

Introdução à Teoria da Computação

A Tese de Church-Turing

Professor Murilo V. G. da Silva

Departamento de Informática
Universidade Federal do Paraná

2026 / 2

Perspectiva Histórica

Perspectiva Histórica

Início do século XX: matemáticos notaram que a demonstração de teoremas se assemelha a um procedimento mecânico.

Início do século XX: matemáticos notaram que a demonstração de teoremas se assemelha a um procedimento mecânico.

- [David Hilbert \(1928\)](#): queria um “procedimento efetivo” que decida se uma afirmação matemática é verdadeira ou falsa.

Início do século XX: matemáticos notaram que a demonstração de teoremas se assemelha a um procedimento mecânico.

- [David Hilbert \(1928\)](#): queria um “procedimento efetivo” que decida se uma afirmação matemática é verdadeira ou falsa. (queria um algoritmo para o *Entscheidungsproblem*)

Perspectiva Histórica

Início do século XX: matemáticos notaram que a demonstração de teoremas se assemelha a um procedimento mecânico.

- **David Hilbert (1928)**: queria um “procedimento efetivo” que decida se uma afirmação matemática é verdadeira ou falsa. (queria um algoritmo para o *Entscheidungsproblem*)
- **Kurt Gödel (1931)**: provou que nem todo teorema verdadeiro pode ser provado dentro de sistemas consistentes e suficientemente expressivos.

Perspectiva Histórica

Início do século XX: matemáticos notaram que a demonstração de teoremas se assemelha a um procedimento mecânico.

- **David Hilbert (1928)**: queria um “procedimento efetivo” que decida se uma afirmação matemática é verdadeira ou falsa. (queria um algoritmo para o *Entscheidungsproblem*)
- **Kurt Gödel (1931)**: provou que nem todo teorema verdadeiro pode ser provado dentro de sistemas consistentes e suficientemente expressivos.

Mas, pelo menos, existe um algoritmo capaz de decidir se há uma prova?

Início do século XX: matemáticos notaram que a demonstração de teoremas se assemelha a um procedimento mecânico.

- **David Hilbert (1928)**: queria um “procedimento efetivo” que decida se uma afirmação matemática é verdadeira ou falsa. (queria um algoritmo para o *Entscheidungsproblem*)
- **Kurt Gödel (1931)**: provou que nem todo teorema verdadeiro pode ser provado dentro de sistemas consistentes e suficientemente expressivos.

Mas, pelo menos, existe um algoritmo capaz de decidir se há uma prova?

- **Alonzo Church e Alan Turing (1936)**: provaram que não existe tal algoritmo.

Perspectiva Histórica

Início do século XX: matemáticos notaram que a demonstração de teoremas se assemelha a um procedimento mecânico.

- [David Hilbert \(1928\)](#): queria um “procedimento efetivo” que decida se uma afirmação matemática é verdadeira ou falsa. (queria um algoritmo para o *Entscheidungsproblem*)
- [Kurt Gödel \(1931\)](#): provou que nem todo teorema verdadeiro pode ser provado dentro de sistemas consistentes e suficientemente expressivos.

Mas, pelo menos, existe um algoritmo capaz de decidir se há uma prova?

- [Alonzo Church](#) e [Alan Turing \(1936\)](#): provaram que não existe tal algoritmo. Ambos formalizaram o conceito de algoritmo.

Início do século XX: matemáticos notaram que a demonstração de teoremas se assemelha a um procedimento mecânico.

- [David Hilbert \(1928\)](#): queria um “procedimento efetivo” que decida se uma afirmação matemática é verdadeira ou falsa. (queria um algoritmo para o *Entscheidungsproblem*)
- [Kurt Gödel \(1931\)](#): provou que nem todo teorema verdadeiro pode ser provado dentro de sistemas consistentes e suficientemente expressivos.

Mas, pelo menos, existe um algoritmo capaz de decidir se há uma prova?

- [Alonzo Church](#) e [Alan Turing \(1936\)](#): provaram que não existe tal algoritmo. Ambos formalizaram o conceito de algoritmo.
- Turing ainda introduziu a propriedade de *universalidade*: computadores de propósito geral podem executar algoritmos como entrada.

Início do século XX: matemáticos notaram que a demonstração de teoremas se assemelha a um procedimento mecânico.

- **David Hilbert (1928)**: queria um “procedimento efetivo” que decida se uma afirmação matemática é verdadeira ou falsa. (queria um algoritmo para o *Entscheidungsproblem*)
- **Kurt Gödel (1931)**: provou que nem todo teorema verdadeiro pode ser provado dentro de sistemas consistentes e suficientemente expressivos.

Mas, pelo menos, existe um algoritmo capaz de decidir se há uma prova?

- **Alonzo Church e Alan Turing (1936)**: provaram que não existe tal algoritmo. Ambos formalizaram o conceito de algoritmo.
- Turing ainda introduziu a propriedade de *universalidade*: computadores de propósito geral podem executar algoritmos como entrada.
- Vamos discutir a tese de Máquinas de Turing (e modelos equivalentes), de fato, correspondem ao que queremos dizer por algoritmo.

Perspectiva Histórica

Início do século XX: matemáticos notaram que a demonstração de teoremas se assemelha a um procedimento mecânico.

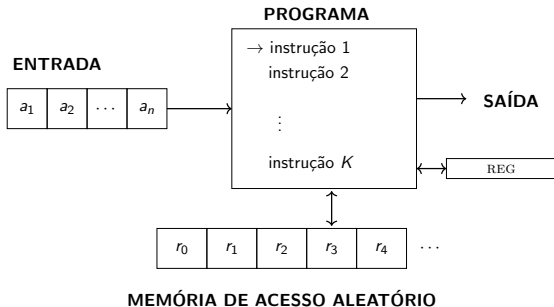
- **David Hilbert (1928)**: queria um “procedimento efetivo” que decida se uma afirmação matemática é verdadeira ou falsa. (queria um algoritmo para o *Entscheidungsproblem*)
- **Kurt Gödel (1931)**: provou que nem todo teorema verdadeiro pode ser provado dentro de sistemas consistentes e suficientemente expressivos.

Mas, pelo menos, existe um algoritmo capaz de decidir se há uma prova?

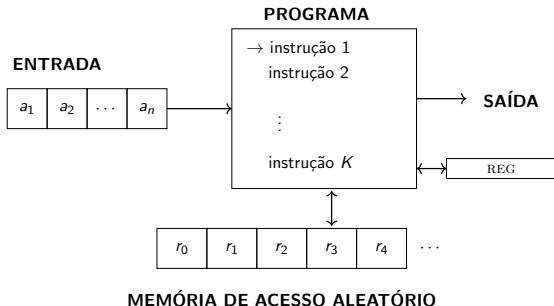
- **Alonzo Church e Alan Turing (1936)**: provaram que não existe tal algoritmo. Ambos formalizaram o conceito de algoritmo.
- Turing ainda introduziu a propriedade de *universalidade*: computadores de propósito geral podem executar algoritmos como entrada.
- Vamos discutir a tese de Máquinas de Turing (e modelos equivalentes), de fato, correspondem ao que queremos dizer por algoritmo.

⇒ Antes disso, vamos provar que MTs são capazes de expressar programas em assembly

O Modelo RAM – Programas Assembly

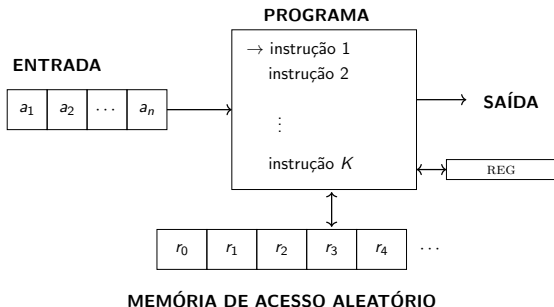


O Modelo RAM – Programas Assembly



Exemplo: Programa para somar dois números

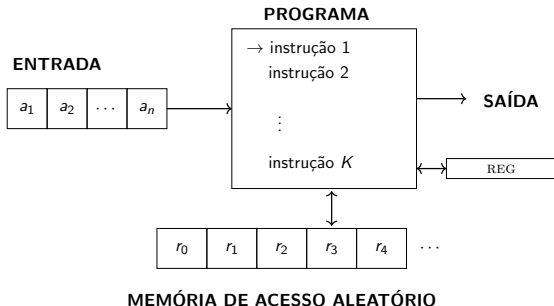
O Modelo RAM – Programas Assembly



Exemplo: Programa para somar dois números

1. READ
2. STORE ^1
3. READ
4. ADD ^1
5. STORE ^1
6. HALT

O Modelo RAM – Programas Assembly

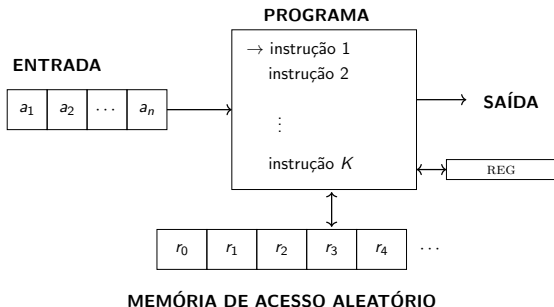


Exemplo: Programa para somar dois números

1. READ
2. STORE ^1
3. READ
4. ADD ^1
5. STORE ^1
6. HALT

Lê número inteiro a_1 da entrada e carrega em REG

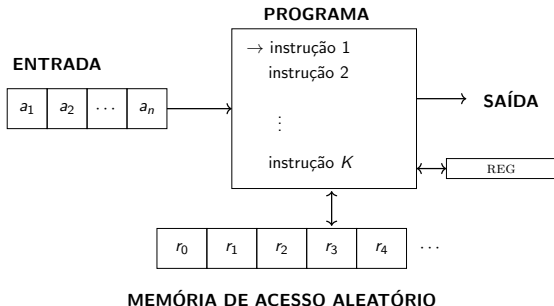
O Modelo RAM – Programas Assembly



Exemplo: Programa para somar dois números

- | | | | |
|----|-------|------------|--|
| 1. | READ | | Lê número inteiro a_1 da entrada e carrega em REG |
| 2. | STORE | $\wedge 1$ | Armazena o número de REG na posição de memória r_1 |
| 3. | READ | | |
| 4. | ADD | $\wedge 1$ | |
| 5. | STORE | $\wedge 1$ | |
| 6. | HALT | | |

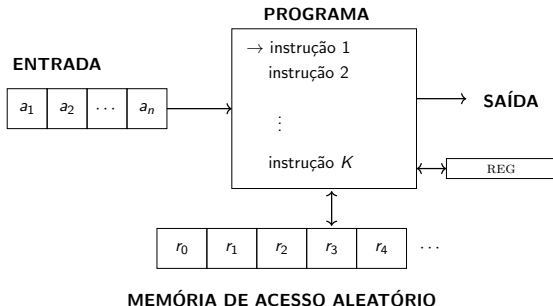
O Modelo RAM – Programas Assembly



Exemplo: Programa para somar dois números

- | | | | |
|----|-------|------------|--|
| 1. | READ | | Lê número inteiro a_1 da entrada e carrega em REG |
| 2. | STORE | $\wedge 1$ | Armazena o número de REG na posição de memória r_1 |
| 3. | READ | | Lê número inteiro a_2 da entrada e carrega em REG |
| 4. | ADD | $\wedge 1$ | |
| 5. | STORE | $\wedge 1$ | |
| 6. | HALT | | |

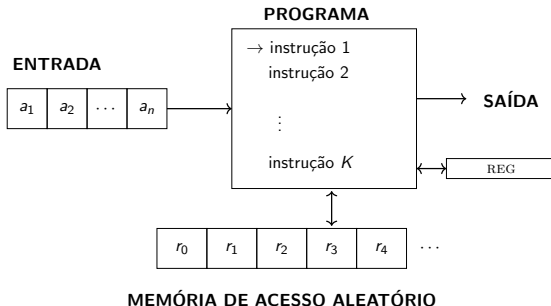
O Modelo RAM – Programas Assembly



Exemplo: Programa para somar dois números

- | | | | |
|----|-------|------------|--|
| 1. | READ | | Lê número inteiro a_1 da entrada e carrega em REG |
| 2. | STORE | $\wedge 1$ | Armazena o número de REG na posição de memória r_1 |
| 3. | READ | | Lê número inteiro a_2 da entrada e carrega em REG |
| 4. | ADD | $\wedge 1$ | Soma número em REG com número em r_1 ; |
| 5. | STORE | $\wedge 1$ | |
| 6. | HALT | | |

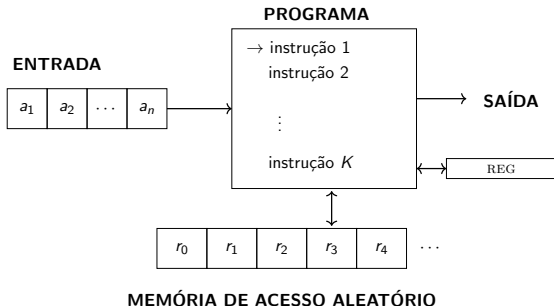
O Modelo RAM – Programas Assembly



Exemplo: Programa para somar dois números

- | | | | |
|----|-------|------------|--|
| 1. | READ | | Lê número inteiro a_1 da entrada e carrega em REG |
| 2. | STORE | $\wedge 1$ | Armazena o número de REG na posição de memória r_1 |
| 3. | READ | | Lê número inteiro a_2 da entrada e carrega em REG |
| 4. | ADD | $\wedge 1$ | Soma número em REG com número em r_1 ; Resultado fica em REG |
| 5. | STORE | $\wedge 1$ | |
| 6. | HALT | | |

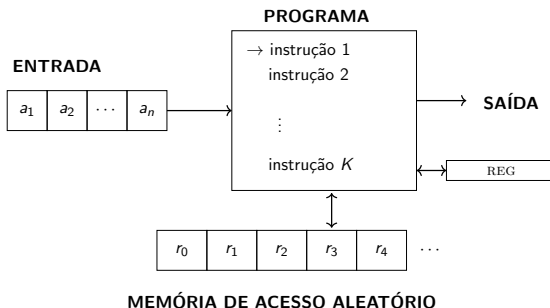
O Modelo RAM – Programas Assembly



Exemplo: Programa para somar dois números

- | | | | |
|----|-------|------------|--|
| 1. | READ | | Lê número inteiro a_1 da entrada e carrega em REG |
| 2. | STORE | $\wedge 1$ | Armazena o número de REG na posição de memória r_1 |
| 3. | READ | | Lê número inteiro a_2 da entrada e carrega em REG |
| 4. | ADD | $\wedge 1$ | Soma número em REG com número em r_1 ; Resultado fica em REG |
| 5. | STORE | $\wedge 1$ | Armazena valor de REG em r_1 |
| 6. | HALT | | |

O Modelo RAM – Programas Assembly



Exemplo: Programa para somar dois números

- | | | | |
|----|-------|------------|--|
| 1. | READ | | Lê número inteiro a_1 da entrada e carrega em REG |
| 2. | STORE | $\wedge 1$ | Armazena o número de REG na posição de memória r_1 |
| 3. | READ | | Lê número inteiro a_2 da entrada e carrega em REG |
| 4. | ADD | $\wedge 1$ | Soma número em REG com número em r_1 ; Resultado fica em REG |
| 5. | STORE | $\wedge 1$ | Armazena valor de REG em r_1 |
| 6. | HALT | | Para a execução |

Modelo RAM – Semântica completa

READ		A i -ésima chamada desta instrução carrega em REG o valor a_i da entrada.
WRITE		Escreve o conteúdo de REG na saída.
LOAD	i	Carrega o inteiro i em REG.
LOAD	$\wedge i$	Carrega o conteúdo de r_i em REG.
STORE	$\wedge i$	Armazena o conteúdo de REG no registrador r_i da memória.
ADD	i	Seja $v(\text{REG})$ o valor contido em REG. Depois da instrução, o registrador REG passa a conter o valor $v(\text{REG}) + i$.
ADD	$\wedge i$	Sejam $v(\text{REG})$ e $v(r_i)$ os valores em REG e r_i , respectivamente. Depois da instrução, o valor em REG passa a ser $v(\text{REG}) + v(r_i)$.
SUB	i	Seja $v(\text{REG})$ o valor contido em REG. Depois da instrução, o registrador REG passa a conter o valor $v(\text{REG}) - i$.
SUB	$\wedge i$	Sejam $v(\text{REG})$ e $v(r_i)$ os valores em REG e r_i , respectivamente. Depois da instrução, o valor em REG passa a ser $v(\text{REG}) - v(r_i)$.
MULT	i	Seja $v(\text{REG})$ o valor contido em REG. Depois da instrução, o registrador REG passa a conter o valor $v(\text{REG}) \times i$.
MULT	$\wedge i$	Sejam $v(\text{REG})$ e $v(r_i)$ os valores em REG e r_i , respectivamente. Depois da instrução, o valor em REG passa a ser $v(\text{REG}) \times v(r_i)$.
JUMP	i	Muda o valor do contador de programa para i .
JZERO	i	Muda o contador de programa para i , caso o conteúdo de REG seja zero.
JPOS	i	Muda o contador de programa para i , caso o conteúdo de REG seja positivo.
JNEG	i	Muda o contador de programa para i , caso o conteúdo de REG seja negativo.
HALT		Interrompe a execução do programa. (ver livro)

Modelo RAM

Exemplo: Programa que lê três números a, b, c e verifica se $a + b = c$.

Modelo RAM

Exemplo: Programa que lê três números a, b, c e verifica se $a + b = c$.

- Se $a + b = c$, escreve 1 na saída; caso contrário, escreve 0.

Modelo RAM

Exemplo: Programa que lê três números a, b, c e verifica se $a + b = c$.

- Se $a + b = c$, escreve 1 na saída; caso contrário, escreve 0.

```
1.  READ
2.  STORE  ~1
3.  READ
4.  ADD    ~1
5.  STORE  ~1
6.  READ
7.  SUB    ~1
8.  JZERO  11
9.  WRITE  0
10. HALT
11. WRITE  1
12. HALT
```

Simulando instruções usando MTs de 5 fitas

Vamos simular uma Máquina RAM usando uma MT de 5 fitas.

Simulando instruções usando MTs de 5 fitas

Vamos simular uma Máquina RAM usando uma MT de 5 fitas.

Fita 1: Entrada;

Simulando instruções usando MTs de 5 fitas

Vamos simular uma Máquina RAM usando uma MT de 5 fitas.

Fita 1: Entrada;

Fita 2: Memória

Simulando instruções usando MTs de 5 fitas

Vamos simular uma Máquina RAM usando uma MT de 5 fitas.

Fita 1: Entrada;

Fita 2: Memória (registradores $r_1, r_2, \dots$ em uso);

Simulando instruções usando MTs de 5 fitas

Vamos simular uma Máquina RAM usando uma MT de 5 fitas.

Fita 1: Entrada;

Fita 2: Memória (registradores $r_1, r_2, \dots$ em uso);

Fita 3: Registrador REG;

Simulando instruções usando MTs de 5 fitas

Vamos simular uma Máquina RAM usando uma MT de 5 fitas.

Fita 1: Entrada;

Fita 2: Memória (registradores $r_1, r_2, \dots$ em uso);

Fita 3: Registrador REG;

Fita 4: *Contador de programa*;

Simulando instruções usando MTs de 5 fitas

Vamos simular uma Máquina RAM usando uma MT de 5 fitas.

Fita 1: Entrada;

Fita 2: Memória (registradores $r_1, r_2, \dots$ em uso);

Fita 3: Registrador REG;

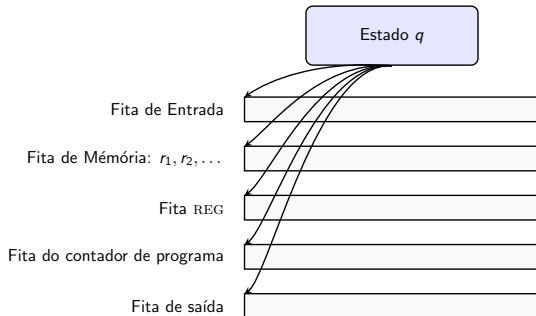
Fita 4: *Contador de programa*;

Fita 5: Saída;

Simulando instruções usando MTs de 5 fitas

Vamos simular uma Máquina RAM usando uma MT de 5 fitas.

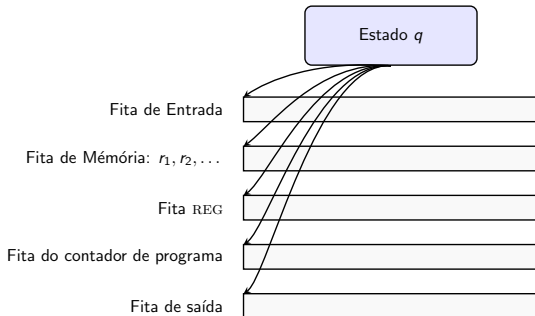
- Fita 1: Entrada;
- Fita 2: Memória (registradores $r_1, r_2, \dots$ em uso);
- Fita 3: Registrador REG;
- Fita 4: *Contador de programa*;
- Fita 5: Saída;



Simulando instruções usando MTs de 5 fitas

Vamos simular uma Máquina RAM usando uma MT de 5 fitas.

- Fita 1: Entrada;
- Fita 2: Memória (registradores $r_1, r_2, \dots$ em uso);
- Fita 3: Registrador REG;
- Fita 4: *Contador de programa*;
- Fita 5: Saída;



- Exemplo: Digamos que na memória r_1 contenha 7 e r_3 contenha 4:

Simulando instruções usando MTs de 5 fitas

Vamos simular uma Máquina RAM usando uma MT de 5 fitas.

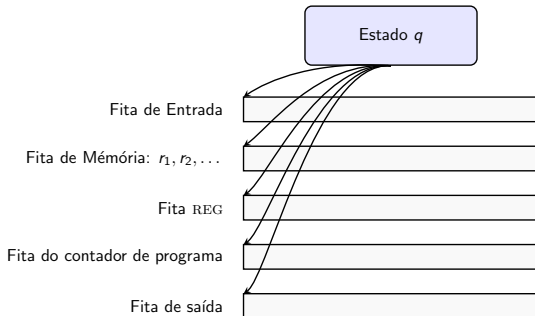
Fita 1: Entrada;

Fita 2: Memória (registradores $r_1, r_2, \dots$ em uso);

Fita 3: Registrador REG;

Fita 4: *Contador de programa*;

Fita 5: Saída;

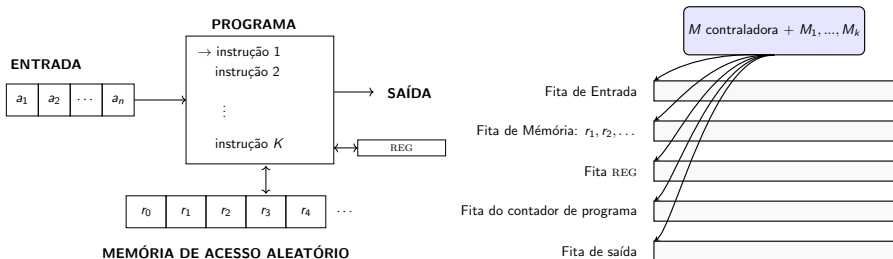


- Exemplo: Digamos que na memória r_1 contenha 7 e r_3 contenha 4:

Neste caso a fita 2 contém $\sqcup 1 \sqcup \# \sqcup 7 \sqcup \# \sqcup 3 \sqcup \# \sqcup 4 \sqcup$

Simulando instruções usando MTs de 5 fitas

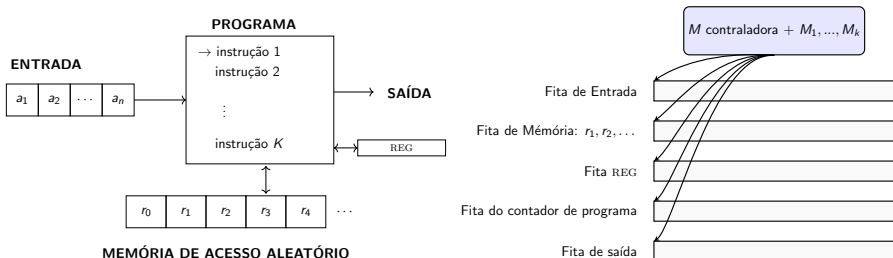
Simulando instruções usando MTs de 5 fitas



Exercício 1:

(a) Simular a instrução ADD^1 :

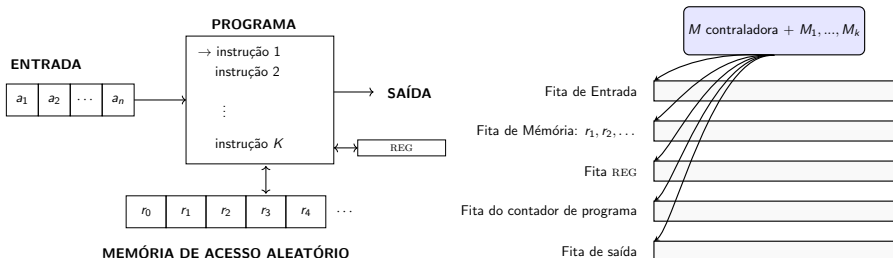
Simulando instruções usando MTs de 5 fitas



Exercício 1:

(a) Simular a instrução ADD^1 : soma conteúdo da fita REG ao conteúdo de r_3

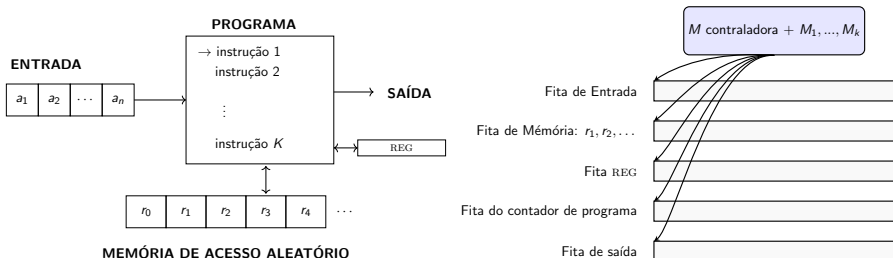
Simulando instruções usando MTs de 5 fitas



Exercício 1:

- (a) Simular a instrução ADD^1 : soma conteúdo da fita REG ao conteúdo de r_3 (memória),

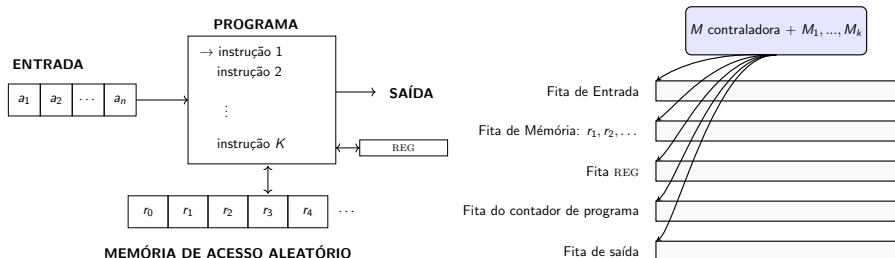
Simulando instruções usando MTs de 5 fitas



Exercício 1:

- (a) Simular a instrução ADD^1 : soma conteúdo da fita REG ao conteúdo de r_3 (memória), guarda o resultado na fita REG ,

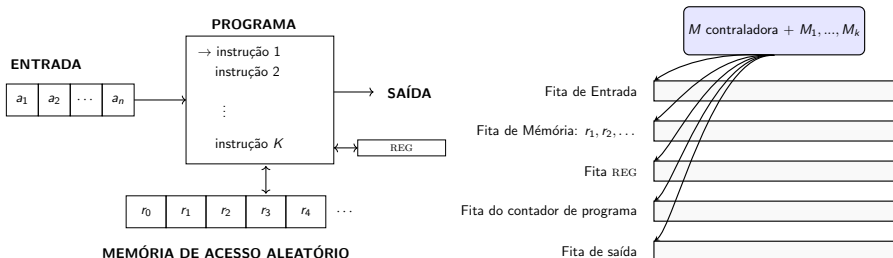
Simulando instruções usando MTs de 5 fitas



Exercício 1:

- (a) Simular a instrução ADD^1 : soma conteúdo da fita REG ao conteúdo de r_3 (memória), guarda o resultado na fita REG, e termina sem alteração nas demais fitas,

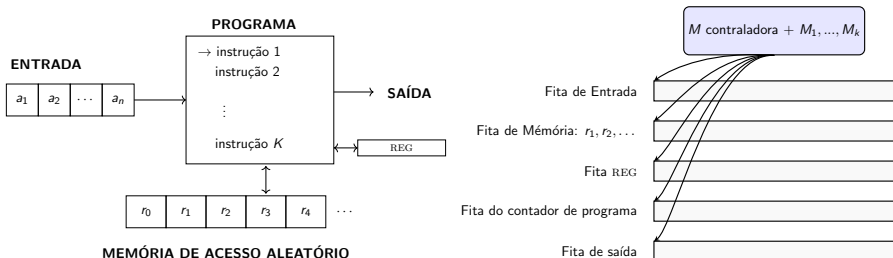
Simulando instruções usando MTs de 5 fitas



Exercício 1:

- (a) Simular a instrução ADD^1 : soma conteúdo da fita REG ao conteúdo de r_3 (memória), guarda o resultado na fita REG, e termina sem alteração nas demais fitas,
- (b) Simular a instrução READ:

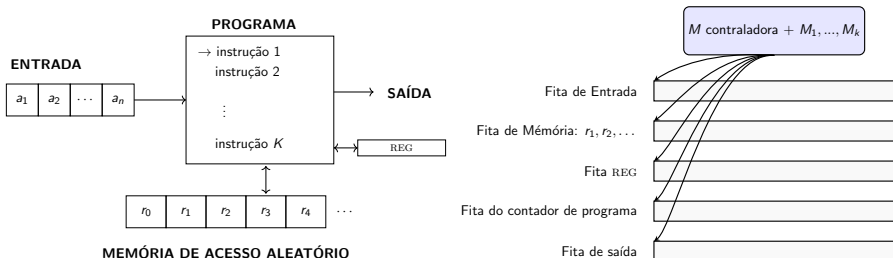
Simulando instruções usando MTs de 5 fitas



Exercício 1:

- Simular a instrução `ADD^1`: soma conteúdo da fita `REG` ao conteúdo de r_3 (memória), guarda o resultado na fita `REG`, e termina sem alteração nas demais fitas,
- Simular a instrução `READ`: lê o conteúdo da fita de entrada e armazena na fita `REG`

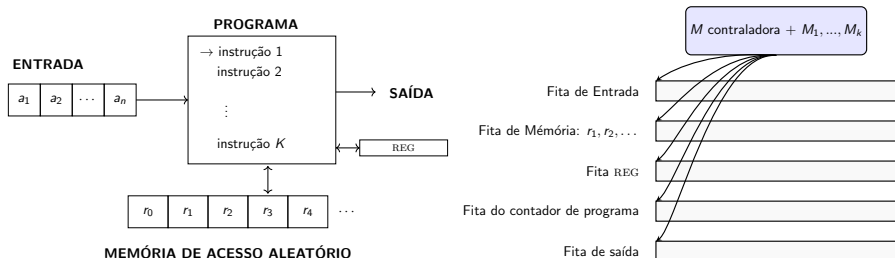
Simulando instruções usando MTs de 5 fitas



Exercício 1:

- (a) Simular a instrução ADD^1 : soma conteúdo da fita REG ao conteúdo de r_3 (memória), guarda o resultado na fita REG, e termina sem alteração nas demais fitas,
- (b) Simular a instrução READ: lê o conteúdo da fita de entrada e armazena na fita REG guarda o resultado na fita REG,

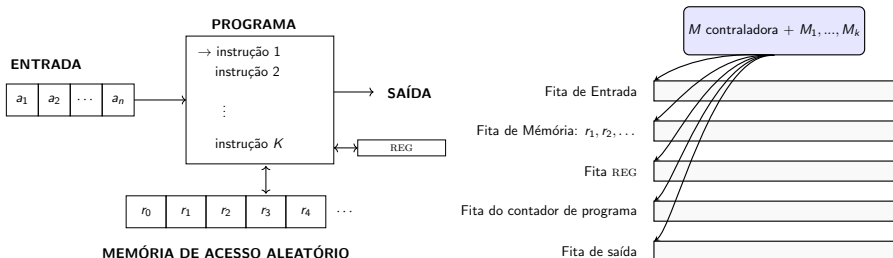
Simulando instruções usando MTs de 5 fitas



Exercício 1:

- (a) Simular a instrução ADD^1 : soma conteúdo da fita REG ao conteúdo de r_3 (memória), guarda o resultado na fita REG, e termina sem alteração nas demais fitas,
- (b) Simular a instrução READ: lê o conteúdo da fita de entrada e armazena na fita REG guarda o resultado na fita REG, e termina sem alteração nas demais fitas,
- (c) Simular a instrução STORE:

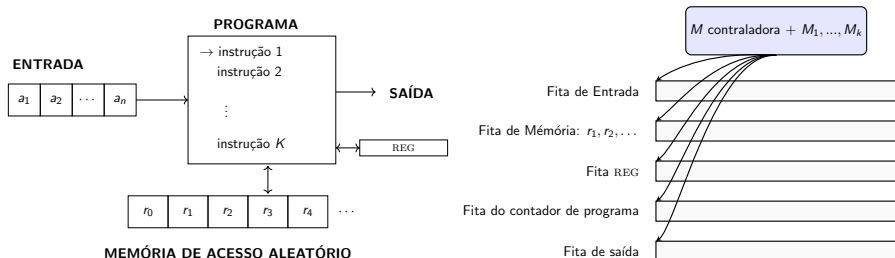
Simulando instruções usando MTs de 5 fitas



Exercício 1:

- (a) Simular a instrução ADD^1 : soma conteúdo da fita REG ao conteúdo de r_3 (memória), guarda o resultado na fita REG, e termina sem alteração nas demais fitas,
- (b) Simular a instrução READ: lê o conteúdo da fita de entrada e armazena na fita REG, guarda o resultado na fita REG, e termina sem alteração nas demais fitas,
- (c) Simular a instrução STORE: lê o conteúdo da REG e armazena na posição 1 da fita de memória

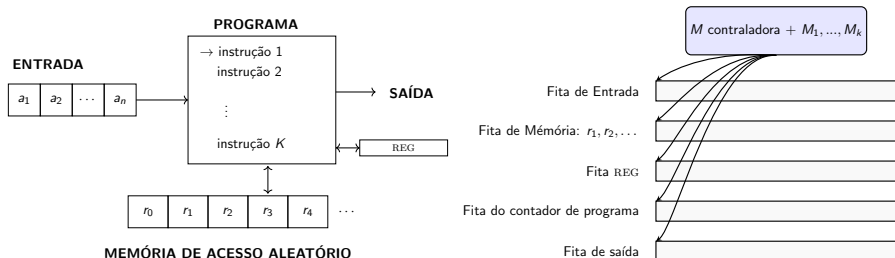
Simulando instruções usando MTs de 5 fitas



Exercício 1:

- Simular a instrução ADD^1 : soma conteúdo da fita REG ao conteúdo de r_3 (memória), guarda o resultado na fita REG, e termina sem alteração nas demais fitas,
- Simular a instrução READ: lê o conteúdo da fita de entrada e armazena na fita REG, guarda o resultado na fita REG, e termina sem alteração nas demais fitas,
- Simular a instrução STORE: lê o conteúdo da REG e armazena na posição 1 da fita de memória e termina sem alteração nas demais fitas,
- Simular a instrução HALT:

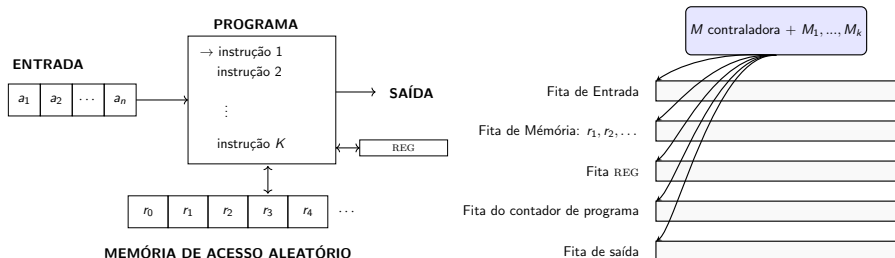
Simulando instruções usando MTs de 5 fitas



Exercício 1:

- (a) Simular a instrução ADD^1 : soma conteúdo da fita REG ao conteúdo de r_3 (memória), guarda o resultado na fita REG, e termina sem alteração nas demais fitas,
- (b) Simular a instrução READ: lê o conteúdo da fita de entrada e armazena na fita REG, guarda o resultado na fita REG, e termina sem alteração nas demais fitas,
- (c) Simular a instrução STORE: lê o conteúdo da REG e armazena na posição 1 da fita de memória e termina sem alteração nas demais fitas,
- (d) Simular a instrução HALT: para a execução

Simulando instruções usando MTs de 5 fitas



Exercício 1:

- (a) Simular a instrução ADD^1 : soma conteúdo da fita REG ao conteúdo de r_3 (memória), guarda o resultado na fita REG, e termina sem alteração nas demais fitas,
- (b) Simular a instrução READ: lê o conteúdo da fita de entrada e armazena na fita REG guarda o resultado na fita REG, e termina sem alteração nas demais fitas,
- (c) Simular a instrução STORE: lê o conteúdo da REG e armazena na posição 1 da fita de memória e termina sem alteração nas demais fitas,
- (d) Simular a instrução HALT: para a execução

Exercício 2: Converta o programa RAM para somar dois números para máquina de Turing.

O Modelo RAM – Programas Assembly

Solução do Exercício 2: Considere o programa abaixo para somar dois números:

```
1.  READ
2.  STORE  ^1
3.  READ
4.  ADD    ^1
5.  STORE  ^1
6.  HALT
```

O Modelo RAM – Programas Assembly

Solução do Exercício 2: Considere o programa abaixo para somar dois números:

1. READ
2. STORE ^1
3. READ
4. ADD ^1
5. STORE ^1
6. HALT

Segue o esquema de simulação:

O Modelo RAM – Programas Assembly

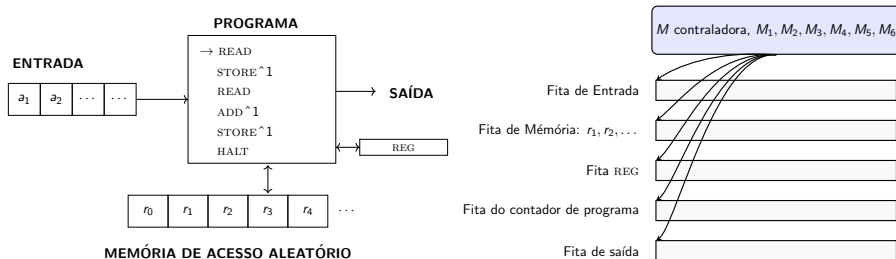
Solução do Exercício 2: Considere o programa abaixo para somar dois números:

- ```

1. READ
2. STORE ^1
3. READ
4. ADD ^1
5. STORE ^1
6. HALT

```

Segue o esquema de simulação:



# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Esboço da Prova:** Vamos criar Máquinas de Turing distintas, cada uma delas com cinco fitas, que executam cada uma das instruções assembly.

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Esboço da Prova:** Vamos criar Máquinas de Turing distintas, cada uma delas com cinco fitas, que executam cada uma das instruções assembly. Mais precisamente,

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Esboço da Prova:** Vamos criar Máquinas de Turing distintas, cada uma delas com cinco fitas, que executam cada uma das instruções assembly. Mais precisamente,

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Esboço da Prova:** Vamos criar Máquinas de Turing distintas, cada uma delas com cinco fitas, que executam cada uma das instruções assembly. Mais precisamente,

- A partir de  $\Pi = \pi_1, \pi_2, \dots, \pi_k$ , primeiramente, especifique as MTs  $M_1, M_2, \dots, M_k$ ;

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Esboço da Prova:** Vamos criar Máquinas de Turing distintas, cada uma delas com cinco fitas, que executam cada uma das instruções assembly. Mais precisamente,

- A partir de  $\Pi = \pi_1, \pi_2, \dots, \pi_k$ , primeiramente, especifique as MTs  $M_1, M_2, \dots, M_k$ ; (cada MT pode ser construída como no exercício anterior)

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Esboço da Prova:** Vamos criar Máquinas de Turing distintas, cada uma delas com cinco fitas, que executam cada uma das instruções assembly. Mais precisamente,

- A partir de  $\Pi = \pi_1, \pi_2, \dots, \pi_k$ , primeiramente, especifique as MTs  $M_1, M_2, \dots, M_k$ ; (cada MT pode ser construída como no exercício anterior)
- Partindo destas  $k$  máquinas, podemos construir a MT  $M_\Pi$  com cinco fitas que contém “subrotinas”  $M_1, \dots, M_k$ .

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Esboço da Prova:** Vamos criar Máquinas de Turing distintas, cada uma delas com cinco fitas, que executam cada uma das instruções assembly. Mais precisamente,

- A partir de  $\Pi = \pi_1, \pi_2, \dots, \pi_k$ , primeiramente, especifique as MTs  $M_1, M_2, \dots, M_k$ ; (cada MT pode ser construída como no exercício anterior)
- Partindo destas  $k$  máquinas, podemos construir a MT  $M_\Pi$  com cinco fitas que contém “subrotinas”  $M_1, \dots, M_k$ . A MT  $M_\Pi$  executa em ciclos:

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Esboço da Prova:** Vamos criar Máquinas de Turing distintas, cada uma delas com cinco fitas, que executam cada uma das instruções assembly. Mais precisamente,

- A partir de  $\Pi = \pi_1, \pi_2, \dots, \pi_k$ , primeiramente, especifique as MTs  $M_1, M_2, \dots, M_k$ ; (cada MT pode ser construída como no exercício anterior)
- Partindo destas  $k$  máquinas, podemos construir a MT  $M_\Pi$  com cinco fitas que contém “subrotinas”  $M_1, \dots, M_k$ . A MT  $M_\Pi$  executa em ciclos:
  - A cada ciclo,  $M_\Pi$  olha na fita do contador de programa e decide qual sub-rotina  $M_i$  deve ser executada;

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Esboço da Prova:** Vamos criar Máquinas de Turing distintas, cada uma delas com cinco fitas, que executam cada uma das instruções assembly. Mais precisamente,

- A partir de  $\Pi = \pi_1, \pi_2, \dots, \pi_k$ , primeiramente, especifique as MTs  $M_1, M_2, \dots, M_k$ ; (cada MT pode ser construída como no exercício anterior)
- Partindo destas  $k$  máquinas, podemos construir a MT  $M_\Pi$  com cinco fitas que contém “subrotinas”  $M_1, \dots, M_k$ . A MT  $M_\Pi$  executa em ciclos:
  - A cada ciclo,  $M_\Pi$  olha na fita do contador de programa e decide qual sub-rotina  $M_i$  deve ser executada;
  - Após executar  $M_i$ , o contador é atualizado por  $M_\Pi$  da seguinte forma:

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Esboço da Prova:** Vamos criar Máquinas de Turing distintas, cada uma delas com cinco fitas, que executam cada uma das instruções assembly. Mais precisamente,

- A partir de  $\Pi = \pi_1, \pi_2, \dots, \pi_k$ , primeiramente, especifique as MTs  $M_1, M_2, \dots, M_k$ ; (cada MT pode ser construída como no exercício anterior)
- Partindo destas  $k$  máquinas, podemos construir a MT  $M_\Pi$  com cinco fitas que contém “subrotinas”  $M_1, \dots, M_k$ . A MT  $M_\Pi$  executa em ciclos:
  - A cada ciclo,  $M_\Pi$  olha na fita do contador de programa e decide qual sub-rotina  $M_i$  deve ser executada;
  - Após executar  $M_i$ , o contador é atualizado por  $M_\Pi$  da seguinte forma: se a sub-rotina não for do tipo JUMP, atualiza o valor para  $i + 1$ ;

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Esboço da Prova:** Vamos criar Máquinas de Turing distintas, cada uma delas com cinco fitas, que executam cada uma das instruções assembly. Mais precisamente,

- A partir de  $\Pi = \pi_1, \pi_2, \dots, \pi_k$ , primeiramente, especifique as MTs  $M_1, M_2, \dots, M_k$ ; (cada MT pode ser construída como no exercício anterior)
- Partindo destas  $k$  máquinas, podemos construir a MT  $M_\Pi$  com cinco fitas que contém “subrotinas”  $M_1, \dots, M_k$ . A MT  $M_\Pi$  executa em ciclos:
  - A cada ciclo,  $M_\Pi$  olha na fita do contador de programa e decide qual sub-rotina  $M_i$  deve ser executada;
  - Após executar  $M_i$ , o contador é atualizado por  $M_\Pi$  da seguinte forma:  
se a sub-rotina não for do tipo JUMP, atualiza o valor para  $i + 1$ ;  
caso contrário, a alteração já terá sido feita pela própria sub-rotina  $M_i$ .

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Esboço da Prova:** Vamos criar Máquinas de Turing distintas, cada uma delas com cinco fitas, que executam cada uma das instruções assembly. Mais precisamente,

- A partir de  $\Pi = \pi_1, \pi_2, \dots, \pi_k$ , primeiramente, especifique as MTs  $M_1, M_2, \dots, M_k$ ; (cada MT pode ser construída como no exercício anterior)
- Partindo destas  $k$  máquinas, podemos construir a MT  $M_\Pi$  com cinco fitas que contém “subrotinas”  $M_1, \dots, M_k$ . A MT  $M_\Pi$  executa em ciclos:
  - A cada ciclo,  $M_\Pi$  olha na fita do contador de programa e decide qual sub-rotina  $M_i$  deve ser executada;
  - Após executar  $M_i$ , o contador é atualizado por  $M_\Pi$  da seguinte forma:  
se a sub-rotina não for do tipo `JUMP`, atualiza o valor para  $i + 1$ ;  
caso contrário, a alteração já terá sido feita pela própria sub-rotina  $M_i$ .

**Teorema:** Para toda MT  $M$ , existe um Programa Assembly  $\Pi_M$  tal que  $L(\Pi_M) = L(M)$ .

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Esboço da Prova:** Vamos criar Máquinas de Turing distintas, cada uma delas com cinco fitas, que executam cada uma das instruções assembly. Mais precisamente,

- A partir de  $\Pi = \pi_1, \pi_2, \dots, \pi_k$ , primeiramente, especifique as MTs  $M_1, M_2, \dots, M_k$ ; (cada MT pode ser construída como no exercício anterior)
- Partindo destas  $k$  máquinas, podemos construir a MT  $M_\Pi$  com cinco fitas que contém “subrotinas”  $M_1, \dots, M_k$ . A MT  $M_\Pi$  executa em ciclos:
  - A cada ciclo,  $M_\Pi$  olha na fita do contador de programa e decide qual sub-rotina  $M_i$  deve ser executada;
  - Após executar  $M_i$ , o contador é atualizado por  $M_\Pi$  da seguinte forma:  
se a sub-rotina não for do tipo JUMP, atualiza o valor para  $i + 1$ ;  
caso contrário, a alteração já terá sido feita pela própria sub-rotina  $M_i$ .

**Teorema:** Para toda MT  $M$ , existe um Programa Assembly  $\Pi_M$  tal que  $L(\Pi_M) = L(M)$ .

**Esboço da Prova:** A partir de  $M$ , podemos especificar um programa que tem uma tabela para a função  $\delta$ .

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Esboço da Prova:** Vamos criar Máquinas de Turing distintas, cada uma delas com cinco fitas, que executam cada uma das instruções assembly. Mais precisamente,

- A partir de  $\Pi = \pi_1, \pi_2, \dots, \pi_k$ , primeiramente, especifique as MTs  $M_1, M_2, \dots, M_k$ ; (cada MT pode ser construída como no exercício anterior)
- Partindo destas  $k$  máquinas, podemos construir a MT  $M_\Pi$  com cinco fitas que contém “subrotinas”  $M_1, \dots, M_k$ . A MT  $M_\Pi$  executa em ciclos:
  - A cada ciclo,  $M_\Pi$  olha na fita do contador de programa e decide qual sub-rotina  $M_i$  deve ser executada;
  - Após executar  $M_i$ , o contador é atualizado por  $M_\Pi$  da seguinte forma:  
se a sub-rotina não for do tipo JUMP, atualiza o valor para  $i + 1$ ;  
caso contrário, a alteração já terá sido feita pela própria sub-rotina  $M_i$ .

**Teorema:** Para toda MT  $M$ , existe um Programa Assembly  $\Pi_M$  tal que  $L(\Pi_M) = L(M)$ .

**Esboço da Prova:** A partir de  $M$ , podemos especificar um programa que tem uma tabela para a função  $\delta$ . A cada passo, aplica a função  $\delta$  na entrada.

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Teorema:** Para toda MT  $M$ , existe um Programa Assembly  $\Pi_M$  tal que  $L(\Pi_M) = L(M)$ .

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Teorema:** Para toda MT  $M$ , existe um Programa Assembly  $\Pi_M$  tal que  $L(\Pi_M) = L(M)$ .

# Simulando Máquinas RAM com MTs

**Teorema:** Para todo Programa Assembly  $\Pi$ , existe uma MT  $M_\Pi$  tal que  $L(M_\Pi) = L(\Pi)$ .

**Teorema:** Para toda MT  $M$ , existe um Programa Assembly  $\Pi_M$  tal que  $L(\Pi_M) = L(M)$ .

**Teorema:** Uma MT  $M$  decide  $L \Leftrightarrow$  Um algoritmo em pseudocódigo decide  $L$ .

# Tese de Church-Turing: Introdução

# Tese de Church-Turing: Introdução

**Teorema:** Se um algoritmo pode ser escrito em um dos modelos de computação abaixo, então também pode ser escrito no outro modelo:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo.

# Tese de Church-Turing: Introdução

**Teorema:** Se um algoritmo pode ser escrito em um dos modelos de computação abaixo, então também pode ser escrito no outro modelo:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo.

- Quais outros modelos matemáticos são equivalentes a Máquinas de Turing?

# Tese de Church-Turing: Introdução

**Teorema:** Se um algoritmo pode ser escrito em um dos modelos de computação abaixo, então também pode ser escrito no outro modelo:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo.

- Quais outros modelos matemáticos são equivalentes a Máquinas de Turing? Muitos!

# Tese de Church-Turing: Introdução

**Teorema:** Se um algoritmo pode ser escrito em um dos modelos de computação abaixo, então também pode ser escrito no outro modelo:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo.

- Quais outros modelos matemáticos são equivalentes a Máquinas de Turing? Muitos!
- Alguns modelos são **puramente abstratos**, outros se propõe a **capturar fenômenos físicos**

# Tese de Church-Turing: Introdução

**Teorema:** Se um algoritmo pode ser escrito em um dos modelos de computação abaixo, então também pode ser escrito no outro modelo:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo.

- Quais outros modelos matemáticos são equivalentes a Máquinas de Turing? Muitos!
  - Alguns modelos são **puramente abstratos**, outros se propõe a **capturar fenômenos físicos**
- Modelo muito importante:**

# Tese de Church-Turing: Introdução

**Teorema:** Se um algoritmo pode ser escrito em um dos modelos de computação abaixo, então também pode ser escrito no outro modelo:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo.

- Quais outros modelos matemáticos são equivalentes a Máquinas de Turing? Muitos!
  - Alguns modelos são **puramente abstratos**, outros se propõe a **capturar fenômenos físicos**
- Modelo muito importante:** Modelo para especificar algoritmos quânticos

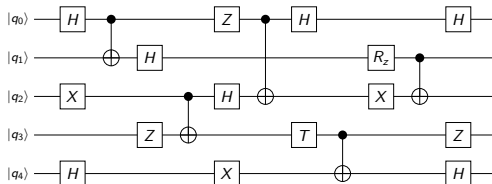
# Tese de Church-Turing: Introdução

**Teorema:** Se um algoritmo pode ser escrito em um dos modelos de computação abaixo, então também pode ser escrito no outro modelo:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo.

- Quais outros modelos matemáticos são equivalentes a Máquinas de Turing? Muitos!
  - Alguns modelos são **puramente abstratos**, outros se propõe a **capturar fenômenos físicos**
- Modelo muito importante:** Modelo para especificar algoritmos quânticos

⇒ **modelo de circuitos quânticos**



# Tese de Church-Turing: Introdução

# Tese de Church-Turing: Introdução

**Teorema da Equivalência:** Se um algoritmo pode ser escrito em um dos três modelos de computação abaixo, então também pode ser escrito nos outros dois modelos:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo;

# Tese de Church-Turing: Introdução

**Teorema da Equivalência:** Se um algoritmo pode ser escrito em um dos três modelos de computação abaixo, então também pode ser escrito nos outros dois modelos:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo;
- (3) Circuitos Quânticos

# Tese de Church-Turing: Introdução

**Teorema da Equivalência:** Se um algoritmo pode ser escrito em um dos três modelos de computação abaixo, então também pode ser escrito nos outros dois modelos:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo;
- (3) Circuitos Quânticos

No modelo (3), um programa pode ser descrito por uma *matriz*  $U$ , a entrada é um *vetor*  $v$

# Tese de Church-Turing: Introdução

**Teorema da Equivalência:** Se um algoritmo pode ser escrito em um dos três modelos de computação abaixo, então também pode ser escrito nos outros dois modelos:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo;
- (3) Circuitos Quânticos

No modelo (3), um programa pode ser descrito por uma *matriz*  $U$ , a entrada é um *vetor*  $v$   
Execução do programa:

# Tese de Church-Turing: Introdução

**Teorema da Equivalência:** Se um algoritmo pode ser escrito em um dos três modelos de computação abaixo, então também pode ser escrito nos outros dois modelos:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo;
- (3) Circuitos Quânticos

No modelo (3), um programa pode ser descrito por uma *matriz*  $U$ , a entrada é um *vetor*  $v$   
Execução do programa: Basta fazer multiplicação  $U \cdot v$  para obter a saída

# Tese de Church-Turing: Introdução

**Teorema da Equivalência:** Se um algoritmo pode ser escrito em um dos três modelos de computação abaixo, então também pode ser escrito nos outros dois modelos:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo;
- (3) Circuitos Quânticos

No modelo (3), um programa pode ser descrito por uma *matriz*  $U$ , a entrada é um *vetor*  $v$   
Execução do programa: Basta fazer multiplicação  $U \cdot v$  para obter a saída

- Richard Feynman (1982):

# Tese de Church-Turing: Introdução

**Teorema da Equivalência:** Se um algoritmo pode ser escrito em um dos três modelos de computação abaixo, então também pode ser escrito nos outros dois modelos:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo;
- (3) Circuitos Quânticos

No modelo (3), um programa pode ser descrito por uma *matriz*  $U$ , a entrada é um *vetor*  $v$   
Execução do programa: Basta fazer multiplicação  $U \cdot v$  para obter a saída

- **Richard Feynman (1982):** De acordo com as leis da mecânica quântica, a evolução temporal de um sistema físico isolado (átomos, elétrons, fótons, etc) pode ser modelada, com precisão arbitrária, por uma abstração matemática chamada *circuito quântico*.

# Tese de Church-Turing: Introdução

**Teorema da Equivalência:** Se um algoritmo pode ser escrito em um dos três modelos de computação abaixo, então também pode ser escrito nos outros dois modelos:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo;
- (3) Circuitos Quânticos

No modelo (3), um programa pode ser descrito por uma *matriz*  $U$ , a entrada é um *vetor*  $v$   
Execução do programa: Basta fazer multiplicação  $U \cdot v$  para obter a saída

- **Richard Feynman (1982)**: De acordo com as leis da mecânica quântica, a evolução temporal de um sistema físico isolado (átomos, elétrons, fótons, etc) pode ser modelada, com precisão arbitrária, por uma abstração matemática chamada *circuito quântico*.
- **David Deutsch (1985)**: Qualquer sistema físico, e não apenas sistemas quânticos microscópicos, pode, em princípio, ser simulado por um computador quântico universal.

# Tese de Church-Turing: Introdução

**Teorema da Equivalência:** Se um algoritmo pode ser escrito em um dos três modelos de computação abaixo, então também pode ser escrito nos outros dois modelos:

- (1) Máquinas de Turing com  $k$  fitas, para qualquer  $k$ ;
- (2) Algoritmos em pseudocódigo;
- (3) Circuitos Quânticos

No modelo (3), um programa pode ser descrito por uma *matriz*  $U$ , a entrada é um *vetor*  $v$   
Execução do programa: Basta fazer multiplicação  $U \cdot v$  para obter a saída

- **Richard Feynman (1982)**: De acordo com as leis da mecânica quântica, a evolução temporal de um sistema físico isolado (átomos, elétrons, fótons, etc) pode ser modelada, com precisão arbitrária, por uma abstração matemática chamada *circuito quântico*.
- **David Deutsch (1985)**: Qualquer sistema físico, e não apenas sistemas quânticos microscópicos, pode, em princípio, ser simulado por um computador quântico universal.

Em outras palavras, **circuitos quânticos, em certo sentido, modelam “qualquer objeto físico”**

# Tese de Church-Turing: Introdução

# Tese de Church-Turing: Introdução

O enunciado do Teorema da Equivalência poderia ser expandido para incluir muitos outros modelos de computação que se propõe a modelar fenômenos naturais. Por exemplo:

# Tese de Church-Turing: Introdução

O enunciado do Teorema da Equivalência poderia ser expandido para incluir muitos outros modelos de computação que se propõe a modelar fenômenos naturais. Por exemplo:

- (1) O modelo *P-systems* para computação celular;

# Tese de Church-Turing: Introdução

O enunciado do Teorema da Equivalência poderia ser expandido para incluir muitos outros modelos de computação que se propõe a modelar fenômenos naturais. Por exemplo:

- (1) O modelo *P-systems* para computação celular;
- (2) O modelo *SCRNs* para computação por reações químicas;

# Tese de Church-Turing: Introdução

O enunciado do Teorema da Equivalência poderia ser expandido para incluir muitos outros modelos de computação que se propõe a modelar fenômenos naturais. Por exemplo:

- (1) O modelo *P-systems* para computação celular;
- (2) O modelo *SCRNs* para computação por reações químicas;
- (3) Modelo ATAM para computação com moléculas de DNA;

# Tese de Church-Turing: Introdução

O enunciado do Teorema da Equivalência poderia ser expandido para incluir muitos outros modelos de computação que se propõe a modelar fenômenos naturais. Por exemplo:

- (1) O modelo *P-systems* para computação celular;
  - (2) O modelo *SCRNs* para computação por reações químicas;
  - (3) Modelo ATAM para computação com moléculas de DNA;
- etc...

# Tese de Church-Turing: Introdução

O enunciado do Teorema da Equivalência poderia ser expandido para incluir muitos outros modelos de computação que se propõe a modelar fenômenos naturais. Por exemplo:

- (1) O modelo *P-systems* para computação celular;
  - (2) O modelo *SCRNs* para computação por reações químicas;
  - (3) Modelo ATAM para computação com moléculas de DNA;
- etc...

Além disso, máquinas de Turing são equivalentes a outros modelos abstratos:

# Tese de Church-Turing: Introdução

O enunciado do Teorema da Equivalência poderia ser expandido para incluir muitos outros modelos de computação que se propõe a modelar fenômenos naturais. Por exemplo:

- (1) O modelo *P-systems* para computação celular;
  - (2) O modelo *SCRNs* para computação por reações químicas;
  - (3) Modelo ATAM para computação com moléculas de DNA;
- etc...

Além disso, máquinas de Turing são equivalentes a outros modelos abstratos:

- (1) Máquinas de Turing com  $k$  fitas multidimensionais, para qualquer  $k$ ;

# Tese de Church-Turing: Introdução

O enunciado do Teorema da Equivalência poderia ser expandido para incluir muitos outros modelos de computação que se propõe a modelar fenômenos naturais. Por exemplo:

- (1) O modelo *P-systems* para computação celular;
  - (2) O modelo *SCRNs* para computação por reações químicas;
  - (3) Modelo ATAM para computação com moléculas de DNA;
- etc...

Além disso, máquinas de Turing são equivalentes a outros modelos abstratos:

- (1) Máquinas de Turing com  $k$  fitas multidimensionais, para qualquer  $k$ ;
- (2) Cálculo  $\lambda$ ;

# Tese de Church-Turing: Introdução

O enunciado do Teorema da Equivalência poderia ser expandido para incluir muitos outros modelos de computação que se propõe a modelar fenômenos naturais. Por exemplo:

- (1) O modelo *P-systems* para computação celular;
  - (2) O modelo *SCRNs* para computação por reações químicas;
  - (3) Modelo ATAM para computação com moléculas de DNA;
- etc...

Além disso, máquinas de Turing são equivalentes a outros modelos abstratos:

- (1) Máquinas de Turing com  $k$  fitas multidimensionais, para qualquer  $k$ ;
- (2) Cálculo  $\lambda$ ;
- (3) Funções  $\mu$ -recursivas;

# Tese de Church-Turing: Introdução

O enunciado do Teorema da Equivalência poderia ser expandido para incluir muitos outros modelos de computação que se propõe a modelar fenômenos naturais. Por exemplo:

- (1) O modelo *P-systems* para computação celular;
  - (2) O modelo *SCRNs* para computação por reações químicas;
  - (3) Modelo ATAM para computação com moléculas de DNA;
- etc...

Além disso, máquinas de Turing são equivalentes a outros modelos abstratos:

- (1) Máquinas de Turing com  $k$  fitas multidimensionais, para qualquer  $k$ ;
- (2) Cálculo  $\lambda$ ;
- (3) Funções  $\mu$ -recursivas;
- (4) Jogo da Vida de Conway.

# Tese de Church-Turing: Introdução

O enunciado do Teorema da Equivalência poderia ser expandido para incluir muitos outros modelos de computação que se propõe a modelar fenômenos naturais. Por exemplo:

- (1) O modelo *P-systems* para computação celular;
  - (2) O modelo *SCRNs* para computação por reações químicas;
  - (3) Modelo ATAM para computação com moléculas de DNA;
- etc...

Além disso, máquinas de Turing são equivalentes a outros modelos abstratos:

- (1) Máquinas de Turing com  $k$  fitas multidimensionais, para qualquer  $k$ ;
  - (2) Cálculo  $\lambda$ ;
  - (3) Funções  $\mu$ -recursivas;
  - (4) Jogo da Vida de Conway.
- etc...

# O que é a Tese de Church-Turing

**Tese de Church-Turing (TCT):** Máquinas de Turing capturam o conceito de algoritmo.

# O que é a Tese de Church-Turing

**Tese de Church-Turing (TCT):** Máquinas de Turing capturam o conceito de algoritmo.

- Ao invés de dizer algoritmo, as vezes a tese é enunciada usando outros termos:

# O que é a Tese de Church-Turing

**Tese de Church-Turing (TCT):** Máquinas de Turing capturam o conceito de algoritmo.

- Ao invés de dizer algoritmo, as vezes a tese é enunciada usando outros termos: *computação efetiva, procedimento efetivo, processo computacional*, etc

# O que é a Tese de Church-Turing

**Tese de Church-Turing (TCT):** Máquinas de Turing capturam o conceito de algoritmo.

- Ao invés de dizer algoritmo, as vezes a tese é enunciada usando outros termos: *computação efetiva, procedimento efetivo, processo computacional*, etc

Para entender o enunciado da TCT, vamos  
parar por um instante para filosofar um pouco!

# O que é a Tese de Church-Turing

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

Você consegue notar que os enunciados acima são de natureza diferente?

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

Você consegue notar que os enunciados acima são de natureza diferente?

1. O primeiro enunciado é uma definição nominal;

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

Você consegue notar que os enunciados acima são de natureza diferente?

1. O primeiro enunciado é uma definição nominal;  
Exemplo semelhante:

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

Você consegue notar que os enunciados acima são de natureza diferente?

1. O primeiro enunciado é uma definição nominal;  
Exemplo semelhante: Um número primo é um inteiro  $n \geq 2$  divisível apenas por 1 e  $n$ .

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

Você consegue notar que os enunciados acima são de natureza diferente?

1. O primeiro enunciado é uma definição nominal;  
Exemplo semelhante: Um número primo é um inteiro  $n \geq 2$  divisível apenas por 1 e  $n$ .
2. O segundo enunciado é uma proposição matemática;

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

Você consegue notar que os enunciados acima são de natureza diferente?

1. O primeiro enunciado é uma definição nominal;  
Exemplo semelhante: Um número primo é um inteiro  $n \geq 2$  divisível apenas por 1 e  $n$ .
2. O segundo enunciado é uma proposição matemática;  
Exemplo semelhante:

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

Você consegue notar que os enunciados acima são de natureza diferente?

1. O primeiro enunciado é uma definição nominal;  
Exemplo semelhante: Um número primo é um inteiro  $n \geq 2$  divisível apenas por 1 e  $n$ .
2. O segundo enunciado é uma proposição matemática;  
Exemplo semelhante: Todo inteiro maior que 2 é a soma de dois primos.

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

Você consegue notar que os enunciados acima são de natureza diferente?

1. O primeiro enunciado é uma definição nominal;  
Exemplo semelhante: Um número primo é um inteiro  $n \geq 2$  divisível apenas por 1 e  $n$ .
2. O segundo enunciado é uma proposição matemática;  
Exemplo semelhante: Todo inteiro maior que 2 é a soma de dois primos.
3. O terceiro enunciado é um juízo

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

Você consegue notar que os enunciados acima são de natureza diferente?

1. O primeiro enunciado é uma definição nominal;  
Exemplo semelhante: Um número primo é um inteiro  $n \geq 2$  divisível apenas por 1 e  $n$ .
2. O segundo enunciado é uma proposição matemática;  
Exemplo semelhante: Todo inteiro maior que 2 é a soma de dois primos.
3. O terceiro enunciado é um juízo de adequação conceitual;

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

Você consegue notar que os enunciados acima são de natureza diferente?

1. O primeiro enunciado é uma definição nominal;  
Exemplo semelhante: Um número primo é um inteiro  $n \geq 2$  divisível apenas por 1 e  $n$ .
2. O segundo enunciado é uma proposição matemática;  
Exemplo semelhante: Todo inteiro maior que 2 é a soma de dois primos.
3. O terceiro enunciado é um juízo de adequação conceitual;  
Exemplo semelhante:

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

Você consegue notar que os enunciados acima são de natureza diferente?

1. O primeiro enunciado é uma definição nominal;  
Exemplo semelhante: Um número primo é um inteiro  $n \geq 2$  divisível apenas por 1 e  $n$ .
2. O segundo enunciado é uma proposição matemática;  
Exemplo semelhante: Todo inteiro maior que 2 é a soma de dois primos.
3. O terceiro enunciado é um juízo de adequação conceitual;  
Exemplo semelhante: O conjunto de funções  $f$  que satisfazem  $\forall \varepsilon > 0, \exists \delta > 0$  tal que  $|x - a| < \delta \Rightarrow |f(x) - f(a)| < \varepsilon$  são as funções que o gráfico não tem nenhum “salto”

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

Você consegue notar que os enunciados acima são de natureza diferente?

1. O primeiro enunciado é uma definição nominal;  
Exemplo semelhante: Um número primo é um inteiro  $n \geq 2$  divisível apenas por 1 e  $n$ .
2. O segundo enunciado é uma proposição matemática;  
Exemplo semelhante: Todo inteiro maior que 2 é a soma de dois primos.
3. O terceiro enunciado é um juízo de adequação conceitual;  
Exemplo semelhante: O conjunto de funções  $f$  que satisfazem  $\forall \varepsilon > 0, \exists \delta > 0$  tal que  $|x - a| < \delta \Rightarrow |f(x) - f(a)| < \varepsilon$  são as funções que o gráfico não tem nenhum “salto” (i.e., a definição padrão com  $\varepsilon$ 's e  $\delta$ 's se adequa ao conceito de função contínua).

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

Você consegue notar que os enunciados acima são de natureza diferente?

1. O primeiro enunciado é uma definição nominal;  
Exemplo semelhante: Um número primo é um inteiro  $n \geq 2$  divisível apenas por 1 e  $n$ .
2. O segundo enunciado é uma proposição matemática;  
Exemplo semelhante: Todo inteiro maior que 2 é a soma de dois primos.
3. O terceiro enunciado é um juízo de adequação conceitual;  
Exemplo semelhante: O conjunto de funções  $f$  que satisfazem  $\forall \epsilon > 0, \exists \delta > 0$  tal que  $|x - a| < \delta \Rightarrow |f(x) - f(a)| < \epsilon$  são as funções que o gráfico não tem nenhum “salto” (i.e., a definição padrão com  $\epsilon$ 's e  $\delta$ 's se adequa ao conceito de função contínua).
4. O quarto enunciado é uma afirmação sobre a natureza.

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

Você consegue notar que os enunciados acima são de natureza diferente?

1. O primeiro enunciado é uma definição nominal;  
Exemplo semelhante: Um número primo é um inteiro  $n \geq 2$  divisível apenas por 1 e  $n$ .
2. O segundo enunciado é uma proposição matemática;  
Exemplo semelhante: Todo inteiro maior que 2 é a soma de dois primos.
3. O terceiro enunciado é um juízo de adequação conceitual;  
Exemplo semelhante: O conjunto de funções  $f$  que satisfazem  $\forall \epsilon > 0, \exists \delta > 0$  tal que  $|x - a| < \delta \Rightarrow |f(x) - f(a)| < \epsilon$  são as funções que o gráfico não tem nenhum “salto” (i.e., a definição padrão com  $\epsilon$ 's e  $\delta$ 's se adequa ao conceito de função contínua).
4. O quarto enunciado é uma afirmação sobre a natureza.  
Exemplo semelhante:

# O que é a Tese de Church-Turing

Considere os quatro seguintes enunciados:

1. Uma máquina de Turing é uma 7-tupla  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , onde  $Q, \Sigma, \Gamma, F$  são conjuntos tal que  $\Sigma \subset \Gamma, F \subseteq Q, q_0 \in Q, B \notin \Gamma$  e  $\delta : Q \setminus F \times \Gamma \mapsto Q \times \Gamma \times \mathcal{D}$ ;
2. Uma máquina de Turing é equivalente a uma máquina RAM;
3. Uma máquina de Turing é a definição do que entendemos por algoritmo;
4. Uma máquina de Turing é equivalente a qualquer modelo de computação que seja fisicamente realizável.

Você consegue notar que os enunciados acima são de natureza diferente?

1. O primeiro enunciado é uma definição nominal;  
Exemplo semelhante: Um número primo é um inteiro  $n \geq 2$  divisível apenas por 1 e  $n$ .
2. O segundo enunciado é uma proposição matemática;  
Exemplo semelhante: Todo inteiro maior que 2 é a soma de dois primos.
3. O terceiro enunciado é um juízo de adequação conceitual;  
Exemplo semelhante: O conjunto de funções  $f$  que satisfazem  $\forall \epsilon > 0, \exists \delta > 0$  tal que  $|x - a| < \delta \Rightarrow |f(x) - f(a)| < \epsilon$  são as funções que o gráfico não tem nenhum “salto” (i.e., a definição padrão com  $\epsilon$ 's e  $\delta$ 's se adequa ao conceito de função contínua).
4. O quarto enunciado é uma afirmação sobre a natureza.  
Exemplo semelhante: A luz viaja no vácuo a 300.000 km/s

# O que é a Tese de Church-Turing

**Tese de Church-Turing (TCT):** Máquinas de Turing capturam o conceito de algoritmo.

# O que é a Tese de Church-Turing

**Tese de Church-Turing (TCT):** Máquinas de Turing capturam o conceito de algoritmo.

- A TCT normalmente é entendida como sendo

# O que é a Tese de Church-Turing

**Tese de Church-Turing (TCT):** Máquinas de Turing capturam o conceito de algoritmo.

- A TCT normalmente é entendida como sendo
  - A TCT é um *juízo de adequação conceitual* que torna matematicamente preciso o que queremos dizer por algoritmo por meio da definição da máquina de Turing.

# O que é a Tese de Church-Turing

**Tese de Church-Turing (TCT):** Máquinas de Turing capturam o conceito de algoritmo.

- A TCT normalmente é entendida como sendo
  - A TCT é um *juízo de adequação conceitual* que torna matematicamente preciso o que queremos dizer por algoritmo por meio da definição da máquina de Turing.
  - A TCT é uma *afirmação sobre a natureza*, ou seja, sobre o que pode ser computado por sistemas fisicamente realizáveis.

# O que é a Tese de Church-Turing

**Tese de Church-Turing (TCT):** Máquinas de Turing capturam o conceito de algoritmo.

- A TCT normalmente é entendida como sendo
  - A TCT é um *juízo de adequação conceitual* que torna matematicamente preciso o que queremos dizer por algoritmo por meio da definição da máquina de Turing.
  - A TCT é uma *afirmação sobre a natureza*, ou seja, sobre o que pode ser computado por sistemas fisicamente realizáveis.

# Juízo de Adequação de Conceito – TCT

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

- Todos temos uma **ideia intuitiva e prévia** do que é um círculo.

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

- Todos temos uma **ideia intuitiva e prévia** do que é um círculo.
- Para estudar o conceito rigorosamente, precisamos de uma **definição exata**.

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

- Todos temos uma **ideia intuitiva e prévia** do que é um círculo.
- Para estudar o conceito rigorosamente, precisamos de uma **definição exata**.

A definição matemática padrão:

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

- Todos temos uma **ideia intuitiva e prévia** do que é um círculo.
- Para estudar o conceito rigorosamente, precisamos de uma **definição exata**.

A definição matemática padrão:

*Para todo real  $r > 0$  e todo  $c = (c_1, c_2) \in \mathbb{R}^2$ ,*

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

- Todos temos uma **ideia intuitiva e prévia** do que é um círculo.
- Para estudar o conceito rigorosamente, precisamos de uma **definição exata**.

A definição matemática padrão:

*Para todo real  $r > 0$  e todo  $c = (c_1, c_2) \in \mathbb{R}^2$ , um círculo é o conjunto  $C$  de pontos  $(x, y) \in \mathbb{R}^2$  a distância  $r$  do centro  $c$ .*

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

- Todos temos uma **ideia intuitiva e prévia** do que é um círculo.
- Para estudar o conceito rigorosamente, precisamos de uma **definição exata**.

A definição matemática padrão:

*Para todo real  $r > 0$  e todo  $c = (c_1, c_2) \in \mathbb{R}^2$ , um círculo é o conjunto  $C$  de pontos  $(x, y) \in \mathbb{R}^2$  a distância  $r$  do centro  $c$ . Mais precisamente*

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

- Todos temos uma **ideia intuitiva e prévia** do que é um círculo.
- Para estudar o conceito rigorosamente, precisamos de uma **definição exata**.

A definição matemática padrão:

*Para todo real  $r > 0$  e todo  $c = (c_1, c_2) \in \mathbb{R}^2$ , um círculo é o conjunto  $C$  de pontos  $(x, y) \in \mathbb{R}^2$  a distância  $r$  do centro  $c$ . Mais precisamente*

$$C = \{(x, y) \in \mathbb{R}^2 : (x - c_1)^2 + (y - c_2)^2 = r^2\}.$$

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

- Todos temos uma **ideia intuitiva e prévia** do que é um círculo.
- Para estudar o conceito rigorosamente, precisamos de uma **definição exata**.

A definição matemática padrão:

*Para todo real  $r > 0$  e todo  $c = (c_1, c_2) \in \mathbb{R}^2$ , um círculo é o conjunto  $C$  de pontos  $(x, y) \in \mathbb{R}^2$  a distância  $r$  do centro  $c$ . Mais precisamente*

$$C = \{(x, y) \in \mathbb{R}^2 : (x - c_1)^2 + (y - c_2)^2 = r^2\}.$$

- Juízo sendo feito:

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

- Todos temos uma **ideia intuitiva e prévia** do que é um círculo.
- Para estudar o conceito rigorosamente, precisamos de uma **definição exata**.

A definição matemática padrão:

*Para todo real  $r > 0$  e todo  $c = (c_1, c_2) \in \mathbb{R}^2$ , um círculo é o conjunto  $C$  de pontos  $(x, y) \in \mathbb{R}^2$  a distância  $r$  do centro  $c$ . Mais precisamente*

$$C = \{(x, y) \in \mathbb{R}^2 : (x - c_1)^2 + (y - c_2)^2 = r^2\}.$$

- **Juízo sendo feito:** A definição acima se adequa à noção de círculo.

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

- Todos temos uma **ideia intuitiva e prévia** do que é um círculo.
- Para estudar o conceito rigorosamente, precisamos de uma **definição exata**.

A definição matemática padrão:

*Para todo real  $r > 0$  e todo  $c = (c_1, c_2) \in \mathbb{R}^2$ , um círculo é o conjunto  $C$  de pontos  $(x, y) \in \mathbb{R}^2$  a distância  $r$  do centro  $c$ . Mais precisamente*

$$C = \{(x, y) \in \mathbb{R}^2 : (x - c_1)^2 + (y - c_2)^2 = r^2\}.$$

- **Juízo sendo feito:** A definição acima se adequa à noção de círculo. Possível justificativa: Outras definições que expressem o queremos dizer por círculo são **equivalente a essa**.

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

- Todos temos uma **ideia intuitiva e prévia** do que é um círculo.
- Para estudar o conceito rigorosamente, precisamos de uma **definição exata**.

A definição matemática padrão:

*Para todo real  $r > 0$  e todo  $c = (c_1, c_2) \in \mathbb{R}^2$ , um círculo é o conjunto  $C$  de pontos  $(x, y) \in \mathbb{R}^2$  a distância  $r$  do centro  $c$ . Mais precisamente*

$$C = \{(x, y) \in \mathbb{R}^2 : (x - c_1)^2 + (y - c_2)^2 = r^2\}.$$

- **Juízo sendo feito:** A definição acima se adequa à noção de círculo. Possível justificativa: Outras definições que expressem o queremos dizer por círculo são **equivalente a essa**.

Exemplo: Um círculo é um espaço homeomorfo ao conjunto  $\{(x, y) \in \mathbb{R}^2 : x^2 + y^2 = 1\}$

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

- Todos temos uma **ideia intuitiva e prévia** do que é um círculo.
- Para estudar o conceito rigorosamente, precisamos de uma **definição exata**.

A definição matemática padrão:

*Para todo real  $r > 0$  e todo  $c = (c_1, c_2) \in \mathbb{R}^2$ , um círculo é o conjunto  $C$  de pontos  $(x, y) \in \mathbb{R}^2$  a distância  $r$  do centro  $c$ . Mais precisamente*

$$C = \{(x, y) \in \mathbb{R}^2 : (x - c_1)^2 + (y - c_2)^2 = r^2\}.$$

- **Juízo sendo feito:** A definição acima se adequa à noção de círculo. Possível justificativa: Outras definições que expressem o queremos dizer por círculo são **equivalente a essa**.  
Exemplo: Um círculo é um espaço homeomorfo ao conjunto  $\{(x, y) \in \mathbb{R}^2 : x^2 + y^2 = 1\}$
- O mesmo raciocínio vale para a Tese de Church-Turing: Uma MT é a **definição formal adequada** do conceito intuitivo de algoritmo.

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

- Todos temos uma **ideia intuitiva e prévia** do que é um círculo.
- Para estudar o conceito rigorosamente, precisamos de uma **definição exata**.

A definição matemática padrão:

*Para todo real  $r > 0$  e todo  $c = (c_1, c_2) \in \mathbb{R}^2$ , um círculo é o conjunto  $C$  de pontos  $(x, y) \in \mathbb{R}^2$  a distância  $r$  do centro  $c$ . Mais precisamente*

$$C = \{(x, y) \in \mathbb{R}^2 : (x - c_1)^2 + (y - c_2)^2 = r^2\}.$$

- **Juízo sendo feito:** A definição acima se adequa à noção de círculo. Possível justificativa: Outras definições que expressem o queremos dizer por círculo são **equivalente a essa**.  
Exemplo: Um círculo é um espaço homeomorfo ao conjunto  $\{(x, y) \in \mathbb{R}^2 : x^2 + y^2 = 1\}$
- O mesmo raciocínio vale para a Tese de Church-Turing: Uma MT é a **definição formal adequada** do conceito intuitivo de algoritmo.
- Portanto, uma vez que aceitamos esta leitura da tese, podemos concluir que:

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

- Todos temos uma **ideia intuitiva e prévia** do que é um círculo.
- Para estudar o conceito rigorosamente, precisamos de uma **definição exata**.

A definição matemática padrão:

*Para todo real  $r > 0$  e todo  $c = (c_1, c_2) \in \mathbb{R}^2$ , um círculo é o conjunto  $C$  de pontos  $(x, y) \in \mathbb{R}^2$  a distância  $r$  do centro  $c$ . Mais precisamente*

$$C = \{(x, y) \in \mathbb{R}^2 : (x - c_1)^2 + (y - c_2)^2 = r^2\}.$$

- **Juízo sendo feito:** A definição acima se adequa à noção de círculo. Possível justificativa: Outras definições que expressem o queremos dizer por círculo são **equivalente a essa**.  
Exemplo: Um círculo é um espaço homeomorfo ao conjunto  $\{(x, y) \in \mathbb{R}^2 : x^2 + y^2 = 1\}$
- O mesmo raciocínio vale para a Tese de Church-Turing: Uma MT é a **definição formal adequada** do conceito intuitivo de algoritmo.
- Portanto, uma vez que aceitamos esta leitura da tese, podemos concluir que:
  - Se existe uma Máquina de Turing que sempre para para decidir um problema, então por definição existe um algoritmo para o problema.

# Juízo de Adequação de Conceito – TCT

Além da analogia das funções contínuas, outro analogia comum é a do círculo:

- Todos temos uma **ideia intuitiva e prévia** do que é um círculo.
- Para estudar o conceito rigorosamente, precisamos de uma **definição exata**.

A definição matemática padrão:

*Para todo real  $r > 0$  e todo  $c = (c_1, c_2) \in \mathbb{R}^2$ , um círculo é o conjunto  $C$  de pontos  $(x, y) \in \mathbb{R}^2$  a distância  $r$  do centro  $c$ . Mais precisamente*

$$C = \{(x, y) \in \mathbb{R}^2 : (x - c_1)^2 + (y - c_2)^2 = r^2\}.$$

- **Juízo sendo feito:** A definição acima se adequa à noção de círculo. Possível justificativa: Outras definições que expressem o queremos dizer por círculo são **equivalente a essa**.  
Exemplo: Um círculo é um espaço homeomorfo ao conjunto  $\{(x, y) \in \mathbb{R}^2 : x^2 + y^2 = 1\}$
- O mesmo raciocínio vale para a Tese de Church-Turing: Uma MT é a **definição formal adequada** do conceito intuitivo de algoritmo.
- Portanto, uma vez que aceitamos esta leitura da tese, podemos concluir que:
  - Se existe uma Máquina de Turing que sempre para para decidir um problema, então por definição existe um algoritmo para o problema.
  - Se não existe tal máquina, então por definição não existe algoritmo.

# Juízo de Adequação de Conceito – TCT

- Outras definições surgiram, mas são equivalentes a Máquinas de Turing:

# Juízo de Adequação de Conceito – TCT

- Outras definições surgiram, mas são equivalentes a Máquinas de Turing:  
e.g., Cálculo  $\lambda$ , funções  $\mu$ -recurvivas, etc

# Juízo de Adequação de Conceito – TCT

- Outras definições surgiram, mas são equivalentes a Máquinas de Turing:  
e.g., Cálculo  $\lambda$ , funções  $\mu$ -recurvivas, etc
- *Hipercomputação*: Área que estuda definições de modelos mais poderosos que MTs

- Outras definições surgiram, mas são equivalentes a Máquinas de Turing:  
e.g., Cálculo  $\lambda$ , funções  $\mu$ -recursivas, etc
- *Hipercomputação*: Área que estuda definições de modelos mais poderosos que MTs  
⇒ O problema é que estas definições sempre acabam tendo passos “mágicos”.

- Outras definições surgiram, mas são equivalentes a Máquinas de Turing:  
e.g., Cálculo  $\lambda$ , funções  $\mu$ -recursivas, etc
- *Hipercomputação*: Área que estuda definições de modelos mais poderosos que MTs  
 $\Rightarrow$  O problema é que estas definições sempre acabam tendo passos “mágicos”.

*“Computabilidade mecanicamente definida foi estabelecida por Alan Turing sem dúvida alguma.”*  
— Kurt Gödel

# A TCT como afirmação sobre a natureza

TESE DE CHURCH-TURING: *Se um problema computacional pode ser resolvido por algum dispositivo fisicamente realizável, então ele pode ser resolvido por uma Máquina de Turing*

# A TCT como afirmação sobre a natureza

TESE DE CHURCH-TURING: *Se um problema computacional pode ser resolvido por algum dispositivo fisicamente realizável, então ele pode ser resolvido por uma Máquina de Turing*

Vantagem desta leitura da tese: há um critério para a refutação, caso a tese se revele falsa:

# A TCT como afirmação sobre a natureza

TESE DE CHURCH-TURING: *Se um problema computacional pode ser resolvido por algum dispositivo fisicamente realizável, então ele pode ser resolvido por uma Máquina de Turing*

Vantagem desta leitura da tese: há um critério para a refutação, caso a tese se revele falsa:

- Assim, como em qualquer hipótese empírica, a Tese de Church-Turing deve ser rejeitada caso seja refutada experimentalmente

# A TCT como afirmação sobre a natureza

TESE DE CHURCH-TURING: *Se um problema computacional pode ser resolvido por algum dispositivo fisicamente realizável, então ele pode ser resolvido por uma Máquina de Turing*

Vantagem desta leitura da tese: há um critério para a refutação, caso a tese se revele falsa:

- Assim, como em qualquer hipótese empírica, a Tese de Church-Turing deve ser rejeitada caso seja refutada experimentalmente (e.g., um dispositivo que resolva, consistentemente, instâncias do problema da parada).

# A TCT como afirmação sobre a natureza

TESE DE CHURCH-TURING: *Se um problema computacional pode ser resolvido por algum dispositivo fisicamente realizável, então ele pode ser resolvido por uma Máquina de Turing*

Vantagem desta leitura da tese: há um critério para a refutação, caso a tese se revele falsa:

- Assim, como em qualquer hipótese empírica, a Tese de Church-Turing deve ser rejeitada caso seja refutada experimentalmente (e.g., um dispositivo que resolva, consistentemente, instâncias do problema da parada).

Razão importante sustentando a Tese:

# A TCT como afirmação sobre a natureza

TESE DE CHURCH-TURING: *Se um problema computacional pode ser resolvido por algum dispositivo fisicamente realizável, então ele pode ser resolvido por uma Máquina de Turing*

Vantagem desta leitura da tese: há um critério para a refutação, caso a tese se revele falsa:

- Assim, como em qualquer hipótese empírica, a Tese de Church-Turing deve ser rejeitada caso seja refutada experimentalmente (e.g., um dispositivo que resolva, consistentemente, instâncias do problema da parada).

Razão importante sustentando a Tese:

- Circuitos quânticos são equivalentes a Máquinas de Turing

# A TCT como afirmação sobre a natureza

TESE DE CHURCH-TURING: *Se um problema computacional pode ser resolvido por algum dispositivo fisicamente realizável, então ele pode ser resolvido por uma Máquina de Turing*

Vantagem desta leitura da tese: há um critério para a refutação, caso a tese se revele falsa:

- Assim, como em qualquer hipótese empírica, a Tese de Church-Turing deve ser rejeitada caso seja refutada experimentalmente (e.g., um dispositivo que resolva, consistentemente, instâncias do problema da parada).

Razão importante sustentando a Tese:

- Circuitos quânticos são equivalentes a Máquinas de Turing
- Vários modelos de física, química, biologia molecular são equivalentes a MTs

# A TCT como afirmação sobre a natureza

**TESE DE CHURCH-TURING:** *Se um problema computacional pode ser resolvido por algum dispositivo fisicamente realizável, então ele pode ser resolvido por uma Máquina de Turing*

Vantagem desta leitura da tese: há um critério para a refutação, caso a tese se revele falsa:

- Assim, como em qualquer hipótese empírica, a Tese de Church-Turing deve ser rejeitada caso seja refutada experimentalmente (e.g., um dispositivo que resolva, consistentemente, instâncias do problema da parada).

Razão importante sustentando a Tese:

- Circuitos quânticos são equivalentes a Máquinas de Turing
- Vários modelos de física, química, biologia molecular são equivalentes a MTs
- *Hipercomputação*: modelos que resolvem problemas não computáveis tipicamente são variações de computação analógica e não parecem ser factíveis:

# A TCT como afirmação sobre a natureza

**TESE DE CHURCH-TURING:** *Se um problema computacional pode ser resolvido por algum dispositivo fisicamente realizável, então ele pode ser resolvido por uma Máquina de Turing*

Vantagem desta leitura da tese: há um critério para a refutação, caso a tese se revele falsa:

- Assim, como em qualquer hipótese empírica, a Tese de Church-Turing deve ser rejeitada caso seja refutada experimentalmente (e.g., um dispositivo que resolva, consistentemente, instâncias do problema da parada).

Razão importante sustentando a Tese:

- Circuitos quânticos são equivalentes a Máquinas de Turing
- Vários modelos de física, química, biologia molecular são equivalentes a MTs
- *Hipercomputação*: modelos que resolvem problemas não computáveis tipicamente são variações de computação analógica e não parecem ser factíveis:
  - Armazenamento de dados com uma quantidade infinita de informação

# A TCT como afirmação sobre a natureza

**TESE DE CHURCH-TURING:** *Se um problema computacional pode ser resolvido por algum dispositivo fisicamente realizável, então ele pode ser resolvido por uma Máquina de Turing*

Vantagem desta leitura da tese: há um critério para a refutação, caso a tese se revele falsa:

- Assim, como em qualquer hipótese empírica, a Tese de Church-Turing deve ser rejeitada caso seja refutada experimentalmente (e.g., um dispositivo que resolva, consistentemente, instâncias do problema da parada).

Razão importante sustentando a Tese:

- Circuitos quânticos são equivalentes a Máquinas de Turing
- Vários modelos de física, química, biologia molecular são equivalentes a MTs
- Hipercomputação*: modelos que resolvem problemas não computáveis tipicamente são variações de computação analógica e não parecem ser factíveis:
  - Armazenamento de dados com uma quantidade infinita de informação (e.g., o número  $\pi$  com todos os seus infinitos dígitos)

# A TCT como afirmação sobre a natureza

**TESE DE CHURCH-TURING:** *Se um problema computacional pode ser resolvido por algum dispositivo fisicamente realizável, então ele pode ser resolvido por uma Máquina de Turing*

Vantagem desta leitura da tese: há um critério para a refutação, caso a tese se revele falsa:

- Assim, como em qualquer hipótese empírica, a Tese de Church-Turing deve ser rejeitada caso seja refutada experimentalmente (e.g., um dispositivo que resolva, consistentemente, instâncias do problema da parada).

Razão importante sustentando a Tese:

- Circuitos quânticos são equivalentes a Máquinas de Turing
- Vários modelos de física, química, biologia molecular são equivalentes a MTs
- *Hipercomputação*: modelos que resolvem problemas não computáveis tipicamente são variações de computação analógica e não parecem ser factíveis:
  - Armazenamento de dados com uma quantidade infinita de informação (e.g., o número  $\pi$  com todos os seus infinitos dígitos)
  - Recuperação destes dados sem erros.

# A TCT como afirmação sobre a natureza

**TESE DE CHURCH-TURING:** *Se um problema computacional pode ser resolvido por algum dispositivo fisicamente realizável, então ele pode ser resolvido por uma Máquina de Turing*

Vantagem desta leitura da tese: há um critério para a refutação, caso a tese se revele falsa:

- Assim, como em qualquer hipótese empírica, a Tese de Church-Turing deve ser rejeitada caso seja refutada experimentalmente (e.g., um dispositivo que resolva, consistentemente, instâncias do problema da parada).

Razão importante sustentando a Tese:

- Circuitos quânticos são equivalentes a Máquinas de Turing
- Vários modelos de física, química, biologia molecular são equivalentes a MTs
- *Hipercomputação*: modelos que resolvem problemas não computáveis tipicamente são variações de computação analógica e não parecem ser factíveis:
  - Armazenamento de dados com uma quantidade infinita de informação (e.g., o número  $\pi$  com todos os seus infinitos dígitos)
  - Recuperação destes dados sem erros.
  - Existem obstáculos para estes modelos impostos pela física teórica

# A TCT como afirmação sobre a natureza

TESE DE CHURCH-TURING: *Se um problema computacional pode ser resolvido por algum dispositivo fisicamente realizável, então ele pode ser resolvido por uma Máquina de Turing*

Vantagem desta leitura da tese: há um critério para a refutação, caso a tese se revele falsa:

- Assim, como em qualquer hipótese empírica, a Tese de Church-Turing deve ser rejeitada caso seja refutada experimentalmente (e.g., um dispositivo que resolva, consistentemente, instâncias do problema da parada).

Razão importante sustentando a Tese:

- Circuitos quânticos são equivalentes a Máquinas de Turing
- Vários modelos de física, química, biologia molecular são equivalentes a MTs
- *Hipercomputação*: modelos que resolvem problemas não computáveis tipicamente são variações de computação analógica e não parecem ser factíveis:
  - Armazenamento de dados com uma quantidade infinita de informação (e.g., o número  $\pi$  com todos os seus infinitos dígitos)
  - Recuperação destes dados sem erros.
  - Existem obstáculos para estes modelos impostos pela física teórica  
Exemplo: *limitante de Bekenstein*

# A TCT como afirmação sobre a natureza

**TESE DE CHURCH-TURING:** *Se um problema computacional pode ser resolvido por algum dispositivo fisicamente realizável, então ele pode ser resolvido por uma Máquina de Turing*

Vantagem desta leitura da tese: há um critério para a refutação, caso a tese se revele falsa:

- Assim, como em qualquer hipótese empírica, a Tese de Church-Turing deve ser rejeitada caso seja refutada experimentalmente (e.g., um dispositivo que resolva, consistentemente, instâncias do problema da parada).

Razão importante sustentando a Tese:

- Circuitos quânticos são equivalentes a Máquinas de Turing
- Vários modelos de física, química, biologia molecular são equivalentes a MTs
- *Hipercomputação*: modelos que resolvem problemas não computáveis tipicamente são variações de computação analógica e não parecem ser factíveis:
  - Armazenamento de dados com uma quantidade infinita de informação (e.g., o número  $\pi$  com todos os seus infinitos dígitos)
  - Recuperação destes dados sem erros.
  - Existem obstáculos para estes modelos impostos pela física teórica  
Exemplo: *limitante de Bekenstein*  $\rightarrow$  impõe um limite à quantidade de informação que uma região finita do espaço pode conter.

# A TCT como afirmação sobre a natureza

**TESE DE CHURCH-TURING:** *Se um problema computacional pode ser resolvido por algum dispositivo fisicamente realizável, então ele pode ser resolvido por uma Máquina de Turing*

Vantagem desta leitura da tese: há um critério para a refutação, caso a tese se revele falsa:

- Assim, como em qualquer hipótese empírica, a Tese de Church-Turing deve ser rejeitada caso seja refutada experimentalmente (e.g., um dispositivo que resolva, consistentemente, instâncias do problema da parada).

Razão importante sustentando a Tese:

- Circuitos quânticos são equivalentes a Máquinas de Turing
- Vários modelos de física, química, biologia molecular são equivalentes a MTs
- *Hipercomputação*: modelos que resolvem problemas não computáveis tipicamente são variações de computação analógica e não parecem ser factíveis:
  - Armazenamento de dados com uma quantidade infinita de informação (e.g., o número  $\pi$  com todos os seus infinitos dígitos)
  - Recuperação destes dados sem erros.
  - Existem obstáculos para estes modelos impostos pela física teórica  
Exemplo: *limitante de Bekenstein*  $\rightarrow$  impõe um limite à quantidade de informação que uma região finita do espaço pode conter.
- As vezes também chamada de “Tese de Church-Turing Física”

# Extrapolações indevidas da Tese de Church-Turing

**Ideia presente no artigo de Alan Turing de 1950:**

# Extrapolações indevidas da Tese de Church-Turing

**Ideia presente no artigo de Alan Turing de 1950:** *Cérebros humanos, sendo “um tipo de máquina”, podem ser simulados por Máquinas de Turing.*

# Extrapolações indevidas da Tese de Church-Turing

**Ideia presente no artigo de Alan Turing de 1950:** *Cérebros humanos, sendo “um tipo de máquina”, podem ser simulados por Máquinas de Turing.*

- Note que, independentemente da posição original de Turing sobre a mente humana,

# Extrapolações indevidas da Tese de Church-Turing

**Ideia presente no artigo de Alan Turing de 1950:** *Cérebros humanos, sendo “um tipo de máquina”, podem ser simulados por Máquinas de Turing.*

- Note que, independentemente da posição original de Turing sobre a mente humana, pode-se dizer em termos contemporâneos que TCT física, de fato, pode implicar essa afirmação

# Extrapolações indevidas da Tese de Church-Turing

**Ideia presente no artigo de Alan Turing de 1950:** *Cérebros humanos, sendo “um tipo de máquina”, podem ser simulados por Máquinas de Turing.*

- Note que, independentemente da posição original de Turing sobre a mente humana, pode-se dizer em termos contemporâneos que TCT física, de fato, pode implicar essa afirmação (supondo que o cérebro respeite as leis da física).

# Extrapolações indevidas da Tese de Church-Turing

**Ideia presente no artigo de Alan Turing de 1950:** *Cérebros humanos, sendo “um tipo de máquina”, podem ser simulados por Máquinas de Turing.*

- Note que, independentemente da posição original de Turing sobre a mente humana, pode-se dizer em termos contemporâneos que TCT física, de fato, pode implicar essa afirmação (supondo que o cérebro respeite as leis da física).

Extrapolação indevida:

# Extrapolações indevidas da Tese de Church-Turing

**Ideia presente no artigo de Alan Turing de 1950:** *Cérebros humanos, sendo “um tipo de máquina”, podem ser simulados por Máquinas de Turing.*

- Note que, independentemente da posição original de Turing sobre a mente humana, pode-se dizer em termos contemporâneos que TCT física, de fato, pode implicar essa afirmação (supondo que o cérebro respeite as leis da física).

**Extrapolação indevida:** Note algo importante que não é afirmado pela TCT física:

# Extrapolações indevidas da Tese de Church-Turing

**Ideia presente no artigo de Alan Turing de 1950:** *Cérebros humanos, sendo “um tipo de máquina”, podem ser simulados por Máquinas de Turing.*

- Note que, independentemente da posição original de Turing sobre a mente humana, pode-se dizer em termos contemporâneos que TCT física, de fato, pode implicar essa afirmação (supondo que o cérebro respeite as leis da física).

**Extrapolação indevida:** Note algo importante que não é afirmado pela TCT física:

- A TCT física não trata de temas como *mente* ou *consciência*,

# Extrapolações indevidas da Tese de Church-Turing

**Ideia presente no artigo de Alan Turing de 1950:** *Cérebros humanos, sendo “um tipo de máquina”, podem ser simulados por Máquinas de Turing.*

- Note que, independentemente da posição original de Turing sobre a mente humana, pode-se dizer em termos contemporâneos que TCT física, de fato, pode implicar essa afirmação (supondo que o cérebro respeite as leis da física).

**Extrapolação indevida:** Note algo importante que não é afirmado pela TCT física:

- A TCT física não trata de temas como *mente* ou *consciência*, sendo que estes temas são centrais quando o assunto é o ser humano.

# Extrapolações indevidas da Tese de Church-Turing

**Ideia presente no artigo de Alan Turing de 1950:** *Cérebros humanos, sendo “um tipo de máquina”, podem ser simulados por Máquinas de Turing.*

- Note que, independentemente da posição original de Turing sobre a mente humana, pode-se dizer em termos contemporâneos que TCT física, de fato, pode implicar essa afirmação (supondo que o cérebro respeite as leis da física).

**Extrapolação indevida:** Note algo importante que não é afirmado pela TCT física:

- A TCT física não trata de temas como *mente* ou *consciência*, sendo que estes temas são centrais quando o assunto é o ser humano.
- Esses temas são estudados filosofia da mente, onde há várias posições a respeito do assunto:

# Extrapolações indevidas da Tese de Church-Turing

**Ideia presente no artigo de Alan Turing de 1950:** *Cérebros humanos, sendo “um tipo de máquina”, podem ser simulados por Máquinas de Turing.*

- Note que, independentemente da posição original de Turing sobre a mente humana, pode-se dizer em termos contemporâneos que TCT física, de fato, pode implicar essa afirmação (supondo que o cérebro respeite as leis da física).

**Extrapolação indevida:** Note algo importante que não é afirmado pela TCT física:

- A TCT física não trata de temas como *mente* ou *consciência*, sendo que estes temas são centrais quando o assunto é o ser humano.
- Esses temas são estudados filosofia da mente, onde há várias posições a respeito do assunto: dualismo, materialismo, funcionalismo, hilomorfismo, etc.

# A Tese de Church-Turing Estendida

**Teorema:** No Teorema da Equivalência, além de ser verdade que os modelos (1) e (2) são equivalentes, também é verdade que eles são polinomialmente equivalentes.

# A Tese de Church-Turing Estendida

**Teorema:** No Teorema da Equivalência, além de ser verdade que os modelos (1) e (2) são equivalentes, também é verdade que eles são polinomialmente equivalentes.

**Tese de Church-Turing Estendida (TCTE):**  
Máquinas de Turing capturam o conceito de *algoritmo eficiente*.

# A Tese de Church-Turing Estendida

**Teorema:** No Teorema da Equivalência, além de ser verdade que os modelos (1) e (2) são equivalentes, também é verdade que eles são polinomialmente equivalentes.

**Tese de Church-Turing Estendida (TCTE):**  
Máquinas de Turing capturam o conceito de *algoritmo eficiente*.

- Ao contrário da TCT, não é a posição normalmente aceita

# A Tese de Church-Turing Estendida

**Teorema:** No Teorema da Equivalência, além de ser verdade que os modelos (1) e (2) são equivalentes, também é verdade que eles são polinomialmente equivalentes.

**Tese de Church-Turing Estendida (TCTE):**  
Máquinas de Turing capturam o conceito de *algoritmo eficiente*.

- Ao contrário da TCT, não é a posição normalmente aceita
- Existem algoritmos quânticos eficientes para problemas que não se sabe se podem ser resolvidos por Máquinas de Turing eficientes.