pode ser*

interpretada através de **diagrama**, como ilustrado na figura a seguir (supondo a transição $\delta(p, x) = (q, y, m)$). Neste caso, os estados inicial e finais são representados como nos autômatos finitos (MENEZES, 2015, p. 106).

A computação de uma máquina de Turing M, para uma palavra de entrada w, tem como sucessiva a aplicação da função programa a partir do estado de início e da cabeça posicionada na célula mais à esquerda da fita, até ser apresentada uma condição de parada.

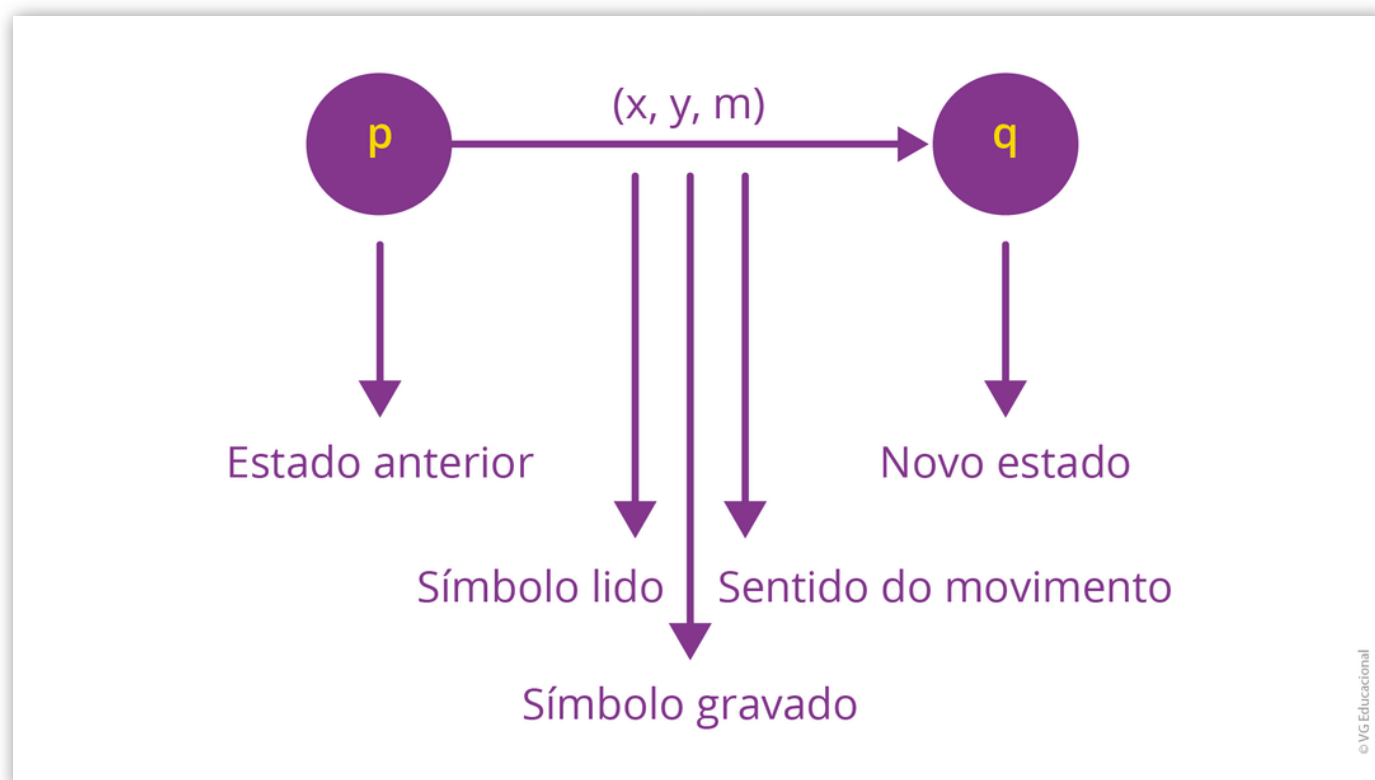


Figura 4.2 - Diagrama de transição da máquina de Turing

Fonte: Medeiros (2015, p. 217).

#PraCegoVer : a imagem está dividida em dois círculos, com uma seta da esquerda para a direita, isto é, (p) para (q), em função de (x,y,m), onde temos, no estado (p), o estado anterior, (x) o símbolo lido, (y) o símbolo gravado, (m) o sentido do movimento e q o novo estado representado.

Logo, o processamento de M para a entrada w pode parar ou ficar processando indefinidamente (ciclo ou loop infinito).

SAIBA MAIS

A parada do processamento de uma máquina de Turing para uma entrada w pode ser de duas maneiras:

1) aceita a entrada w . Atinge um estado final: a máquina para, e a palavra w é aceita;

2) Rejeita a entrada w . São duas possibilidades:

2.1 - a função programa é indefinida para o argumento (símbolo lido e estado corrente): a máquina para, e a palavra w é rejeitada;

2.2 - o argumento corrente da função programa define um movimento à esquerda, e a cabeça da fita já se encontra na célula mais à esquerda: a máquina para, e a palavra w é rejeitada.

Veja mais sobre as possibilidades de aceitação e funcionamento da máquina de Turing no vídeo:

ASSISTIR

Por fim, vimos, em diversos momentos, que a máquina de Turing é aceita como uma formalização do conceito de algoritmo. Entretanto, também é usual considerar que o conceito de algoritmo corresponde a uma máquina de Turing que sempre interrompe seu fluxo, para qualquer entrada. Nesse caso, uma máquina que eventualmente fica processando indefinidamente (em *loop* infinito) não seria considerada um algoritmo. Para finalizar este tópico, vamos

fixar o conhecimento por meio de uma atividade. Vamos lá?

Conhecimento

Teste seus Conhecimentos

(Atividade não pontuada)

Em 1936, Alan Turing propôs um formalismo para a representação de procedimentos efetivos. O trabalho de Turing é particularmente significativo para a teoria da computação, porque foi o primeiro a identificar programas escritos para uma “máquina computacional”, como noções intuitivas do efetivamente computável. Logo, temos que uma máquina de Turing é um autômato cuja fita não possui tamanho máximo e pode ser usada simultaneamente como dispositivo de entrada, de saída e de memória de trabalho.

DIVERIO. **A teoria da computação** : Máquina universais e computabilidade. 10. ed. Porto Alegre: Grupo Sagah, 2015.

Diante das definições apresentadas e dos estudos sobre a máquina de Turing e seu funcionamento, temos que, intuitivamente, um algoritmo que roda na máquina de Turing deve possuir pelo menos as seguintes características:

- ☐ **a)** Ter uma descrição infinita e ter passos executáveis com tempo finito.
- ☐ **b)** Ter uma descrição finita e passos executáveis com tempo infinito.

- **c)** Ter uma descrição infinita e não ter passos executáveis com tempo finito.
- **d)** Ter uma descrição finita e ter passos executáveis com tempo finito.
- **e)** Ter uma descrição finita e não ter passos executáveis com tempo finito.

Modelos Equivalentes à Máquina de Turing

Ao longo do tempo, vários modelos surgiram em paralelo à máquina de Turing, porém, todos, apesar das várias modificações, não se mostraram diferentes do próprio funcionamento da máquina de Turing. Logo, mudando-se as variáveis, observou-se que a máquina de Turing é o modelo mais basilar da teoria da computação.

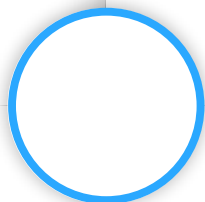
Outras máquinas equivalentes foram se desenvolvendo ao longo do tempo, dentre elas as que mais se destacaram na tentativa de reformular os preceitos básicos da máquina de turing, foram o

autômatos com múltiplas estruturas de pilhas, a máquina de turing não determinística, a máquina de turing com fita infinita à esquerda e à direita, a máquina de turing com múltiplas fitas, a máquina de turing multidimensional, a máquina de turing com múltiplas cabeças e combinações diversas com a máquina de turing (MENEZES, 2015, p. 106).

Caro(a) estudante, um dos motivos para considerar a máquina de Turing o modelo computacional mais basilar entre os dispositivos computacionais é o fato de que todos os demais modelos e máquinas propostos, bem como as diversas modificações da máquina de Turing existentes, possuem, no máximo, o mesmo poder computacional da máquina de Turing. Veja no elemento a seguir as modificações dos modelos.

Então, a partir das diferentes combinações que podemos ter de modelos equivalentes à máquina de Turing, fica explícita a importância desta como modelo fundamental, tendo em vista que, tradicionalmente, a diferença de capacidade nos modelos equivalentes não demonstra grandes divergências em relação à máquina de Turing, a qual permanece, assim, como modelo fundamental da teoria da computação.

SAIBA MAIS



Para saber mais sobre a máquina de Turing e a aplicabilidade da tese de Church, assista ao vídeo a seguir e perceba como o emprego da máquina de turing é fundamental para a teoria da computação.

ASSISTIR

A tese de Church, que veremos a seguir, pressupõe o seguinte: a capacidade de computação representada pela máquina de Turing é o limite máximo que pode ser atingido por qualquer dispositivo de computação. Façamos, agora, uma atividade para fortalecer nosso conhecimento sobre a máquina de Turing.

Conhecimento

Teste seus Conhecimentos

(Atividade não pontuada)

2) A máquina de Turing é o dispositivo de computação mais basilar e dele derivam outros modelos propostos. Nestes, apesar das várias modificações em relação à máquina de Turing, temos praticamente, no máximo, o mesmo poder computacional da máquina de Turing.

DIVERIO, T. A. **Teoria da computação** : Máquinas universais e computabilidade. Porto Alegre: Grupo A, 2015.

Dada uma máquina de Turing com função programa, dependendo do estado corrente da máquina e do símbolo lido em cada uma das fitas, ela gravará um novo símbolo em cada fita, assumindo um estado novo único. Assinale a alternativa correta que indica a qual modelo de máquina de Turing as características apresentadas se referem.

- ☐ **a)** Máquina de Turing com múltiplas fitas.
- ☐ **b)** Máquina de Turing com múltiplas cabeças.
- ☐ **c)** Máquina de Turing multidimensional.
- ☐ **d)** Máquina de Turing não determinística.
- ☐ **e)** Máquina de Turing com fita infinita.

Hipótese de Church

Vamos, agora, trabalhar a **hipótese de Church** ou modelo conceitual de Church. O modelo abstrato de computação proposto por Turing, em 1936, conhecido como máquina de Turing, tinha como objetivo **explorar** os limites da capacidade de expressar **soluções** de problemas.

Logo, trata-se de uma proposta de **definição** formal da noção **intuitiva** de algoritmo. Diversos outros trabalhos, como Cálculo Lambda e funções recursivas, resultaram em conceitos **equivalentes** ao da máquina Turing (DIVERIO, 2015). É latente que todos esses trabalhos que foram elaborados de forma independente geraram o mesmo resultado em termos de capacidade de expressar computabilidade, logo, são um forte reforço no que é conhecido como tese de Church ou tese de Turing-Church. A tese ou hipótese é a seguinte: “A capacidade de computação representada pela máquina de Turing é o limite máximo que pode ser atingido por qualquer dispositivo de computação” (DIVERIO, 2015, p. 79).

Assim, a hipótese de Church afirma que qualquer outra forma de expressar algoritmos terá, no máximo, a mesma capacidade computacional da máquina de Turing (MENEZES, 2015). Como a noção de algoritmo ou função computável é intuitiva, a hipótese de Church não se apresenta demonstrável.

Entretanto, como é assumida como verdadeira (ou seja, é assumida como hipótese) para toda ciência da computação, também é conhecida como hipótese de Church ou hipótese de Turing-Church.

Vejamos, agora, a aplicabilidade da respectiva hipótese, na máquina de Turing, como reconhecedor, com classe de linguagens recursivamente enumeráveis. No elemento a seguir, vamos conferir as seguintes classes de linguagens que são definidas a partir do formalismo máquina de Turing.

- 1) Classe das linguagens recursivamente enumeráveis:
composta pelas linguagens para as quais existe uma máquina

de Turing capaz de determinar se uma palavra w pertence à linguagem, ou seja, se $w \in L$. Entretanto, se $w \in \sim L$, o algoritmo pode:

- parar, identificando que a palavra não pertence à linguagem;
- ficar em loop infinito.

2) Classe das linguagens recursivas: composta pelas linguagens para as quais existe pelo menos uma máquina de Turing que sempre para, capaz de determinar se uma determinada palavra w pertence ou não à linguagem, ou seja, para uma dada linguagem L , existe uma máquina de Turing que determina se $w \in L$ ou $w \in \sim L$.

Lembrando que a existência dessas duas classes contradiz a intuição da maioria das pessoas, pois estabelece que reconhecer o complemento de uma linguagem pode ser impossível, mesmo que seja possível reconhecer a linguagem. Veja como isso se dá na figura a seguir:

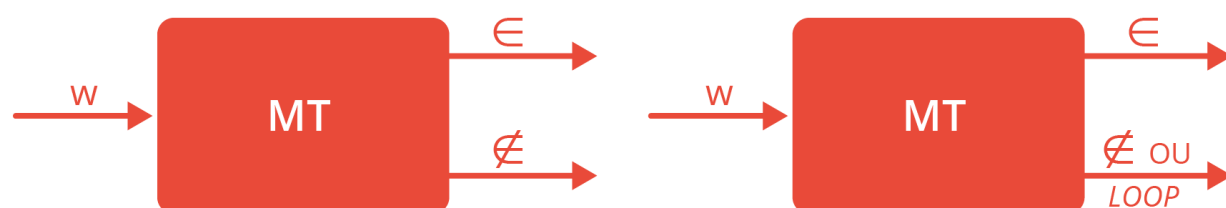


Figura 4.4 - Esquematização recursiva (esquerda) e recursivamente enumerável (direita) da máquina de Turing

Fonte: Menezes (2015, p. 223).

#PraCegoVer : na ilustração acima, temos a esquematização dos conceitos 1 e 2 que vimos anteriormente, isto é, à direita, temos a representação da classe das linguagens recursivamente enumeráveis e à esquerda temos a representação da classe das linguagens recursivas.

Por fim, vamos abordar um último tópico muito importante sobre a hipótese de Church e a máquina de Turing, que são as linguagens recursivas. Por definição, uma linguagem L é dita linguagem recursiva se existe pelo menos uma máquina de Turing M , tal que:

1) ACEITA (M) = L

2) REJEITA (M) = $\sim L$

Ou seja, uma linguagem é recursiva se existe uma máquina de Turing que aceita a linguagem e sempre para qualquer entrada.

Já no caso de uma gramática irrestrita ou uma linguagem recursivamente enumerável, temos que uma gramática irrestrita é simplesmente uma gramática sem qualquer restrição nas produções. Portanto, o termo “irrestrita” serve somente para destacar tal fato (DIVERIO, 2015). Já fizemos um bom progresso até aqui, não é mesmo? Então, está na hora de praticar um pouco.

praticar

Vamos Praticar

Considerando que, segundo a Hipótese de Church, a máquina de Turing é o dispositivo de computação mais geral, pode-se afirmar que a classe das linguagens recursivamente enumeráveis representa todas as linguagens que podem ser reconhecidas, isto é, compiladas mecanicamente.

Trata-se, portanto, de uma classe de linguagens muito rica. Entretanto, há conjuntos que não são recursivamente enumeráveis, ou seja, linguagens para as quais não é possível desenvolver uma máquina de Turing que as reconheça.

Comando da atividade prática: Ao analisar os estudos sobre a máquina de Turing e a hipótese de Church, apresente, baseado no teorema de Church, pelo menos, uma linguagem que não é recursivamente enumerável.

Decidibilidade

Finalizando o conteúdo deste material, temos o estudo sobre **decidibilidade**, que é um dos ramos da computabilidade. Vejamos, inicialmente, o estudo da computabilidade.

O estudo da computabilidade tem como **objetivo** determinar a solucionabilidade de problemas, a partir da existência de algoritmos. Portanto, investiga os limites do que pode ser implementado em um computador, evitando a pesquisa de soluções inexistentes.

A abordagem da computabilidade é centrada nos problemas de **decisão** (do tipo sim/não) e usa frequentemente o princípio da **redução**: investigação da solucionabilidade de um problema a partir de outro. Infelizmente, muitos dos problemas importantes são “não solucionáveis”, por exemplo, temos o detector universal de *loops*. De fato, informalmente falando, existem muito mais problemas não solucionáveis do que solucionáveis (MENEZES, 2015).

Lembrando que a existência dessas duas classes contradiz a intuição da maioria das pessoas, pois estabelece que reconhecer o complemento de uma linguagem pode ser impossível, mesmo que seja possível reconhecer a linguagem. Veja como isso se dá.

O objetivo do estudo da computabilidade é, então, analisar a possibilidade de resolução de problemas a partir de algoritmos computacionais, ou seja, investigar a existência ou não de algoritmos que solucionem determinada classe de problemas. Mais ainda: investigar os limites da computabilidade e, conseqüentemente, os limites do que pode efetivamente ser implementado em um computador. Logo, temos, então, uma questão de decidibilidade se desenhando. Por consequência, o estudo da solucionabilidade objetiva busca evitar a pesquisa de soluções inexistentes de problemas, ou seja, temos situações que, de fato, não têm resolução (DIVERIO, 2015).

Um exemplo de problema que não tinha solução, mas que foi estudado por muitos anos (cerca de 70 anos), foi um dos **23 problemas** propostos por David Hilbert, em Paris, no ano de 1900, durante o II Congresso Internacional de Matemática. Ele apresentou uma lista de 23 problemas ainda não solucionados que seriam o desafio para o novo século. Entre eles, consta como o décimo problema o seguinte enunciado: “Dada uma equação diofantina $P(a_0, \dots, a_n) = 0$, onde P é um polinômio com coeficientes inteiros, quer-se saber se existe um procedimento efetivo que seja capaz de determinar, em um tempo finito, se suas raízes são inteiras” (MENEZES, 2015, p. 124).

Sabe-se que as equações **diofantinas** são equações algébricas com coeficientes inteiros. Na situação proposta, buscava-se estabelecer **fundamentos** rigorosos para a aritmética, visando representar todas as leis científicas em equações matemáticas. O princípio era estabelecer a consistência formal da **aritmética** e, em seguida, estender às demais áreas do conhecimento.

Procurou-se definir uma linguagem puramente **sintática** (sem significado) denominada de sistema formal. Recaiu-se, então, no problema da existência ou não de um procedimento mecânico efetivo que decidisse a veracidade ou falsidade de qualquer enunciado aritmético.

Em 1928, David Hilbert propôs o problema de decisão, o qual questionava a existência de um **procedimento mecânico** (baseado no trabalho de Gottfried

Leibniz, que buscava mecanismo mecânico de manipulação de fórmulas) capaz de decidir se, dado um enunciado (proposição) da lógica de primeira ordem, ele seria válido ou não, em um tempo finito.

Esta ideia de considerar a matemática um sistema formal empolgou os matemáticos no século XIX. Pretendia-se obter uma teoria aritmética como um sistema formal consistente e completo. Mas isso, infelizmente, não foi possível, como afirmado por David Hilbert.

A título de ilustração, o teorema da não completude apresenta que todas as formulações axiomáticas consistentes da teoria dos números incluem proposições indecidíveis, ou seja, que não podem ser provadas como verdadeiras ou como falsas. Portanto, se um sistema formal é consistente, ele não pode ser completo, e a consistência dos axiomas não pode ser provada usando o próprio sistema formal (DIVERIO, 2015, p. 88).

A resolução do problema de decisão começou a apresentar resultados em 1936, quando Alonzo Church provou que não existia nenhum **algoritmo** definido por meio de funções recursivas que decidisse se, para duas expressões dadas em cálculo Lambda, elas seriam ou não equivalentes. Alan Turing também contribuiu para a resolução do problema de **decisão**, transformando-o em um problema da parada em sua máquina.

Na resolução do problema de decidibilidade, uma boa parte do sucesso se deu pelo uso da técnica da **redução** (princípio da redução), na qual partes de um problema são **transformadas** em outro, o que permite usar resultados já conhecidos para obter a resolução de partes do problema atual e, assim, resolvê-lo (DIVERIO, 2015).

A solução definitiva do problema de decisão foi apresentada em 1970, pelo matemático Yuri Matiyasevich, o qual provou que tal procedimento efetivo (algoritmo) não existe. A prova é complexa, envolvendo teoria dos números e resultados anteriores sobre esse problema. Basicamente, tem-se:

Se R é uma relação que pode ser codificada em um número natural, então, R é

uma equação diofantina se e somente se R é enumerável. O resultado disso é que o conjunto das equações diofantinas com raízes inteiras não é solucionável, isto é, não é decidível.

Abarcando, então, os problemas de **decidibilidade** em específico, ou problemas de decisão, estes são tratados como problemas de decisão ou do tipo sim/não. A ideia básica é verificar se a **função** associada a eles é ou não **computável** em uma máquina universal. Como, pela hipótese de Church, uma máquina universal é o dispositivo mais geral de computação, se a solução de um problema não puder ser expressa por uma máquina universal, então, tal problema é completamente insolúvel (MENEZES, 2015).

A essência por trás de um problema de decisão é dada pelo seguinte pressuposto: dado um programa P para uma máquina universal M , decidir se a função computada $\langle P, M \rangle$ é total (ou seja, se a correspondente computação é finita).

A existência de problemas não solucionáveis é fundamental por diversas razões, dentre elas demonstrar limitações da capacidade de se expressar soluções por meio de softwares ou de algoritmos computacionais. Outra razão é, que a existência de um problema não solucionável pode ser usada para verificar que outros problemas também o são. Isso é possível usando o princípio da redução, o qual consiste em “reduzir” o problema que se está investigando, em outro problema que já se saiba ser não solucionável. A investigação da solucionabilidade de problemas em máquinas universais pode ser verificada como um problema de reconhecimento de linguagens (MENEZES, 2015, p. 110).

Lembre-se de que alguns dos formalismos estudados, como a máquina de Turing, constituem, de fato, um programa para uma máquina. É importante reparar que o que está sendo discutido são métodos gerais. Portanto, é perfeitamente possível que existam métodos específicos para programas particulares.

praticar

Vamos Praticar

Vamos demonstrar o funcionamento de uma codificação bijetora de programas, dado que:

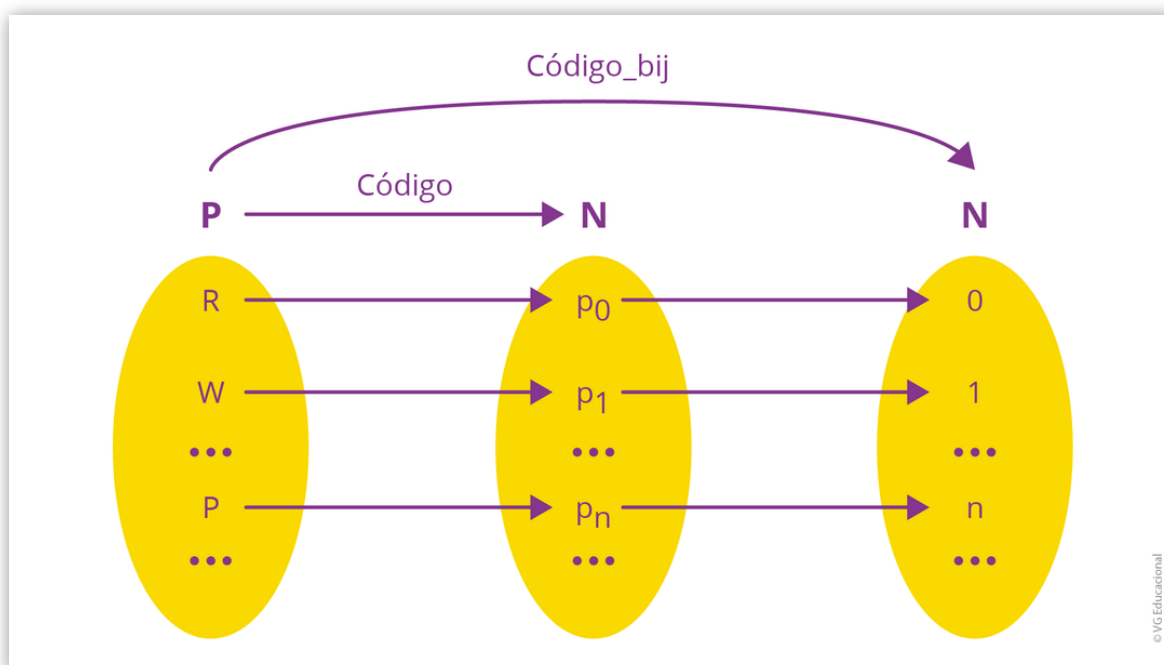


Figura - Ilustração da codificação bijetora de um programa

Fonte: Menezes (2015, p. 133).

#PraCegoVer : na ilustração acima, apresentamos a função bijetora de um programa, em que o código transita entre conjuntos de estados diferentes, com início em P , que é o que temos como código recursivo, o que, por conseguinte, perpassa a função N para derivar os estados 0 ou 1 , de forma iterativa.

Sendo P o conjunto de todos os programas do tipo que está sendo considerado (monolítico, iterativo ou recursivo), considere a sequência de inteiros p_0, p_1, p_2, \dots formada pelos códigos dos programas de P (ordenados pela relação “menor” sobre os naturais). Então:

$\text{código_bij}: P \rightarrow \mathbb{N}$

onde, para qualquer $P \in P$, se o código de P é p_n , então: $\text{código_bij}(P) = n$

Nesse caso, explique o funcionamento e as associações realizadas no programa, a partir da função bijetora $\text{código_bij}(P) = n$.

Material Complementar



FILME

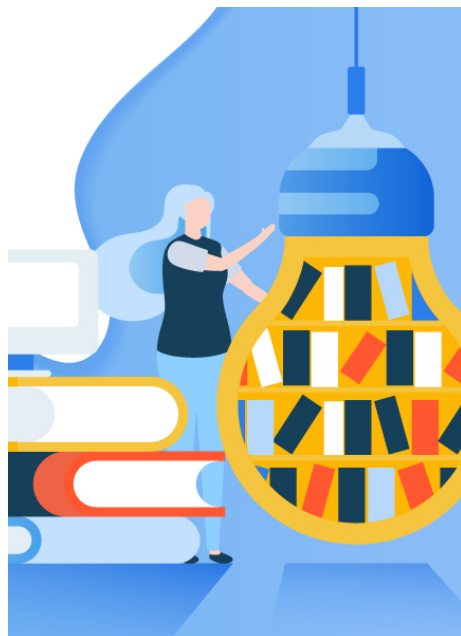
Pirates of Silicon Valley (Piratas da Informática)

Ano: 1999

Comentário: o filme retrata os bastidores da corrida travada entre Bill Gates e Steve Jobs no desenvolvimento do computador pessoal (PC), e são demonstrados os fundamentos da elaboração das linguagens de programação e os desafios da época para conseguir escalar a produção de computadores pessoais e sua respectiva distribuição em todo o mundo. Vale a pena assistir ao filme para compreender como os conceitos apresentados ao longo deste estudo foram fundamentais para criar as duas maiores empresas globais de tecnologia da história.

Para conhecer mais sobre o filme, assista ao trailer disponível em:

TRAILER



LIVRO

Linguagens formais e autômatos

Paulo Menezes

Editora: Grupo A

ISBN: 8577807657

Comentário: este livro é o principal referencial sobre autômatos aplicados à teoria da computação. Em especial, trata, em seu último tópico, sobre a máquina de Turing e as especificidades que estudamos ao longo do estudo. Vale a pena se aprofundar através do livro, tendo em vista os vários exemplos práticos que são propostos.

Caro(a) estudante, chegamos ao fim do material e, após a leitura, podemos concluir a importância dos tópicos aqui apresentados, isto é, definição, modelo e propriedades da máquina de Turing, modelos equivalentes à máquina de Turing, hipóteses de Church e, por último, a definição por trás dos problemas de decidibilidade ou problemas de decisão, presentes na teoria da computação e explorados a partir dos conceitos computacionais que aprendemos.

Demonstramos, claramente, a relevância do assunto para a teoria da computação, para o desenvolvimento de novas linguagens de programação e, por que não dizer, para o desenvolvimento de novos paradigmas de programação, tendo em vista que, a cada novo ano, novos paradigmas e modelos arquiteturais surgem, fomentando, assim, novas aplicações a partir dos conceitos que estudamos ao longo do material. Dessa forma, é fundamental se debruçar nos estudos dos respectivos tópicos, a fim de melhor compreender o futuro das linguagens de programação.

Referências

MENEZES, P. F. B. **Linguagens formais e autômatos** . Porto Alegre: Grupo A, 2015.

DIVERIO, T. A. **Teoria da computação : máquinas universais e computabilidade**. Porto Alegre: Grupo A, 2015.





TRAILER | Piratas da Informática |
Piratas do Vale do Silício (1999)
Legendado. [S. l.: s. n.], 2020. 1 vídeo (2
min 1 s). Publicado pelo canal Clique
Trailers. Disponível em:
[https://www.youtube.com
/watch?v=qma8YVHev0c](https://www.youtube.com/watch?v=qma8YVHev0c). Acesso em:
17 jun. 2021.