Tese de Church-Turing e MT Universal

Prof. Rafael S. Durelli

rafael.durelli@dcc.ufla.br

Departamento de Ciências da Computação

Universidade Federal de Lavras

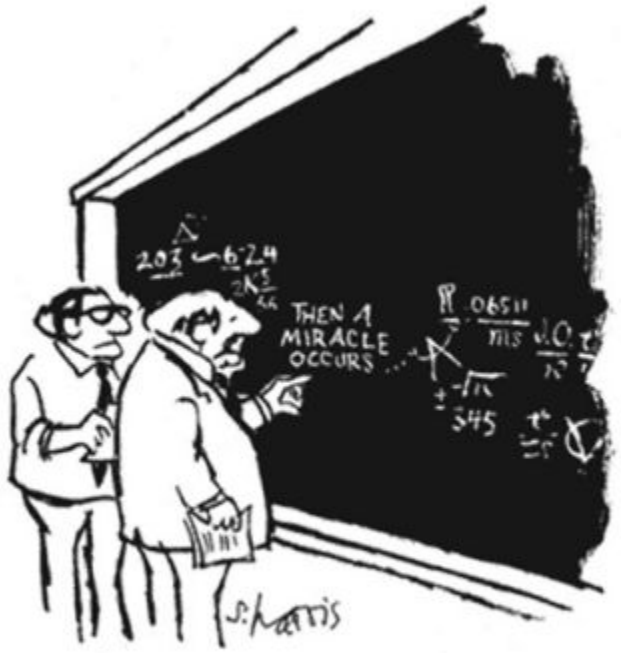
CIÊNCIA DA
COMPUTAÇÃO

Origem do termo *algoritmo*

- Utilizado por um matemático árabe (Abu Ja'far Muhammad ibn Musa al-Khwarizmi) que viveu no século IX
  - Em um livro, apresentou um conjunto de regras para resolver equações lineares e quadráticas
- Os limites de algoritmos têm sido foco de estudos mais a partir do século XX
  - Transformação de strings
    - Sistemas de Post (Post, 1936)
    - Sistemas Markov (Markov, 1961)
  - Avaliação de funções
    - Funções parciais e u-recursivas (Gödel, 1931; Kleene, 1936)
    - Cálculo lambda (Church, 1941)
  - Máquinas de computação abstrata
    - Máquinas de registradores (Shepherson, 1963)
    - Máquinas de Turing
  - Linguagens de Programação
    - Programas While (Kfoury et al., 1982)

Origem do termo *algoritmo*

Noção intuitiva de algoritmo: é um conjunto finito de instruções que podem ser executadas mecanicamente em tempo finito para resolver algum problema. Com dados de entrada apropriados ao problema, o algoritmo tem que parar e produzir a resposta correta.



"I THINK YOU SHOULD BE MORE EXPLICIT HERE IN STEP TWO."

Programas while

- É o uso de linguagens de programação mínimas
  - Compostas apenas de comandos de
    - **atribuição**,
    - **Condicionais** e
    - **Repetição**

Máquinas de Turing

- É uma escolha padrão na análise da computabilidade, mas outros ’sistemas’ poderiam ser escolhidos

- A computabilidade é uma característica do problema e não do sistema de verificação

A Tese de Church-Turing valida essa intuição

Máquinas de Turing

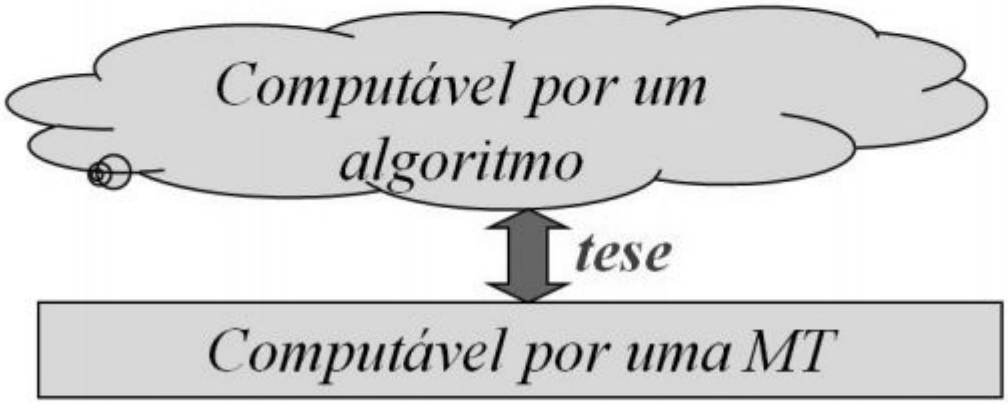
- Como esses sistemas são equivalentes, acredita-se que eles definem os limites da computação algorítmica

Tese de Church Turing

- De acordo com a tese de Church–Turing, se um cálculo puder ser feito de forma automatizada — por um dado método, num número finito de passos — então também pode ser feito por uma máquina de Turing.

Tese de Church Turing

Tese de Church-Turing (1936):
"Qualquer função de teoria dos números é computável por um algoritmo se, e somente se, for computável por uma Máquina de Turing."



Tese de Church Turing

Para problemas de decisão:

- Há um procedimento efetivo (algoritmo) para resolver um problema de decisão se, e somente se, há uma máquina de Turing que pára para todas as cadeias de entrada e que resolve o problema

Tese de Church Turing

Solução parcial para um problema de decisão

- Uma solução parcial não é necessariamente completa, mas que retorne a resposta **sim** para qualquer instância do problema que deva ser positiva. Se a resposta for **não**, a máquina pode responder não ou falhar na produção da resposta

Tese de Church Turing

Para problemas de Reconhecimento

- Um problema de decisão é parcialmente resolvível se, e somente se, há uma máquina de Turing que aceita precisamente instâncias do problema cujas respostas sejam **sim**

Tese de Church Turing

- Mesmo que a máquina de Turing seja utilizada para a computação de funções (abordagem funcional), o retorno da computação pode ser um ’1’ ou ’0’ na fita, ao final da computação, para problemas de decisão.
- Isso aumenta a formulação da tese de Church-Turing em termos de funções computáveis

Tese de Church Turing

- ***Para computação de função***
- Uma função *f* é efetivamente computável se, e somente se, há uma máquina de Turing que computa *f*

Aqui vem a visitação da Tese de Church Turing depois da definição das funções μ-recursivas

Tese de Church Turing

- Não é uma formulação matemática
  - Não pode ser provada
  - Necessitaria de uma definição formal de algoritmo
- Para ’desacreditar’ a tese seria necessário um procedimento que pudesse ser computável e para o qual não se conseguisse construir uma máquina de Turing
- A equivalência entre a MT e os outros sistemas algorítmicos e a falta de um contra-exemplo são fortes evidências de que a tese se mantém

Tese de Church Turing

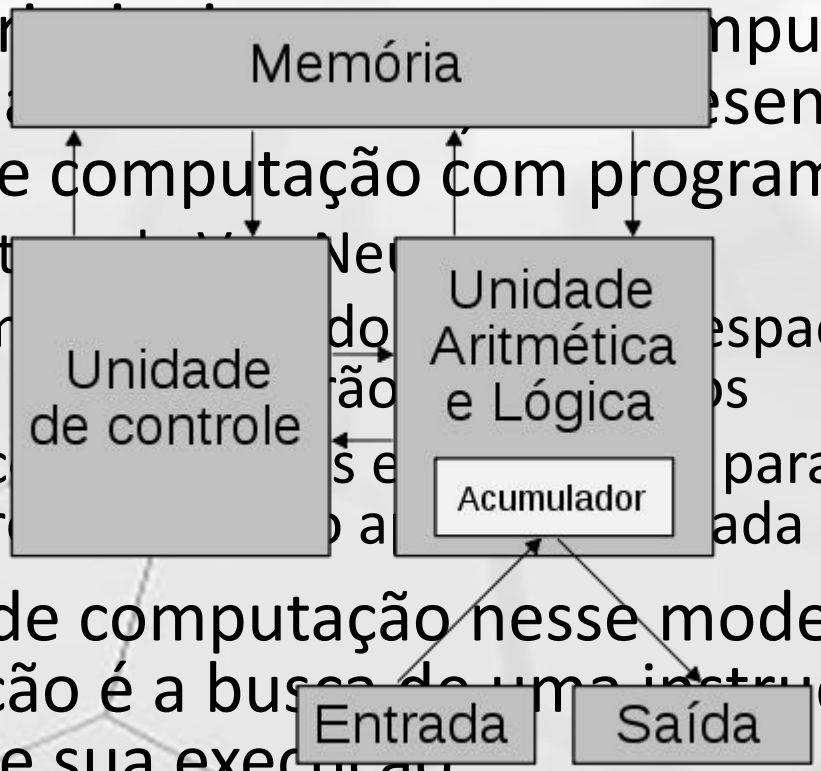
- A prova pela Tese de Church-Turing é um atalho frequentemente utilizado na decisão da existência de um algoritmo de decisão.
  - Ao invés de construir uma máquina de Turing, explicitamente, define-se um procedimento efetivo que resolva o problema
  - A tese garante que se pode construir uma MT que resolva o problema
  - Algumas máquinas de Turing são muito complexas…

Máquina de Turing Universal

- Um dos principais avanços na computação, em meados da década de 40, foi o desenvolvimento do modelo de computação com programa armazenado
  - Arquitetura de Von Neumann
  - Programa armazenado no mesmo espaço de memória em que os dados serão manipulados
  - Antes, computadores eram criados para executar apenas uma tarefa, variando apenas a entrada
- Um ciclo de computação nesse modelo de computação é a busca de uma instrução em memória e sua execução

Máquina de Turing Universal

- Um dos pr… …nputação, em meados d… …esenvolvimento do modelo de computação com programa armazenado
  - Arquitet… …Ne…
  - Program… …do… …espaço de memória em que … …rão… …os
  - Antes, c… …s e… … para executar apenas uma tar… …o a… …ada
- Um ciclo de computação nesse modelo de computação é a busca de uma instrução em memória e sua execução

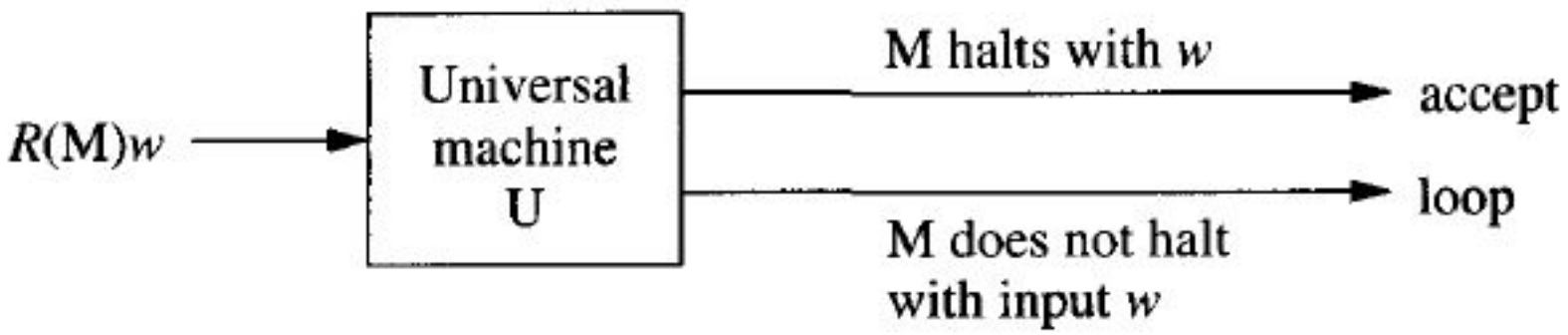


Máquina de Turing Universal

- As máquinas de Turing até agora são como os primeiros computadores, feitas para executar apenas uma tarefa
- Mas podemos arquitetar uma MT que siga o conceito de programa armazenado: **Máquina de Turing Universal**
- Utilizada para simular as computações de uma máquina de Turing qualquer

Máquina de Turing Universal (MTU)

- A entrada de uma *MTU* é uma representação de uma máquina de Turing *M* e a cadeia *w* a ser processada por *M*



Máquina de Turing Universal (MTU)

- Primeiro passo na construção de uma MTU: definir a cadeia de representação de uma máquina de Turing M
  - Dado que podemos codificar qualquer símbolo com o uso de cadeias sobre {0,1}, escolhemos máquina de Turing com esse mesmo alfabeto de entrada ({0,1}) e alfabeto da fita como {0,1,B}
  - Os estados de M são tidos como $\{q_0,q_1,\ldots,q_n\}$, sendo q_0 o estado inicial

Máquina de Turing Universal (MTU)

- M é definida por sua função de transição
- Uma transição tem a forma

$$\delta(q_i, x) = [q_j, y, d]$$

  - Em que q_i e $q_j \in Q$; $x, y \in \Gamma$; e $d \in \{L, R\}$
- Codificamos os elementos de M, utilizando strings de 1's

Máquina de Turing Universal (MTU)

- M é
- Uma
  - E
- Cod ings de 1

Símbolo	Codificação
0	1
1	11
B	111
q_0	1
q_1	11
…	…
q_n	1^{n+1}
L	1
R	11

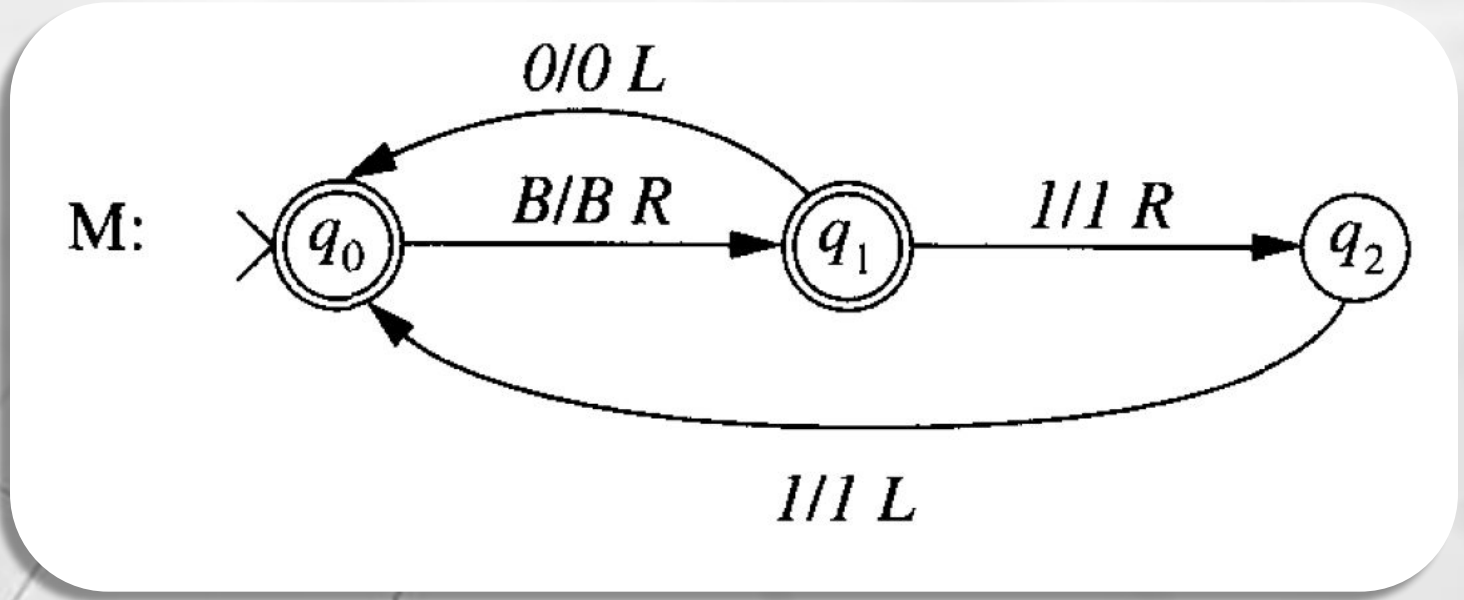
Máquina de Turing Universal (MTU)

- Seja *en(z)* a codificação do símbolo *z,* a transição $\delta(q_i, x) = [q_j, y, d]$ é codificada como:

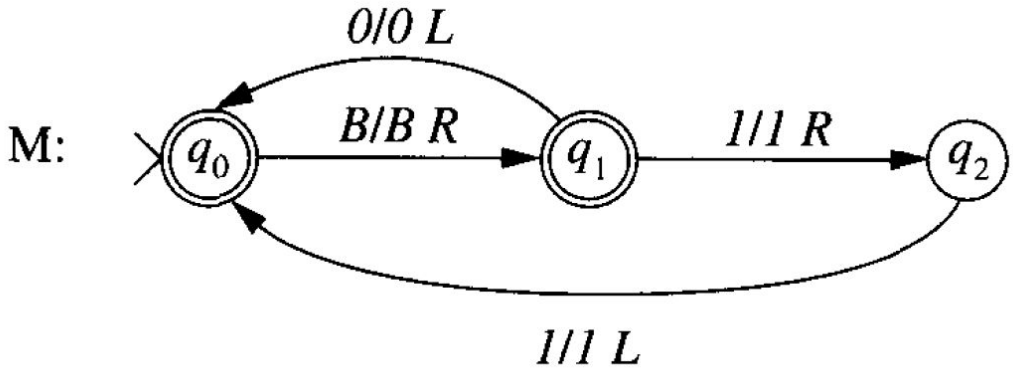
$$en(q_i)0en(x)0en(q_j)0en(y)0en(d)$$

- Os 0's separam os componentes da transição
- A representação de M é feita codificando-se suas transições
- Dois 0's consecutivos separam transições diferentes
- O início e fim de uma representação utiliza três 0's consecutivos

Exemplo 1



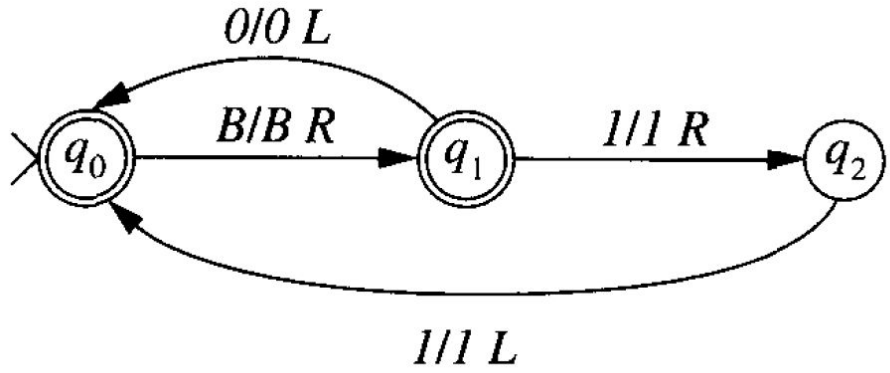
Exemplo 1



Transition	Encoding
$\delta(q_0, B) = [q_1, B, R]$	*10111011011011*
$\delta(q_1, 0) = [q_0, 0, L]$	*1101010101*
$\delta(q_1, 1) = [q_2, 1, R]$	*11011011011011*
$\delta(q_2, 1) = [q_0, 1, L]$	*11101101011101*

Exemplo 1

M:



0001011101101111011001101010101001101101110110110011101101011010000.

Transition	Encoding
$\delta(q_0, B) = [q_1, B, R]$	*1011101101111011*
$\delta(q_1, 0) = [q_0, 0, L]$	*1101010101*
$\delta(q_1, 1) = [q_2, 1, R]$	*1101101110111011*
$\delta(q_2, 1) = [q_0, 1, L]$	*1110110101101*

Máquina de Turing Universal

•

Descreva o que uma máquina de Turing deve fazer para verificar se uma determina entrada $u \in \{0,1\}*$ codifica corretamente uma máquina de Turing determinística

Máquina de Turing Universal

- Descreva o que uma máquina de Turing deve fazer para verificar se uma determina entrada $u \in \{0,1\}*$ codifica corretamente uma máquina de Turing determinística

- Iniciar e terminar com *000*
- Verificar um conjunto de instruções separada por *00*
  - Padrão $en(q_i)0en(x)0en(q_j)0en(y)0en(d)$
- Se combinação de estado e símbolo de entrada em cada transição for distinto: máquina determinística

Máquina de Turing Universal

Vamos criar uma máquina de Turing *U* com 3 fitas:

- A fita 1 inicia a com entrada, que tem uma cadeia na forma *R(M)w*
- A computação de *M* é simulada na fita 3

Máquina de Turing Universal (U)

U tem as seguintes ações:

1. Se a entrada não tem a forma R(M)w, U move para a direita indefinidamente
2. w é escrita no início da fita 3 e sua cabeça retorna para o início
3. Um 1 é escrito na fita 2, representando o estado inicial q_0
4. Uma transição de M é simulada na fita 3. A transição de M é determinada pelo símbolo lido na fita 3 e o estado codificado na fita 2

Máquina de Turing Universal (U)

4. Uma transição de *M* é simulada na fita 3. A transição de *M* é determinada pelo símbolo lido na fita 3 e o estado codificado na fita 2. Seja *x* o símbolo lido na fita 3 e q_i o estado codificado na fita 2:
   a. A fita 1 é lida em busca de uma transição que tenha o primeiro componente equivalente a en(q_i) e en(x). Caso não haja essa transição, U pára e rejeita a entrada
   b. Se a fita 1 contém uma transição *en(q_i)0en(x)0en(q_j)0en(y)0en(d), então*
      i. *en(q_i)* é substituído por *en(q_j)* na fita 2
      ii. O símbolo y é escrito na fita 3
      iii. A cabeça da fita 3 é movida na direção especificada por *d*
5. A computação continua com o passo 4, para simular a próxima transição de M

Referências

- Sudkamp, T. A. (2006). Languages and machines: an introduction to the theory of computer science. 3rd Edition.
  - Capítulo 11: Decision Problems and the Church-Turing Thesis

Teorema

- A linguagem $L_H = \{R(M)w \mid M\ pára\ com\ w\}$ é recursivamente enumerável

Prova:

A máquina universal *U* aceita cadeias da forma *R(M)w*, em que *R(M)* é a representação de *M* e *M* pára com a entrada *w*. Para todas outras cadeias de entrada, a computação de *U* não pára. Então a linguagem de *U* é L_H

Teorema

- A linguagem $L_H = \{R(M)w \mid M\ pára\ com\ w\}$ é recursivamente enumerável

- L_H é conhecida como a linguagem do Problema da Parada.
- Uma cadeia está em L_H se é a combinação de uma representação de uma máquina de Turing e uma cadeia *w*, tal que *M* pára quando executa sobre *w*

Exemplo 2

Problema: *parar na n-ésima transição*

Entrada: *M*, *w* e inteiro *n*

Saída: *sim*, caso a computação de *M*, lendo *w*, pára exatamente após *n* transições; *não*, caso contrário.

Exemplo 2

Parar na n-ésima transição

Solução: simular a computação de M com w e contar a quantidade de transições

- Utilizamos uma U' adicionando uma quarta fita a U
  - Na fita 4, será armazenada a contagem de transições simuladas por U'
- A entrada *u* desse problema é representada por $R(M)w0001^{n+1}$

Exemplo 2

Parar na n-ésima transição

1. Se *u* não terminar com 0001^{n+1}, *U'* pára e rejeita *u*
2. A cadeia 1^n é escrita na fita 4 e 0001^{n+1} é apagado do final de *u* na fita 1. A cabeça da fita 4 volta para o início
3. Se a cadeia restante na fita 1 não tem a forma *R(M)w*, *U'* pára e rejeita a entrada
4. A cadeia *w* é copiada para a fita 3 e a codificação de q_0 é escrita na fita 2
5. A estratégia de *U* é utilizada: a fita 1 é lida em busca de uma transição que tenha o símbolo *x* lido da fita 3 e estado q_i, codificado na fita 2

Exemplo 2

Parar na n-ésima transição

5. A estratégia de *U* é utilizada: a fita 1 é lida em busca de uma transição que tenha o símbolo *x* lido da fita 3 e estado q_i, codificado na fita 2
   a) Se não houver transição para q_i e *x* e um *1* for lido na fita 4, então *U'* pára e rejeita a entrada
   b) Se não houver transição para q_i e *x* e um *B* é lido na fita 4, então *U'* pára e aceita a entrada
   c) Se houver transição para q_i e *x* e um *B* é lido na fita 4, então *U'* pára e rejeita a entrada
   d) Se houver transição para q_i e *x* e um *1* é lido na fita 4, então a transição é simulada nas fitas 2 e 3 e a cabeça da fita 4 é movida uma célula para a direita
6. A computação continua com o passo 5, para a próxima transição de M