

FUNDAMENTOS TEÓRICOS DA COMPUTAÇÃO

--- MÁQUINAS DE TURING ---

Introdução

Alan Turing

As Máquinas de Turing foram propostas por Alan Turing (1912-1954) em 1936, quando ele tinha somente 24 anos (confiram sua história!)

Filme “Jogo da Imitação”

Considerado o pai da teoria da computação moderna

As MTs são modelos simples de computação, semelhantes aos AFs e APs, mas com memória **ilimitada** e acesso irrestrito (ao contrário de um pilha)



Máquina de Turing

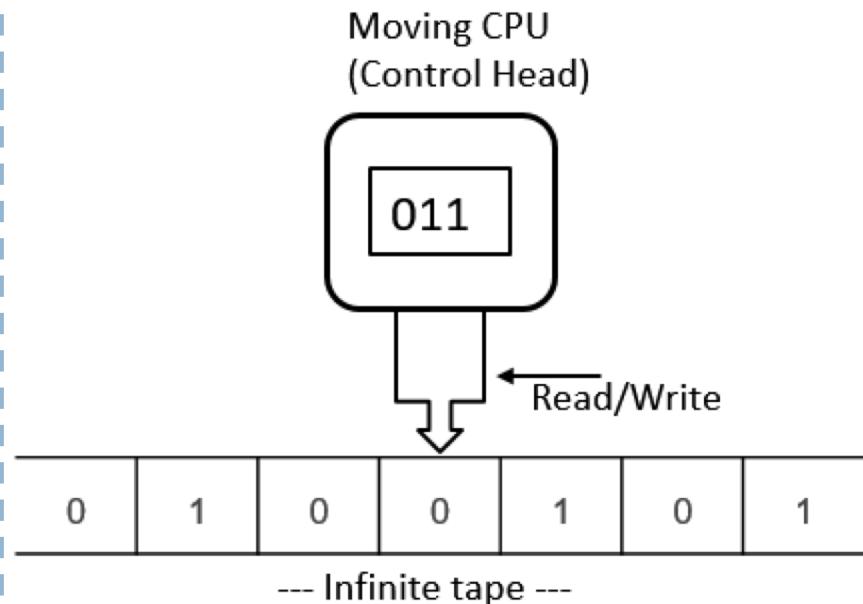


Máquina de Turing feita de madeira. Eram chamadas originalmente de A-Machines

Máquina de Turing

A Máquina de Turing é um dispositivo **teórico** conhecido como **máquina universal**, que consegue simular outra máquina de Turing arbitrária com uma entrada arbitrária

É um modelo **abstrato** de um computador, que se restringe apenas aos **aspectos lógicos** do seu funcionamento (memória, estados e transições), e **não** a sua implementação física



Máquina de Turing

Toda máquina de Turing computa uma certa **função fixa** a partir de uma cadeia de entrada. O mesmo ocorre com um computador moderno com um **programa fixo**

Podemos construir uma máquina de Turing que possui uma cadeia que descreve a **tabela de ação**, seguida por uma cadeia descrevendo sua **fita de entrada**, e finalmente computar a fita que a máquina de Turing codificada computaria

Este modelo é útil para estudar as possibilidades e limites da computação, isto é, a **computabilidade** que é o estudo do que é possível fazer com um computador, que é contrastado com a **complexidade**, que identifica quão difícil é resolver um problema computável

A Bomba

Máquina de Turing pode modelar qualquer computador digital

Turing também se envolveu na construção de máquinas físicas durante a Segunda Guerra Mundial, tendo utilizado alguns dos conceitos teóricos desenvolvidos para o seu modelo de computador universal

Eles produziram uma máquina chamada "**a bomba**" para quebrar os códigos secretos das comunicações alemãs



Enigma

Usando a Máquina "Bomba", Turing e equipe desvendaram o código militar alemão gerado pela **Enigma**, máquina de criptografia supostamente **impenetrável**

A máquina identificava pontos fracos da criptografia e foi responsável por revelar a posição dos submarinos alemães

Acredita-se que o grupo de Turing pode ter colaborado para **encurtar** a duração da guerra



AF, AP e MT

Os dois formalismos vistos até agora (AFs e APs) possuem **poder limitado** para a representação de linguagens formais

AF

$\{a^*\}, \{(aa)^*\}, \{(a,b)^*\}, \{\omega \in (a,b)^* : \omega \text{ não contém } ab\}, \dots$

Permite repetições, mas sem contadores

AP

$\{a^n b^n\}, \{a^n b^{3n}\}, \{a^m b^n a^n b^m\}, \{a^n b^m c^p : n = m + p\}, \dots$

Permite a criação de contadores (aninhados) que podem ser usados uma única vez

AF, AP e MT

A linguagem $L_1 = \{a^n b^k c^n d^k : n, k \geq 0\}$ é livre de contexto?

Não

A linguagem $L_2 = \{a^n b^n c^n : n \geq 0\}$ é livre de contexto?

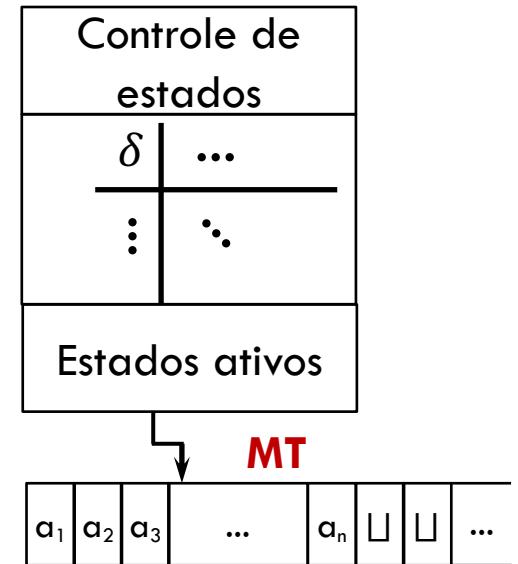
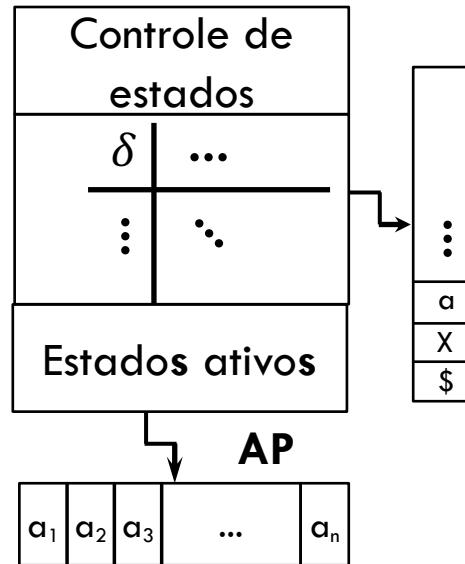
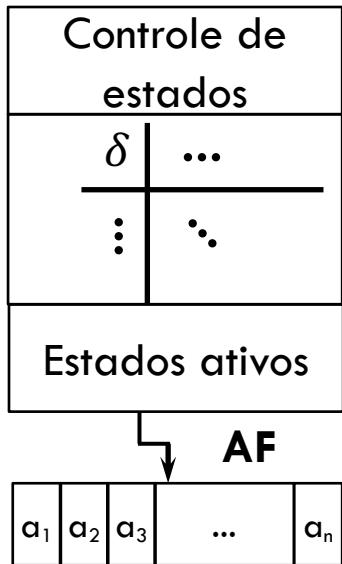
Também não

Precisamos de uma máquina com poder computacional superior às

AFs e APs

R: Máquinas de Turing

AF, AP e MT (parecidas)



Linguagens
Regulares

Linguagens Livres
de Contexto

Linguagens
Decidíveis

* decidibilidade: algoritmo que resolve um problema em número finito de passos

Máquina de Turing

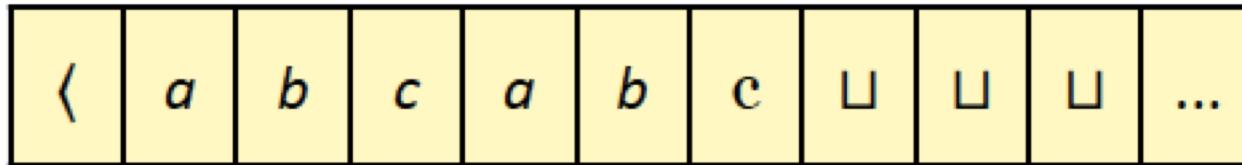
Uma Máquina de Turing (MT) é uma máquina teórica extremamente simples, composta por:

- Um conjunto de estados (como AFs e APs)
- Uma fita de memória com várias células que armazenam um símbolo cada
- Um cabeçote de leitura e escrita que está sempre posicionado em uma das células
- Um conjunto de regras de transição

As MT's são **semelhantes** a um AF, mas com memória **ilimitada** e acesso **irrestrito** a esta memória

É capaz de **fazer tudo** o que um computador **real** pode fazer

Fita de Memória

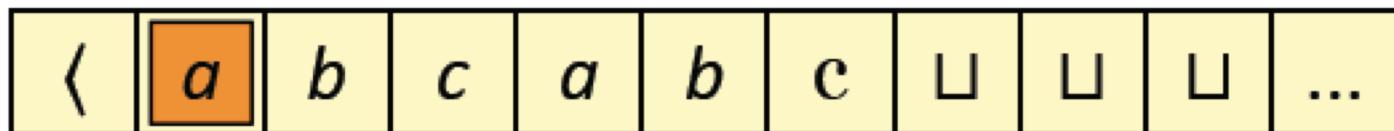


A fita de memória possui **tamanho infinito** e o cabeçote de leitura sempre está posicionado em **uma** das células

As células após a palavra de entrada contêm um símbolo especial para indicar **célula vazia** (será adotado o símbolo **◻**)

Algumas implementações consideram que a **primeira** célula contém um símbolo especial para **indicar o começo da fita**; na figura acima foi adotado o símbolo “<”, outros também usam o símbolo **◻**. Esta é a única célula que **não pode** ser modificada

Cabeçote de leitura e escrita



Configuração inicial para a palavra de entrada abcabc



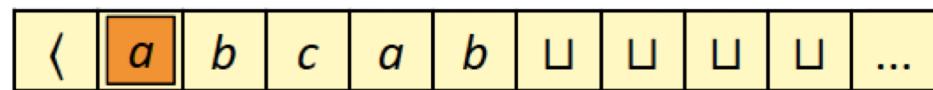
Configuração inicial para entrada vazia

Cabeçote de leitura e escrita

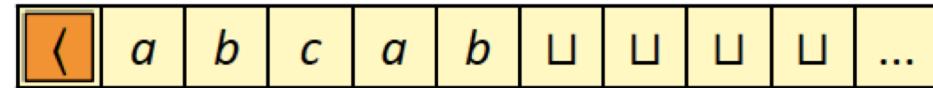
O cabeçote pode mover-se para a esquerda ou para a direita, uma posição por vez

Se o cabeçote estiver na primeira célula, ele não pode mover-se para a esquerda

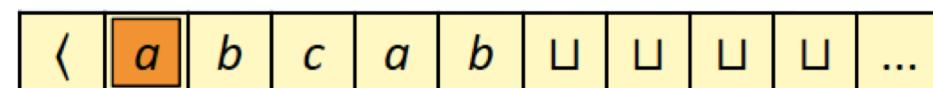
Configuração inicial



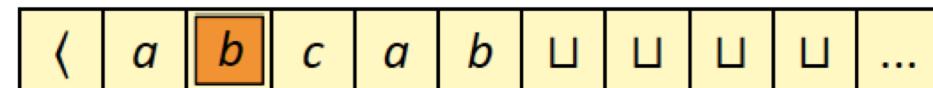
Movimento para esquerda



Movimento para direita



Movimento para direita



Cabeçote de leitura e escrita

Operação básica de uma MT: escrever um símbolo na célula atual e mover o cabeçote para esquerda ou para direita

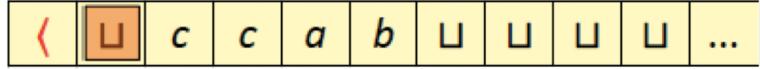
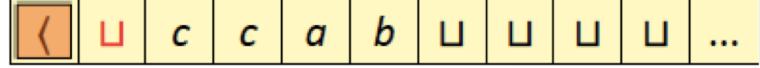
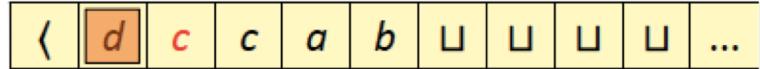
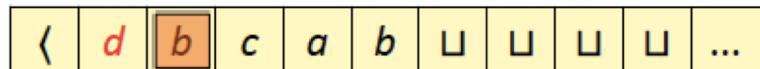
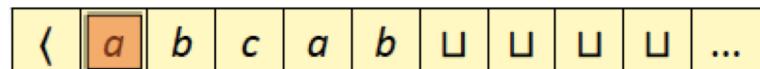
Configuração inicial

Escreve **d** e move direita

Escreve **c** e move esquerda

Escreve **U** e move esquerda

Não escreve em **<** e move direita



Regras de transição

Uma regra de transição depende de dois pre-requisitos: (i) o estado atual; e (ii) o símbolo que está na célula atual (a célula que está sob o cabeçote)

Uma transição de uma MT possui o seguinte formato:

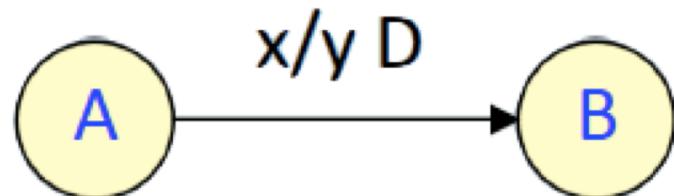
$$\delta(p, a) = (q, b, E) \text{ ou } \delta(p, a) = (q, b, D)$$

Se a MT estiver no estado p e o símbolo sob o cabeçote for a :

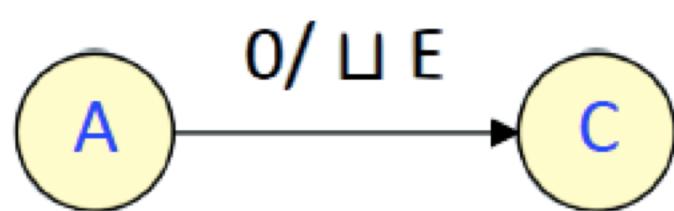
1. Vá para o estado q
2. Escreva o símbolo b na célula atual
3. Mova o cabeçote para esquerda (E) ou para a direita (D)

Regras de transição

$$\delta(A, x) = (B, \gamma, D)$$



$$\delta(A, 0) = (C, \sqcup, E)$$



Regras de transição

Usualmente as transições são **determinísticas**, isto é, não há mais de uma transição para um mesmo estado e símbolo lido

Se não houver transição possível, então a máquina para no estado atual. Às vezes, também há um estado de **rejeição**, embora não seja *exatamente* necessário

Uma MT **aceita** uma palavra de entrada se ela **para** em um estado de **aceitação**

Isto significa que uma palavra não é aceita se a MT parar em um estado de rejeição ou não parar

Não há o conceito de **consumir** toda a palavra

Formalização

Uma máquina de Turing (com uma fita) é definida como uma 6-upla $M = (Q, \Sigma, \Gamma, s, F, \delta)$, onde:

Q é um conjunto finito de estados

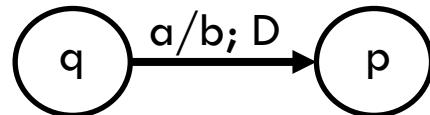
Σ é um alfabeto finito de símbolos (sem o símbolo \sqcup)

Γ é o alfabeto finito de símbolos da fita ($\sqcup \in \Gamma$ e $\Sigma \subseteq \Gamma$)

$s \in Q$ é o estado inicial

$F \subseteq Q$ é o conjunto dos estados de aceitação

$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{E, D\}$ é uma função parcial chamada função de transição, onde E é o movimento para a esquerda e D é o movimento para a direita



$$\delta(q, a) = (p, b, D)$$

Turing Machine Visualization

TURING MACHINE VISUALIZATION binary increment EDIT ▾

LOAD MACHINE REVERT TO DIAGRAM

The visualization shows a Turing Machine with three states: 'right' (blue), 'carry' (orange), and 'done' (green). Transitions are as follows:

- 'right' to 'right': $1, 0 \xrightarrow{\omega} \varnothing$
- 'right' to 'carry': $\varnothing \xrightarrow{\omega \rightarrow L} \varnothing$
- 'carry' to 'done': $0, \varnothing \xrightarrow{1, L} \varnothing$
- 'carry' to 'carry': $1 \xrightarrow{\varnothing} \varnothing$

The tape simulation shows a binary sequence: $[1, 0, 1, 1]$. The first cell is highlighted in yellow.

Buttons: RESET, STEP, RUN

```
1 # Adds 1 to a binary number.
2 input: '1011'
3 blank: ' '
4 start state: right
5 table:
6   # scan to the rightmost digit
7   right:
8     [1, 0]: R
9     '' : {L: carry}
10  # then carry the 1
11  carry:
12    1 : {write: 0, L}
13    [0, ' ']: {write: 1, L: done}
14 done:
15
16
17 # Exercises:
18
19 # • Modify the machine to always halt on the leftmost
20 #   digit
21 #   (regardless of the number's length).
22 #   Hint: add a state between carry and done.
23
24 # • Make a machine that adds 2 instead of 1.
25 #   Hint: 2 is '10' in binary, so the last digit is
26 #   unaffected.
27 #   Alternative hint: chain together two copies of
28 #   the machine from
29 #   the first exercise (renaming the states of the
30 #   second copy).
```

<https://turingmachine.io/>

Turing Machine Visualization

$Q = \{right, carry, done\}$
 $q_0 = right$
 $\Sigma = \{0,1\}$
 $\Gamma = \{0, 1, \sqcup\}$
 $\delta(right, 0) = (right, 0, R)$
 $\delta(right, 1) = (right, 1, R)$
 $\delta(right, \sqcup) = (carry, \sqcup, L)$
 $\delta(carry, 1) = (carry, 0, L)$
 $\delta(carry, 0) = (carry, 1, L)$
 $\delta(carry, \sqcup) = (done, 1, L)$

```
# Adds 1 to a binary number.
input: '1011'
blank: ' '
start state: right
table:
    # scan to the rightmost digit
    right:
        [1,0]: R
        ' ': {L: carry}
    # then carry the 1
    carry:
        1 : {write: 0, L}
        [0, ' ']: {write: 1, L: done}
done:
```

Turing Machine Visualization

```
# Máquina de estados
# pedidos da Amazon
item: Kindle
price: 129.00
start state: waiting
table:
    waiting:
        confirmed: {process: pay}
    pay:
        confirmed: {enqueue: deliver}
    deliver:
        confirmed: done
    done:
    cancel:
```

Parecido com uma ordem de vendas

Funcionamento de uma MT

Primeiro, ω é escrito na fita a partir da posição mais à esquerda

A MT **sempre** começa com o cursor na **primeira posição da fita**

A computação procede de acordo com a função de transição

$$\delta(q, a) = (p, b, D/E)$$

Quando a MT M está em q e lê a da fita:

- M vai para o estado p
- M escreve b na fita sobreescrevendo a
- O cursor vai para a esquerda ou direita

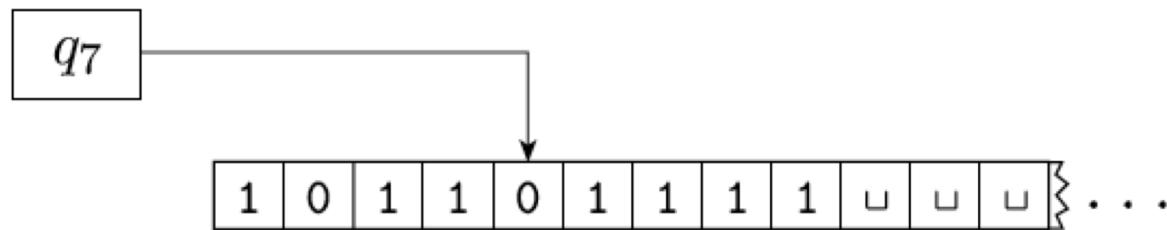
A computação continua até que M alcance um estado de aceitação ou rejeição e a máquina para

Entretanto, a MT pode **nunca parar**, i.e., entrar em loop infinito

Configuração de uma MT

A **configuração** descreve a situação atual da MT:

- Estado atual
- Conteúdo da fita
- Posição do cursor



Configuração: $\underbrace{1011}_{u} \underbrace{q_7}_{v} \underbrace{01111}_{w}$

Quando a MT computa, **mudanças** ocorrem em sua configuração

Configuração de uma MT

Dizemos que uma **configuração** C_1 origina C_2 se a MT M puder ir de C_1 para C_2 , em um único passo, denotado por $C_1 \Rightarrow C_2$

$$\underbrace{1011}_{q_i} \underbrace{01111}_{\#} \Rightarrow \underbrace{1011\#}_{q_j} \underbrace{1111}_{D}$$

$$\delta(q_i, 0) = (q_j, \#, D)$$

Caso especial

$$\underbrace{\omega}_{q_i} \underbrace{101101111}_{\#} \Rightarrow \underbrace{\omega}_{q_j} \underbrace{101101111}_{\#}$$

$$\delta(q_i, 1) = (q_j, \#, E)$$

O cursor nunca ultrapassa a extremidade esquerda

Linguagem reconhecida por uma MT

O conjunto das cadeias aceitas por uma MT M é a linguagem de M , ou a linguagem reconhecida por M , denotada por $L(M)$

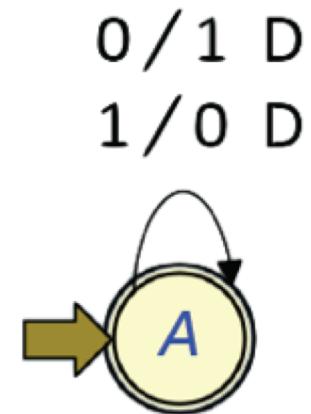
Entretanto, se $\omega \notin L(M)$, então M pode:

- **Parar no estado de rejeição, ou**
- **Entrar em loop**



Usos da MT

Implementação de uma função: dada uma palavra de **entrada**, a fita é **transformada** de acordo com uma regra preestabelecida; neste caso, o **conteúdo da fita** quando a máquina para deve ser observada como uma **saída**. Por exemplo, a MT à direita inverte uma entrada binária, i.e., transforma 0s em 1s e vice-versa



Reconhecedor: Dada uma palavra de entrada, ela faz parte da linguagem se e somente se a MT **parar** em um estado de **aceitação**

Linguagens Recursivamente Enumeráveis

Uma linguagem L é chamada de **Turing-reconhecível** se existe alguma MT M tal que $L = L(M)$, ou seja, se $\omega \in L$, então M para no estado de aceitação (reconhecedora)

As linguagens **Turing-reconhecíveis** são também chamadas de:

- Linguagens **recursivamente enumeráveis** (LRE); ou
- Linguagens **irrestritas** (geradas gramáticas irrestritas (tipo 0))

As gramáticas irrestritas são o tipo mais geral de se definir um conjunto de palavras recursivamente

Gramáticas Irrestritas

Qualquer gramática $G = (\Sigma, V, S, P)$ é irrestrita (ou Tipo 0)

Não existem restrições na forma das produções para as gramáticas desta classe, apenas

$$\alpha \rightarrow \beta$$

com $\alpha \neq \varepsilon$ e $\alpha, \beta \in (\Sigma \cup V)^*$

Exemplo: $G_1 = (\{S, A, B\}, \{a, b, c\}, P, S)$

$$P : \quad S \rightarrow aSBc \mid abc \mid \varepsilon$$

$$cB \rightarrow Bc$$

$$bB \rightarrow bb$$

Que corresponde a linguagem

$$L_1 = \{a^n b^n c^n : n \geq 0\}$$

Gramáticas Irrestritas

Qualquer gramática $G = (\Sigma, V, S, P)$ é irrestrita (ou Tipo 0)

Não existem restrições na forma das produções para as gramáticas desta classe, apenas

$$\alpha \rightarrow \beta$$

com $\alpha \neq \varepsilon$ e $\alpha, \beta \in (\Sigma \cup V)^*$

Exemplo: $G_1 = (\{S, A, B\}, \{a, b, c\}, P, S)$

$$P : \quad S \rightarrow aSBc \mid abc \mid \varepsilon$$

$$cB \rightarrow Bc$$

$$bB \rightarrow bb$$

Que corresponde a linguagem

$$L_1 = \{a^n b^n c^n : n \geq 0\}$$

$$\begin{aligned} S &\Rightarrow aSBc \\ &\Rightarrow aabcBc \\ &\Rightarrow aabBcc \\ &\Rightarrow aabbcc \end{aligned}$$

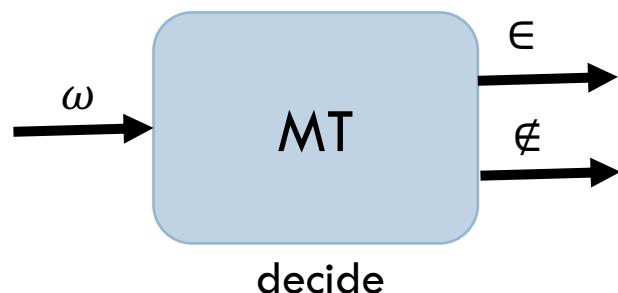
Linguagens Turing-Decidíveis

Ao executar uma MT Podemos ter três resultados ao processar ω :
(i) **aceita**; (ii) **rejeita**; ou (iii) **entra em loop**.

Pode ser difícil decidir se a MT está em loop ou se está demorando para processar a cadeia

As máquinas que sempre **param** são chamadas de “**decisores**” e são as mais interessantes!

Uma linguagem é chamada de **Turing-decidível** se existe alguma MT M que a **decide**

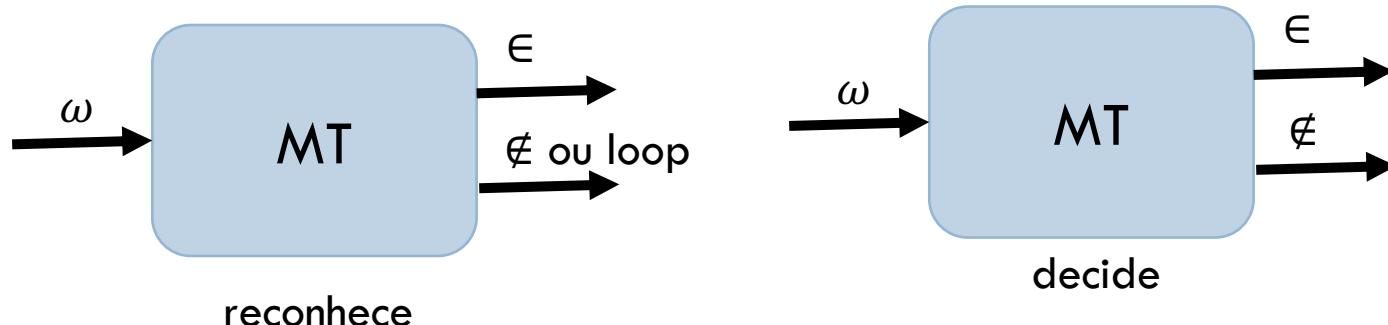


Linguagens Turing-Decidíveis

Decidir a linguagem L é “**melhor**” (mais forte) do que **reconhecer** L

Toda linguagem **Turing-decidível** é também **Turing-reconhecível**, mas não o inverso

Elas não foram incluídas na Hierarquia de Chomsky (estaria entre o Tipo 0 e o Tipo 1)



Hierarquia de Chomsky

Hierarquia de Chomsky	Gramática	Linguagem	Reconhecedor
Tipo 0	Irrestrita	Recursivamente Enumeráveis	Máquina de Turing
Tipo 1	Sensível ao Contexto	Sensíveis ao Contexto	Máquina de Turing com Fita Limitada
Tipo 2	Livre de Contexto	Livres de Contexto	Autômato de Pilha
Tipo 3	Linear	Regulares	Autômato Finito

