Introdução às Máquinas de Turing

O que um computador pode fazer?

- Reconhecer os strings em uma linguagem é um modo formal de expressar qualquer problema, e resolver um problema é uma expressão razoável daquilo que os computadores fazem.

- Importante: quais linguagens podem ser definidas por qualquer dispositivo computacional?

Incomputabilidade (indecidibilidade)

- Linguagens Regulares: reconhecidas por DFA, NFA, ε-NFA ou RE;
- Linguagens Livres de Contexto: reconhecida por CFG;
- Linguagens "Decidíveis": reconhecidas por um computador.


= Máquina de Turing

Máquinas de Turing (TM)

- É um formalismo para computadores;
- Embora uma máquina de Turing não se pareça em nada com um PC e, apesar de ser inviável fabricá-la e vendê-la, a TM é um modelo preciso daquilo que qualquer dispositivo físico de computação é capaz de fazer.

Linguagem

$\Sigma = \{a, b, c\}$

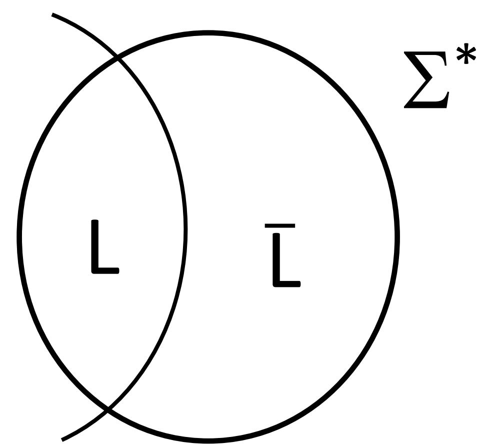
Σ^* = todas as strings que podem ser escritas com Σ

$= \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

$= \{\varepsilon\} \cup \{a,b,c\} \cup \{aa,ab,\ldots\} \cup \{\ldots\}$

$L \subseteq \Sigma^*$

$L = \{w \in \Sigma^* \mid \text{restrição}\}$



Por que os problemas indecidíveis têm de existir?

- Um problema pode ser definido como a pertinência de um string a uma linguagem.
- O número de linguagens diferentes sobre qualquer alfabeto de mais de um símbolo não é enumerável.
- Sendo os programas strings finitos sobre um alfabeto finito, são enumeráveis.
- Conclusão: existem infinitamente menos programas que problemas.

* A única razão pela qual a maioria dos problemas parece ser decidível é que raras vezes estamos interessados em problemas aleatórios (linguagem escolhida ao acaso). Mas, mesmo problemas simples e estruturados podem ser indecidíveis.

Problemas indecidíveis

- Problemas que nenhum computador pode resolver!!!
- Exemplo: Verificar se a primeira saída que um programa imprime é “hello, world”.
  - Dado como entrada um fonte de programa em ‘C’
    - file.c
  - Saída: **sim**, file.c imprime “hello, world”

    não, caso contrário

Problemas indecidíveis

- file.c

```
main(){
    printf("hello, world");
}
```

- file2.c

```
main(){
    printf("xyz");
}
```

Problemas indecidíveis

- file3.c

```
...
fermat(n){
    max = 3;
    while(true){
          for(a=1;a<max;a++)
                for(b=1;b<max;b++)
                      for(c=1;c<max;c++)
                         if (a<b) and (a^n+b^n = c^n)
                           printf("hello, world");
          max++;
    }
}
...
```

Problemas indecidíveis

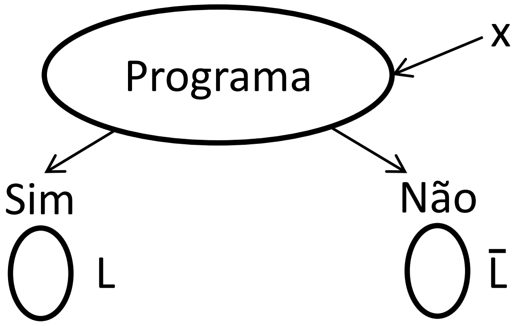
- Vários problemas podem ser convertidos em uma pergunta da forma:

  "esse programa, com essa entrada,

  imprime hello, world?"

- **Não** existe um programa capaz de examinar qualquer programa P e cada entrada I para P, e informar se P, executado com I como sua entrada, imprimiria hello, world.

Indecidibilidade



- $T = \{ L \mid L \subseteq \Sigma^* \} = 2^{\Sigma^*}$
- Pelo Teorema de Cantor: $|T| > |\text{programas}|$
- Conclusão: grande maioria das linguagens são **indecidíveis**;
  - Não existe um programa (DFA, PDA, Máquina de Turing, computador tradicional) que a resolva.

Indecidibilidade

- Problema: uma dado programa com uma determinada entrada imprime hello, world?
  - Não pode ser resolvido por um computador
    → problema indecidível

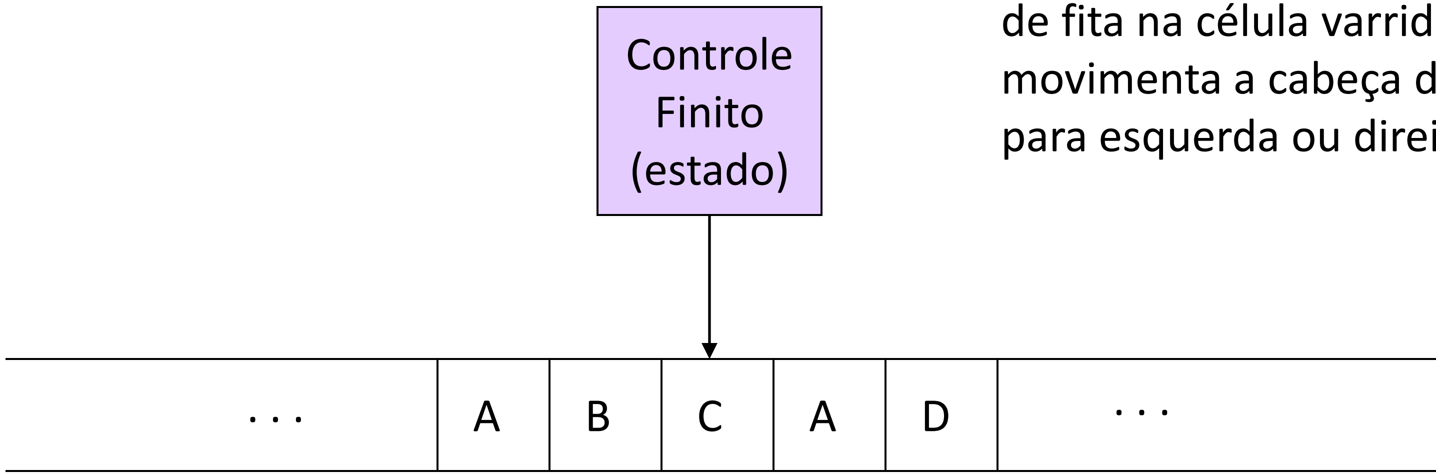
- Suponha que queremos descobrir se algum outro problema pode ou não ser resolvido por um computador.
- É possível tentar escrever um programa para resolvê-lo, mas se não conseguirmos descobrir como fazê-lo, podemos tentar provar que não existe tal programa.

Máquina de Turing

- Em 1936, Alan Turing propôs a Máquina de Turing como um modelo de "qualquer computação possível".
- O propósito da teoria de Máquinas de Turing é provar que certas linguagens específicas não possuem algoritmo para resolvê-las.
- Ferramenta para provar que as questões do dia a dia são indecidíveis ou intratáveis.
  - Intratáveis: uso de soluções heurísticas.
- Reduções podem ser usadas para provar questões de indecibilidade mais simples.

Uma Máquina deTuring

Movimento (basedo no estado e no símbolo de fita varrido): muda o estado, grava um símbolo de fita na célula varrida e movimenta a cabeça da fita para esquerda ou direita.



Fita infinita dividida em células contendo símbolos de fita escolhidos de um alfabeto finito

Por que Máquina de Turing?

- Por que não tratar de programas em C ou algo assim?
- Resposta: Você pode, mas é mais fácil provar a indecidibilidade sobre TM's, pois são um modelo muito simples de computador.
  - E ainda assim são tão poderosas como qualquer computador.
    - Mais poderosas ainda, na realidade, uma vez que têm memória infinita.

Máquina de Turing - Formalismo

Uma TM é descrita por:

1. Um conjunto finito de *estados* (Q).
2. Um *alfabeto de entrada* (Σ).
3. Um *alfabeto de fita* (Γ, $\Gamma \subseteq \Sigma$).
4. Uma *função de transição* (δ).
5. Um *estado inicial* (q_0).
6. Um *símbolo branco* (B, $\in \Gamma - \Sigma$).
   - Toda fita, exceto pela entrada, possui o símbolo branco inicialmente.
7. Um conjunto de *estados finais* ($F \subseteq Q$).

Convenções

- a, b, … são símbolos de entrada.
- …, X, Y, Z são símbolos de fita.
- …, w, x, y, z são strings de símbolo de entrada.
- α, β,… são strings de símbolos de fita.

A Função de Transição

- Os argumentos de δ são:
  1. Um estado q, em Q.
  2. Um símbolo de fita X em Γ.
- $\delta(q, X)$, se ele for definido, é uma tripla (p, Y, D).
  – p é o próximo estado.
  – Y é o novo símbolo de fita gravado na célula que está sendo varrida.
  – D é a *direção* em que a cabeça se move, L ou R.

Movimentos de um PDA

Se $\delta(q, X) = (p, Y, D)$ então, no estado q, varrendo ("scanning") X sob a cabeça da fita, a TM:

1. Mudará o estado para p.
2. Trocará X por Y na fita.
3. Moverá a cabeça uma célula na direção D.
    - D = L: mover para esquerda; D = R; mover para direita.

Exemplo: Máquina de Turing

- Esta TM varre sua entrada a direita, procurando por um 1.
- Se encontrar algum 1, muda para 0, vai para o estado final, e pára.
- Se alcança um branco, muda para 1 e move para esquerda.

Exemplo: Máquina de Turing – (2)

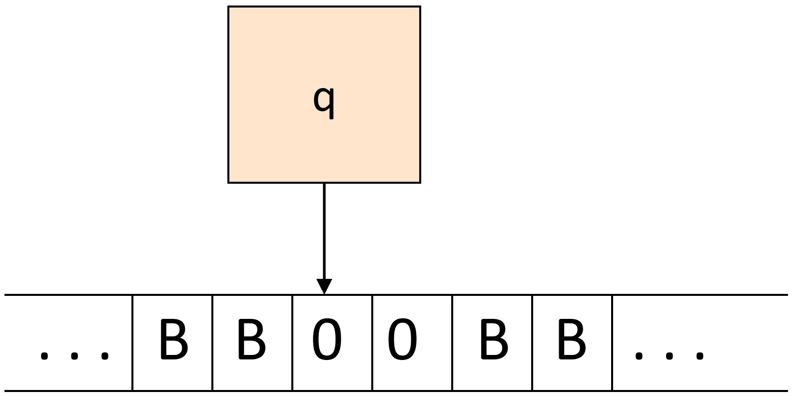
- Estados = {q (inicial), f (final)}.
- Símbolos de entrada = {0, 1}.
- Símbolos de fita = {0, 1, B}.
- $\delta(q, 0) = (q, 0, R)$.
- $\delta(q, 1) = (f, 0, R)$.
- $\delta(q, B) = (q, 1, L)$.

Simulação da TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$

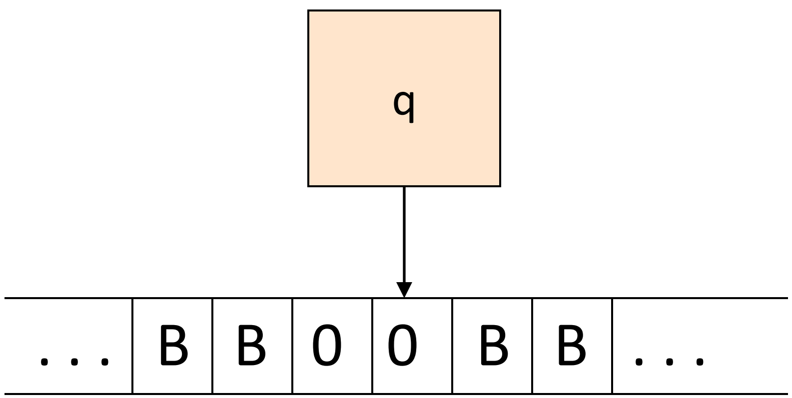


Simulação da TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$

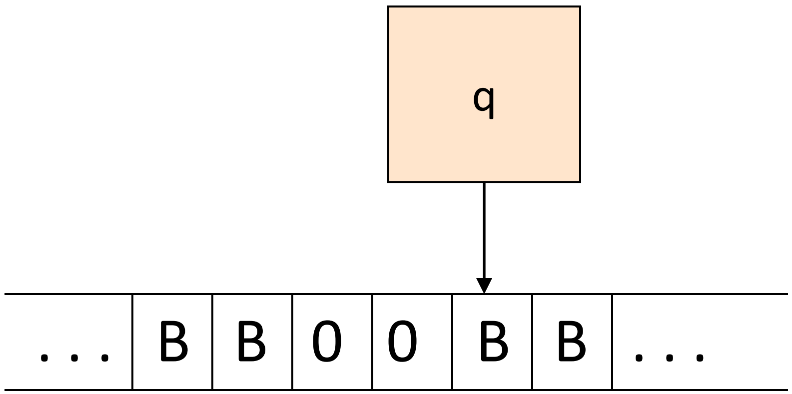


Simulação da TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$

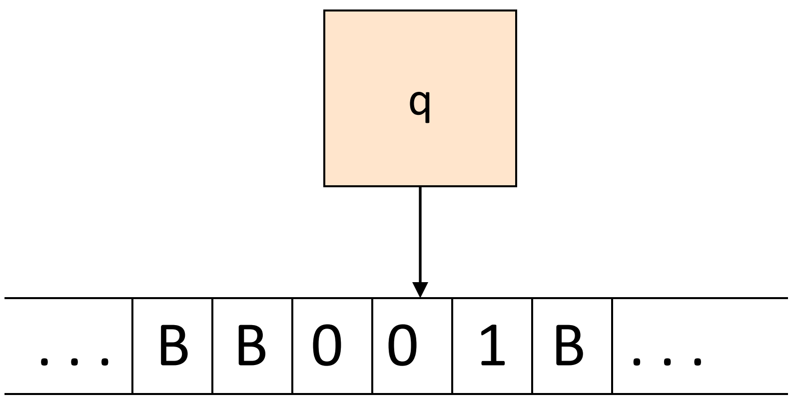


Simulação da TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$

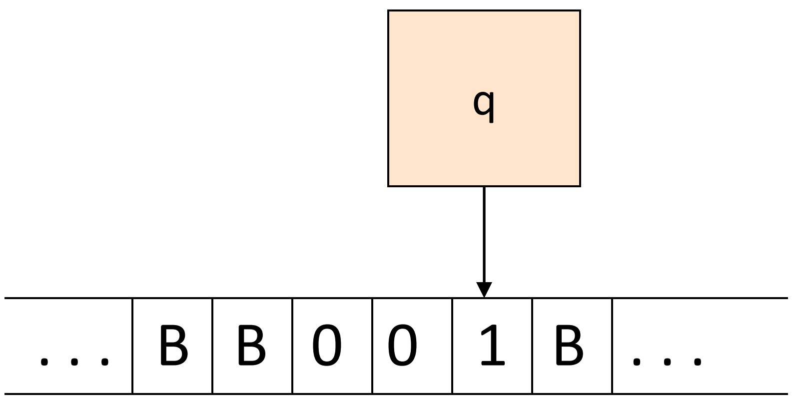


Simulação da TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$

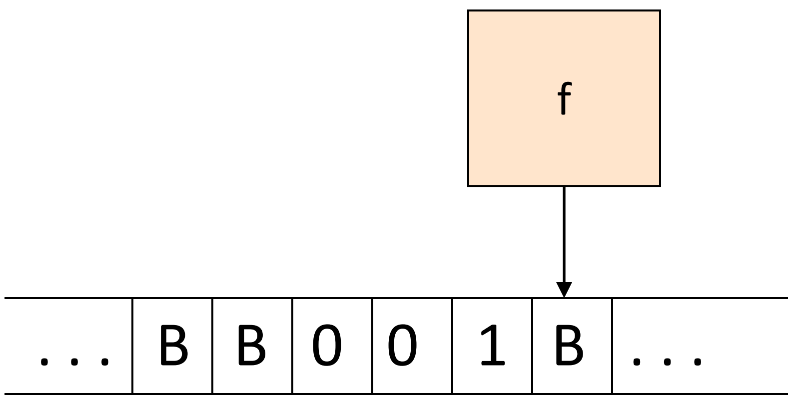


Simulação da TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$



Nenhum movimento é possível.
A TM pára e aceita.

Exemplo 2

$L(M) = \{0^n1^n \mid n \geq 1\}$

TM $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0,1\}, \{0,1,X,Y,B\}, \delta, q_0, B, \{q_4\})$

	Símbolo				
Estado	0	1	X	Y	B
q_0	(q_1, X, R)	-	-	(q_3, Y, R)	-
q_1	$(q_1, 0, R)$	(q_2, Y, L)	-	(q_1, Y, R)	-
q_2	$(q_2, 0, L)$	-	(q_0, X, R)	(q_2, X, L)	-
q_3	-	-	-	(q_3, Y, R)	(q_4, B, R)
q_4	-	-	-	-	-

Descrições Instantâneas de uma Máquina de Turing

- Inicialmente, uma TM tem uma fita consistindo de um string de símbolos de entrada cercado por brancos em ambas direções.
- A TM está no estado inicial, e a cabeça está no símbolo mais a esquerda.

TM ID's – (2)

- Uma ID é um string $\alpha q \beta$, onde $\alpha\beta$ é a fita entre os não-brancos mais à esquerda e mais à direita (inclusive).
- O estado q está imediatamente à esquerda do símbolo de fita varrido.

- Como exceção, se a cabeça estiver à esquerda do não-branco mais à esquerda ou à direita do não-branco mais à direita, algum prefixo ou sufixo de $\alpha\beta$ será branco.

TM ID's – (3)

- Como para PDA's podemos usar símbolos $\vdash$ e $\vdash^*$ para representar "um movimento" e "zero, um ou mais movimentos," respectivamente, sobre ID's.
- Exemplo: Os movimentos da TM do exemplo 2 são :

$$q00 \vdash 0q0 \vdash 00q \vdash 0q01 \vdash 00q1 \vdash 000f$$

Definição Formal de Movimentos

1. Se $\delta(q, Z) = (p, Y, R)$, então
   - $\alpha qZ\beta \vdash \alpha Yp\beta$
   - Se Z é o branco B, então $\alpha q \vdash \alpha Yp$
2. Se $\delta(q, Z) = (p, Y, L)$, então
   - Para algum X, $\alpha XqZ\beta \vdash \alpha pXY\beta$
   - Novamente, $qZ\beta \vdash pBY\beta$

Linguagens de uma TM

- Uma TM define uma linguagem pelo estado final.
- $L(M) = \{w \mid q_0 w \vdash^* I$, onde I é um ID com um estado final$\}$.
- Ou, uma TM pode aceitar uma linguagem por parada.
- $H(M) = \{w \mid q_0 w \vdash^* I$, e não há nenhum movimento possível da ID I$\}$.

Equivalência de Aceitação e Parada

1. Se $L = L(M)$, então há uma TM M' tal que $L = H(M')$.
2. Se $L = H(M)$, então há uma TM M'' tal que $L = L(M'')$.

Prova 1: Aceitação -> Parada

- Modifique M para se tornar M’ como a seguir:
  1. Para cada estado de aceitação de M, remova qualquer movimentos, então M’ pára naquele estado.
  2. Evitar que M’ pare acidentalmente.
     - Introduza um novo estado s, o qual se desloca sempre para direita; isto é $\delta(s, X) = (s, X, R)$ para todos símbolos X.
     - Se q é não aceito, e $\delta(q, X)$ é indefinido, faça $\delta(q, X) = (s, X, R)$.

Prova de 2: Parada -> Aceitação

- Modifique M para se tornar M” como a seguir:
  1. Introduza um novo estado f, o único estado de aceitação de M”.
  2. f não tem movimentos.
  3. Se $\delta(q, X)$ é indefinido para qualquer estado q e símbolo X, defina este por $\delta(q, X) = (f, X, R)$.

Linguagens Recursivamente Enumeráveis

- Podemos ver que as classes de linguagens definidas por TM's usando estado final e parada são as mesmas.
- Esta classe de linguagens é chamada *linguagens recursivamente enumeráveis*.
  – Por que? O termo, na verdade, antecede a máquina de Turing e refere-se a uma outra noção de computação de funções.

Linguagens Recursivas

- Um *algoritmo* é uma TM a qual seja garantida parar (independente de aceitação ou não).
- Se L = L(M) para alguma TM M que é um algoritmo, dizemos que L é uma *linguagem recursiva*.
  - Por que? Mais uma vez, não pergunte; é um termo histórico.

Exemplo: Linguagens Recursivas

- Toda CFL é uma linguagem recursiva.
- Toda linguagem regular é uma CFL (pense no DFA como sendo um PDA que ignora sua pilha); portanto toda linguagem regular é recursiva.
- Quase tudo que você pode pensar é recursiva.

Exercícios

1. Mostre as ID's da máquina de Turing da linguagem 0^n1^n, $n \geq 1$, se a fita de entrada contêm:
   a) 00
   b) 000111
   c) 00111

Exercícios

2. Projete máquinas de Turing para as seguintes linguagens:
   a) Conjunto de strings com um número igual de 0's e 1's.
   b) $\{a^n b^n c^n \mid n \geq 1\}$.
   c) $\{ww^r \mid w$ é qualquer string de 0's e 1's.

Exercícios

3. Considere a máquina de Turing

   $M = (\{q_0,q_1,q_2,q_f\}, \{0,1\}, \{0,1,B\}, \delta, q_0, B, \{q_f\})$

   Descreva de maneira informal, mas clara, a linguagem L(M) se δ consistir nos seguintes conjuntos de regras:

   a) $\delta(q_0, 0) = (q_1, 1, R)$; $\delta(q_1, 1) = (q_0, 0, R)$; $\delta(q_1, B) = (q_f, B, R)$;

   b) $\delta(q_0, 0) = (q_0, B, R)$; $\delta(q_0, 1) = (q_1, B, R)$; $\delta(q_1, 1) = (q_1, B, R)$; $\delta(q_1, B) = (q_f, B, R)$;

   c) $\delta(q_0, 0) = (q_1, 1, R)$; $\delta(q_1, 1) = (q_2, 0, L)$; $\delta(q_2, 1) = (q_0, 1, R)$; $\delta(q_1, B) = (q_f, B, R)$;