SBVLIFA: Linguagens Formais e Autômatos

Aula 10: Introdução às Máquinas de Turing

Bacharelado em Ciência da Computação
Prof. Dr. David Buzatto

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista

Introdução às Máquinas de Turing

- Linguagens Sensíveis ao Contexto, Recursivas e Recursivamente Enumeráveis

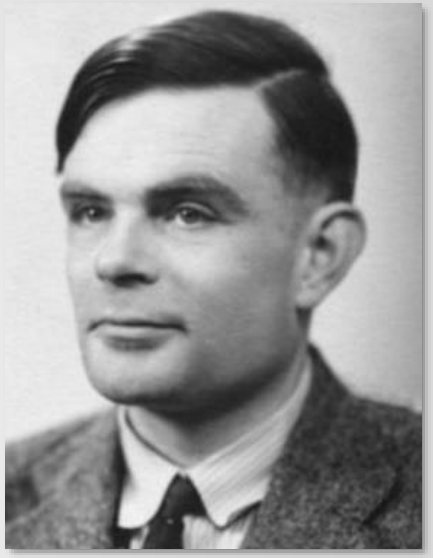
Tipo	Classe de Linguagens	Modelo de Gramática	Modelo de Reconhecedor
0	Recursivamente enumeráveis	Irrestrita	Máquina de Turing
-	Recursivas	Não há	Máquina de Turing que sempre para
1	Sensíveis ao contexto	Sensível ao contexto	Máquina de Turing com fita limitada
2	Livres de contexto	Livre de contexto	Autômato de pilha
3	Regulares	Linear (direita ou esquerda)	Autômato finito

Introdução às Máquinas de Turing

- Até o momento nosso interesse girou em torno de classes simples de linguagens e nas maneiras que elas podem ser usadas para resolver problemas relativamente restritos, como analisar protocolos, fazer buscas em textos ou analisar programas. Inclusive elas serão a base para nosso trabalho na disciplina de Construção de Compiladores do quinto semestre do curso;
- A partir de agora mudaremos nosso foco, começando a examinar a questão de quais linguagens podem ser definidas por qualquer dispositivo computacional.

Introdução às Máquinas de Turing

- Essa questão é equivalente a perguntar o que os computadores podem fazer, pois reconhecer as strings em uma linguagem é um modo formal de expressar qualquer problema, sendo que, resolver um problema, é uma representação razoável daquilo que os computadores fazem;
- Começaremos informalmente mostrando que existem problemas específicos que não podemos resolver usando um computador, sendo que esses problemas são chamados de indecidíveis. Nesta aula aprenderemos um formalismo para a representação de computadores, chamado de Máquina de Turing (*Turing Machine* – TM).



Alan Mathison Turing

Introdução às Máquinas de Turing

Problemas que os Computadores não Podem Resolver

- **Problema específico:** A primeira saída que um programa imprime é `hello, world`?
  - Se sim, quando?
- O problema de não saber quando algo ocorrerá, se é que ocorrerá, é a causa final da nossa incapacidade de saber o que um programa faz.

Introdução às Máquinas de Turing

Programas que imprimem `hello, world`

```c
int main() {
    printf( "hello, world\n" );
    return 0;
}
```

Último teorema de Fermat: "É impossível para um cubo ser escrito como a soma de dois cubos ou uma quarta potência ser escrita como a soma de duas quartas potências ou, em geral, para qualquer número que é uma potência maior do que a segunda, ser escrito como a soma de duas potências com o mesmo expoente. Descobri uma demonstração maravilhosa desta proposição que, no entanto, não cabe nas margens deste livro."

Conjecturado em 1637, demonstrado em 1993 por Andrew Wiles e corrigido em 1994 em conjunto com Richard Taylor.

```c
int exp( int b, int e ) {
    int r = 1;
    for ( int i = 1; i <= e; i++ ) {
        r *= b;
    }
    return r;
}

int main() {
    int total = 3;
    int n;
    scanf( "%d", &n );
    while ( true ) {
        for ( int x = 1; x <= total - 2; x++ ) {
            for ( int y = 1; y <= total - x - 1; y++ ) {
                int z = total - x - y;
                if ( exp( x, n ) + exp( y, n ) == exp( z, n ) ) {
                    printf( "hello, world\n" );
                }
            }
        }
        total++;
    }
    return 0;
}
```

Imprime `hello, world` para um n qualquer caso a igualdade $x^n + y^n = z^n$ exista

Introdução às Máquinas de Turing

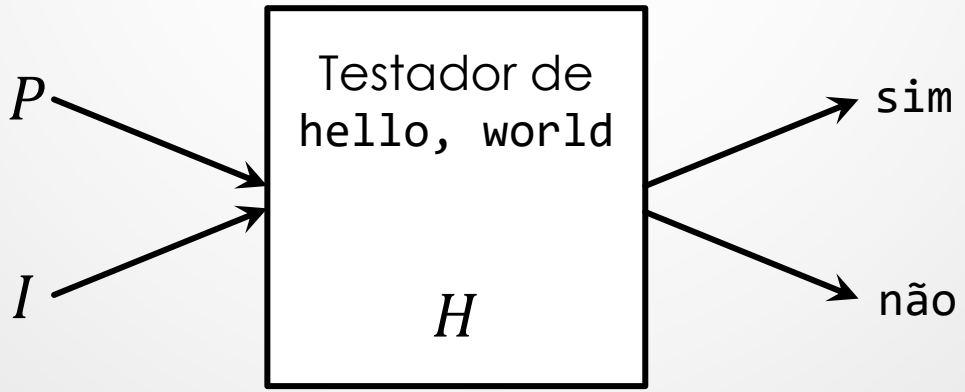
Programas que imprimem `hello, world`

- **Definição do problema de `hello, world`:** descobrir se um dado programa em C, com uma determinada entrada, imprime `hello, world` como os 12 primeiros caracteres que são impressos por ele;
- Não somos capazes de escrever um programa que examine qualquer programa P e cada entrada I para P e que nos informe se P, executando I como entrada, imprimirá `hello, world`;

Provaremos que tal programa não existe!

Introdução às Máquinas de Turing
O Testador Hipotético de `hello, world`

- A prova da impossibilidade de se fazer o teste de `hello, world` é uma prova por contradição;
- Supomos que existe um programa, chamado de H, que recebe como entrada um programa P e uma entrada I e informa se P com a entrada I imprime `hello, world`.



Introdução às Máquinas de Turing

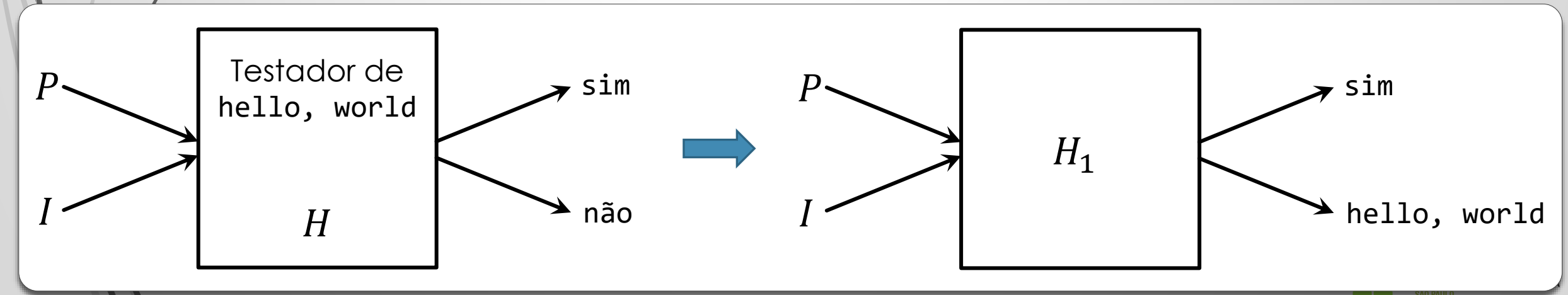
O Testador Hipotético de `hello, world`

- Se um problema tiver um algoritmo como H, que sempre informa corretamente se uma instância do problema tem a resposta `sim` ou `não`, então o problema é dito decidível. Caso contrário, o problema é indecidível;
- Nossa meta é provar que H não existe; isto é, que o problema `hello, world` é indecidível;
- Para a prova, faremos diversas mudanças em H, construindo eventualmente um programa inter-relacionado chamado de H_2 que mostraremos que não existe;
- Se pudermos mostrar que não existe um "testador de `hello, world`" para esses programas restritos, então certamente não haverá nenhum testador desse tipo que possa funcionar para uma classe mais ampla de programas;
- Toda a saída é baseada em caracteres usando `printf`.

Introdução às Máquinas de Turing

O Testador Hipotético de `hello, world`

- Supomos que H existe;
- Nossa primeira modificação consiste em mudar a saída da instrução que imprime `não` do programa H para `hello, world`, criando assim o programa H_1;
- H_1 se comporta como H, exceto por imprimir `hello, world` exatamente quando H imprimiria `não`.

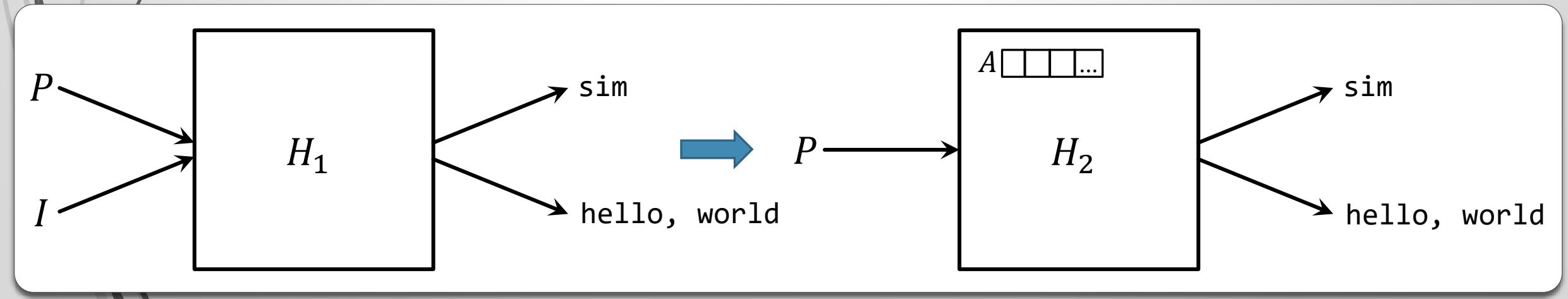


Introdução às Máquinas de Turing
O Testador Hipotético de `hello, world`

- Nossa próxima transformação, mais complicada, é essencialmente a ideia que permitiu a Alan Turing provar seu resultado de indecidibilidade sobre máquinas de Turing;
- Como estamos realmente interessados em programas que recebem outros programas como entrada e informam algo a respeito deles, restringiremos H_1 de forma que ele:
  a) Recebe somente a entrada P, não P e I;
  b) Pergunte o que P faria se sua entrada fosse seu próprio código; isto é, o que H_1 faria tendo como entrada o programa P e também P sendo passado como I;

Introdução às Máquinas de Turing
O Testador Hipotético de `hello, world`

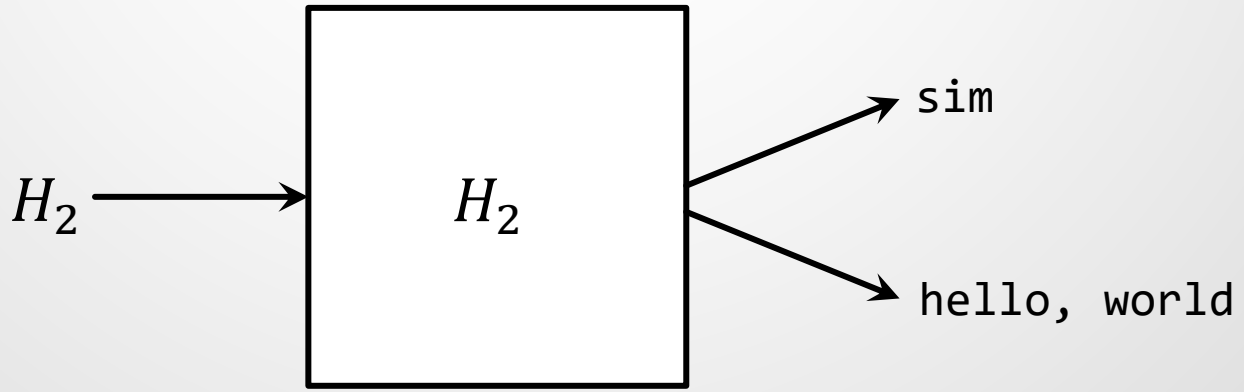
- As modificações que devemos executar em H_1 para produzir o programa H_2 são:
  1. Primeiramente, H_2 lê a entrada P inteira e a armazena em um array A, que ele aloca com um `malloc` para esse propósito;
  2. Posteriormente, H_2 simula H_1 mas, em todos os momentos em que H_1 leria a entrada de P ou I, H_2 lê a entrada da cópia armazenada em A. Para controlar a proporção de P e I que H_1 lesse, H_2 poderia manter dois cursores marcando as posições em A.



Introdução às Máquinas de Turing

O Testador Hipotético de `hello, world`

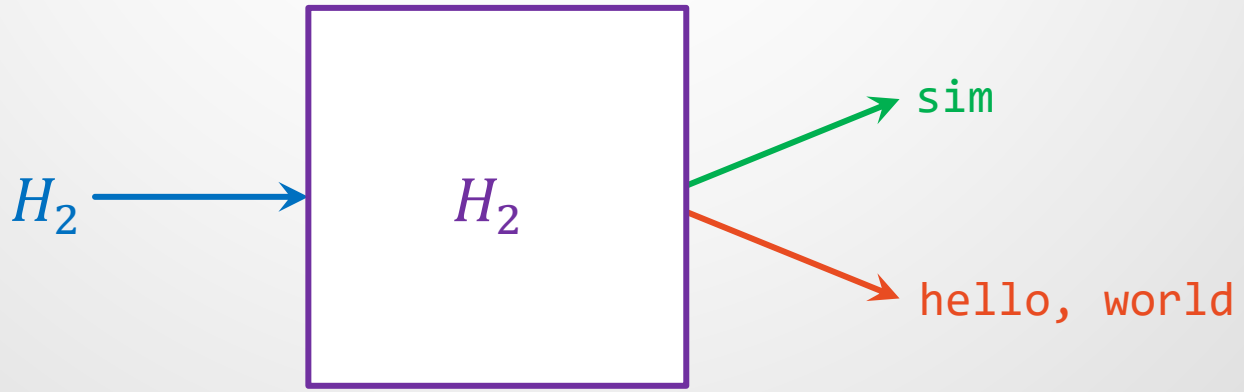
- Agora estamos prontos para provar que H_2 não pode existir e desse modo, H_1 não existe e, da mesma forma, H não existe;
- O núcleo da argumento é prever o que H_2 faz quando recebe a si mesmo como entrada, lembrando que:
  - H_2, dado qualquer programa P como entrada, gera a saída `sim` se P imprime `hello, world` quando recebe a si mesmo como entrada;
  - H_2 imprime `hello, world` se P, recebendo a si mesmo como entrada, não imprime `hello, world` como sua primeira saída;



Introdução às Máquinas de Turing

O Testador Hipotético de `hello, world`

- Supondo que $H_2\ (bloco)$ gera a saída `sim`, quer dizer que $H_2\ (entrada)$ imprimiu `hello, world`;
  - Entretanto, acabamos de supor que a saída que $H_2\ (bloco)$ gera nessa situação é `sim`, não `hello, world`;
- Agora, se $H_2\ (bloco)$ imprime `hello, world`, quer dizer que $H_2\ (entrada)$ imprimiu `sim`;
  - Note que seja qual for a saída que supomos que H_2 gera, podemos demonstrar que ele gera a outra saída!



Introdução às Máquinas de Turing
O Testador Hipotético de `hello, world`

- Essa situação é paradoxal e concluímos que H_2 não pode existir, contradizendo a suposição que H existe;

Provamos assim que nenhum programa H pode saber se um dado programa P com a entrada I imprime ou não `hello, world` como sua primeira saída. □

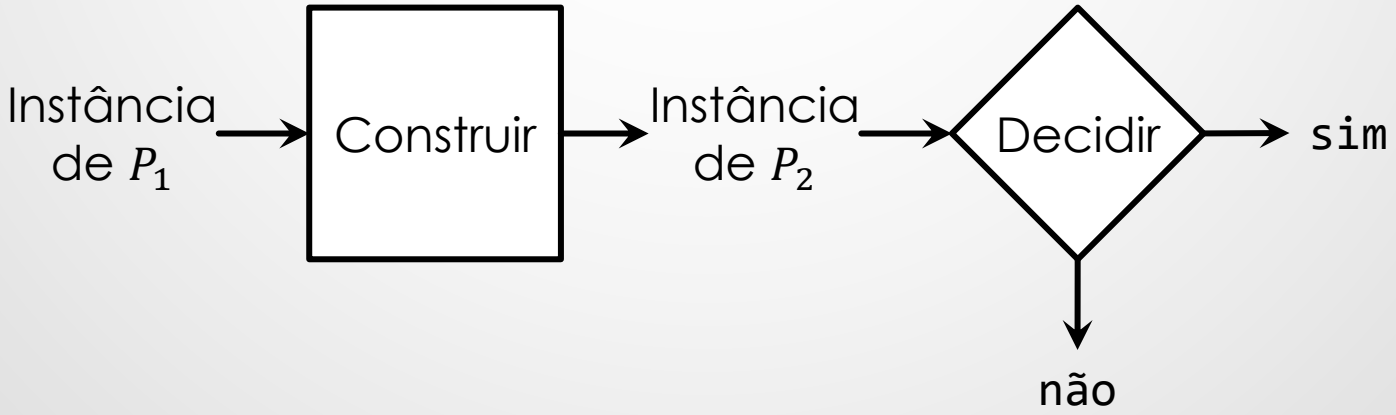
- O problema de `hello, world` que acabamos de provar como indecidível é um exemplo de um problema denominado "O Problema da Parada" (*The Halting Problem*);
- Sugestão de vídeo: *Turing & The Halting Problem*

Introdução às Máquinas de Turing
Reduzindo um Problema a Outro

- Já sabemos que, informalmente, um problema que não pode ser resolvido por um computador é chamado de indecidível;
- E se quiséssemos 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, poderemos tentar provar que não existe tal programa;
- A prova poderia se dar de forma semelhante ao problema do `hello, world`, ou seja, supondo que existe um programa para resolvê-lo e desenvolver um programa paradoxal que tem que executar duas ações contraditórias como o programa H_2.

Introdução às Máquinas de Turing
Reduzindo um Problema a Outro

- No entanto, a partir do momento que sabemos que um programa é indecidível, não precisamos mais provar a situação paradoxal, sendo suficiente mostrar que, se pudéssemos resolver o novo problema, então poderíamos usar essa solução para resolver um problema que já sabemos ser indecidível;
- A estratégia é sugerida no diagrama abaixo;
- Essa técnica é chamada de **redução de P_1 a P_2**.



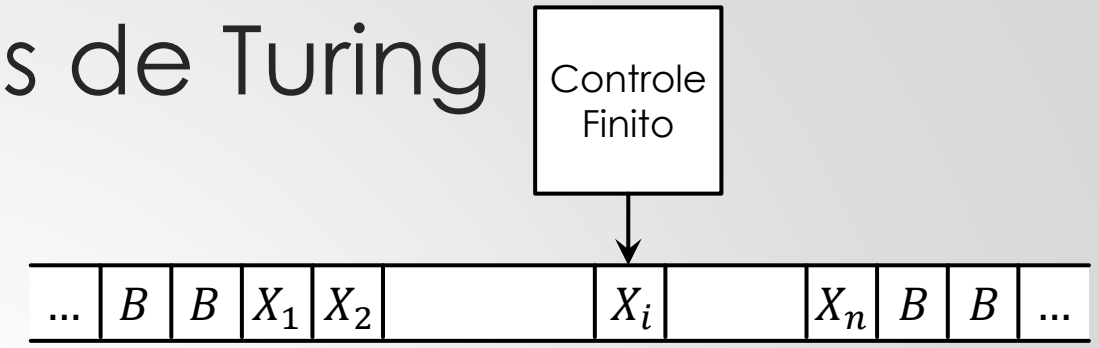
Introdução às Máquinas de Turing

A Máquina de Turing

- Com o que temos até agora seria muito difícil conseguirmos reduzir o problema de `hello, world` à questão de saber se uma gramática é ambígua, pois tais problemas pertencem a domínios não relacionados;
- Precisamos reconstruir nossa teoria da indecidibilidade usando um modelo MUITO mais simples de computador, chamado de Máquina de Turing (TM), ao invés de continuarmos a usar como base programas em C ou outras linguagens;
- A TM é essencialmente um Autômato Finito (*Finite Automata* – FA) que possui uma única fita de comprimento infinito na qual ele pode ler e gravar dados;
- Mais simples que um programa em C sendo executado em um computador real, não é?

Introdução às Máquinas de Turing

A Máquina de Turing



- Inicialmente a **entrada** é colocada na fita;
- Todas as outras células da fita, tanto à esquerda quanto à direita, contém inicialmente um símbolo especial chamado de **branco** (B), que é um **símbolo de fita**, mas não um símbolo de entrada;
- Há um **controle finito** que pode se encontrar em qualquer estado de um conjunto finito de estados;
- Existe uma **cabeça da fita** que fica sempre posicionada em uma das células da fita e dizemos que TM está **varrendo (*scanning*)** essa célula. Inicialmente, a cabeça da fita se encontra na célula mais à esquerda que contém a entrada;
- Um **movimento** da TM é uma função do estado do controle finito e do símbolo da fita varrido. Em um movimento, a TM:
  1. Mudará de estado;
  2. Gravará um símbolo na célula varrida;
  3. Movimentará, obrigatoriamente, a cabeça da fita para a esquerda ou para a direita.

Introdução às Máquinas de Turing

A Máquina de Turing

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

- **Definição Formal:**
  - TM: máquina de Turing determinística, uma 7-upla, onde:
    - Q: conjunto finito de estados;
    - Σ: conjunto finito de símbolos de entrada (alfabeto);
    - Γ: conjunto finito de símbolos de fita, tal que $\Sigma \subset \Gamma$;
    - δ: função de transição, na forma:

      $\delta(q, X) \rightarrow (p, Y, D)$, ou seja, $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times D$, onde:

      - q é um estado;
      - X é um símbolo de fita;
      - p é o próximo estado;
      - Y é o símbolo contido em Γ, gravado na célula que está sendo varrida, que substitui o símbolo que estava nessa célula;
      - D é uma direção ou sentido, sendo L (esquerda) ou R (direita), informando o sentido em que a cabeça se move.

    - q_0: estado inicial, tal que $q_0 \in Q$;
    - B: o símbolo branco, tal que $B \in \Gamma \wedge B \notin \Sigma$. Ele aparece em todas as células da fita, com exceção das células usadas para conter os símbolos de entrada;
    - F: conjunto de estados finais ou de aceitação, tal que $F \subseteq Q$

Introdução às Máquinas de Turing
A Máquina de Turing – Descrições Instantâneas (ID)

- A ID de uma TM representa a configuração atual da TM;
- É representada pela string $X_1X_2 \dots X_{i-1}qX_iX_{i+1} \dots X_n$, onde:
  - q é o estado da máquina de Turing;
  - A cabeça da fita que está varrendo o i-ésimo símbolo a partir da esquerda;
  - $X_1X_2 \dots X_n$ é a parte da fita entre o não-branco mais à esquerda e o não-branco mais à direita, com 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 $X_1X_2 \dots X_n$ será branco e i será 1 ou n, respectivamente.
- Descrevemos movimentos da TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, pela notação $\underset{M}{\vdash}$ ou $\vdash$ quando a TM for subentendida. Também usamos $\overset{*}{\underset{M}{\vdash}}$ ou simplesmente $\overset{*}{\vdash}$ para indicar zero ou mais movimentos da TM.

Introdução às Máquinas de Turing

A Máquina de Turing – Descrições Instantâneas (ID)

- Suponha que $\delta(q, X_i) = (p, Y, L)$, ou seja, um movimento para a esquerda, então:

$$X_1 X_2 \ldots X_{i-1} q X_i X_{i+1} \ldots X_n \underset{M}{\vdash} X_1 X_2 \ldots X_{i-2} p X_{i-1} Y X_{i+1} \ldots X_n$$

- Há duas exceções importantes:

1. Se $i = 1$, então M se move para o branco à esquerda de X_1 que será Y:

$$q X_1 X_2 \ldots X_n \underset{M}{\vdash} p B Y X_2 \ldots X_n$$

2. Se $i = n$ e $Y = B$, então o símbolo B gravado sobre X_n se junta à sequência de brancos finais:

$$X_1 X_2 \ldots X_{n-1} q X_n \underset{M}{\vdash} X_1 X_2 \ldots X_{n-2} p X_{n-1}$$

Introdução às Máquinas de Turing

A Máquina de Turing – Descrições Instantâneas (ID)

- Agora, suponha que $\delta(q, X_i) = (p, Y, R)$, ou seja, um movimento para a direita, então:

$$X_1 X_2 \ldots X_{i-1} q X_i X_{i+1} \ldots X_n \underset{M}{\vdash} X_1 X_2 \ldots X_{i-1} Y p X_{i+1} \ldots X_n$$

- Há duas exceções importantes:

1. Se $i = n$, então M se move para branco à direita de X_n que será Y:

$$X_1 X_2 \ldots X_{n-1} q X_n \underset{M}{\vdash} X_1 X_2 \ldots X_{n-1} Y p B$$

2. Se $i = 1$ e $Y = B$, então o símbolo B gravado sobre X_1 se junta à sequência de brancos iniciais:

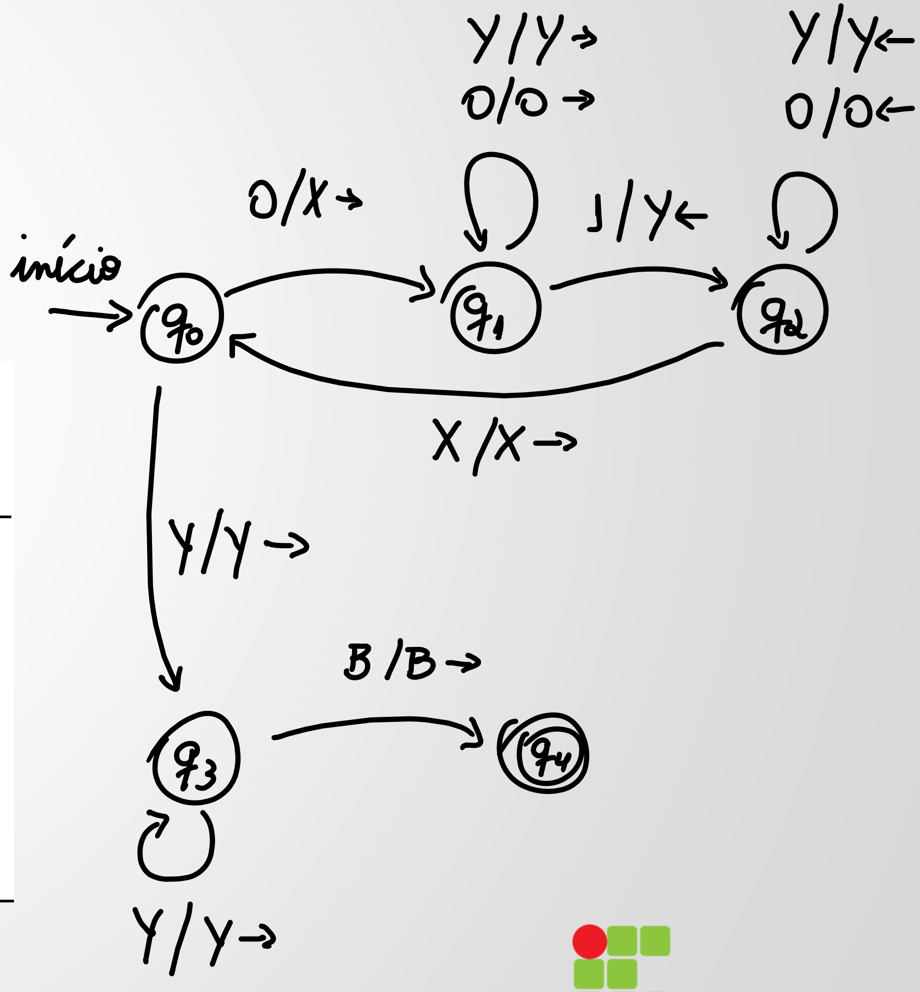
$$q X_1 X_2 \ldots X_n \underset{M}{\vdash} p X_2 \ldots X_n$$

Introdução às Máquinas de Turing

A Máquina de Turing

- **Exemplo:** TM que aceita $L = \{0^n 1^n \mid n \geq 1\}$
- $M = (\underbrace{\{q_0, q_1, q_2, q_3, q_4\}}_{Q}, \underbrace{\{0, 1\}}_{\Sigma}, \underbrace{\{0,1, X, Y, B\}}_{\Gamma}, \delta, q_0, B, \underbrace{\{q_4\}}_{F})$
- δ:

	símbolo na fita				
estado	0	1	X	Y	B
$\rightarrow q_0$	(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, Y, L)	$-$
q_3	$-$	$-$	$-$	(q_3, Y, R)	(q_4, B, R)
$* q_4$	$-$	$-$	$-$	$-$	$-$



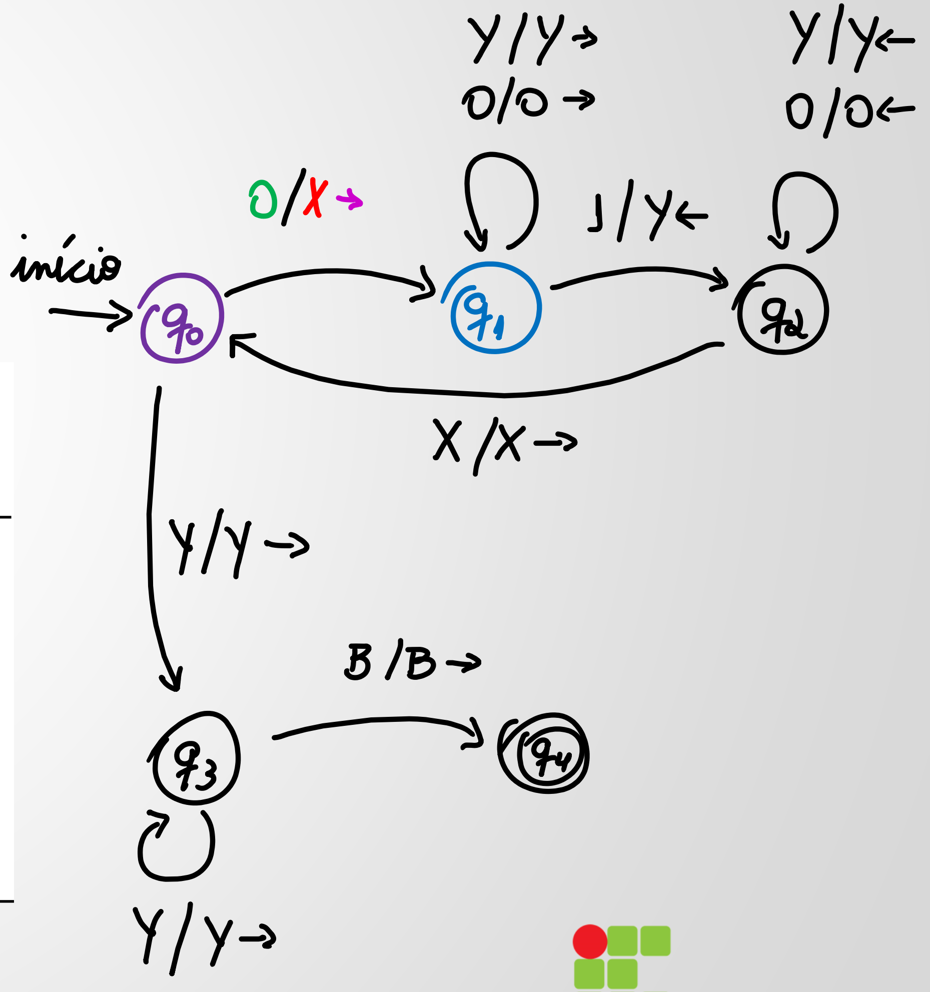
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista

Introdução às Máquinas de Turing

A Máquina de Turing

- **Exemplo:** TM que aceita $L = \{0^n 1^n \mid n \geq 1\}$
- $M = (\underbrace{\{q_0, q_1, q_2, q_3, q_4\}}_{Q}, \underbrace{\{0, 1\}}_{\Sigma}, \underbrace{\{0,1, X, Y, B\}}_{\Gamma}, \delta, q_0, B, \underbrace{\{q_4\}}_{F})$
- δ:

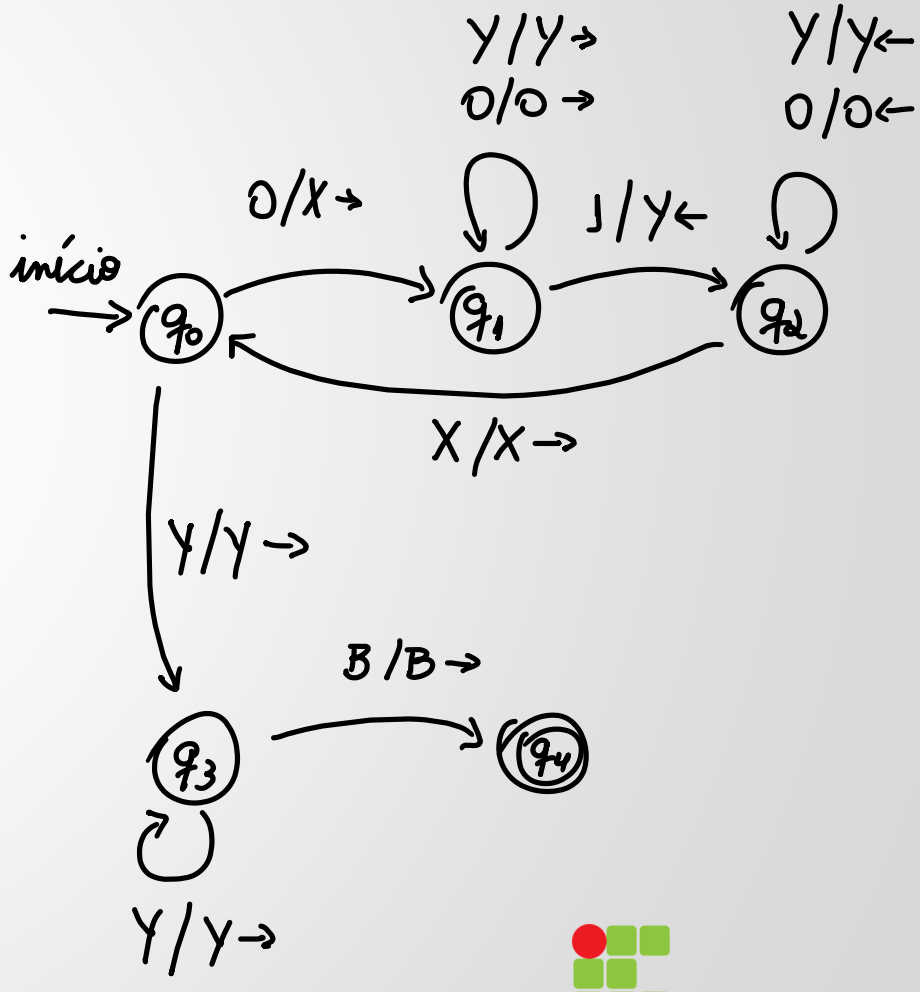
estado	0	1	X	Y	B
	símbolo na fita				
$\rightarrow q_0$	(q_1, X, R)	$-$	$-$	(q_3, Y, R)	$-$
q_1	$(q_1, 0, R)$	(q_2, Y, L)	$-$	(q_1, Y, R)	$-$
q_2	$(q_2, 0, L)$	$-$	(q_0, X, R)	(q_2, Y, L)	$-$
q_3	$-$	$-$	$-$	(q_3, Y, R)	(q_4, B, R)
$* q_4$	$-$	$-$	$-$	$-$	$-$



Introdução às Máquinas de Turing
A Máquina de Turing

- **Exemplo:** TM que aceita $L = \{0^n 1^n \mid n \geq 1\}$
- $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0,1, X, Y, B\}, \delta, q_0, B, \{q_4\})$
- Computação de aceitação por M:
  - Entrada (copiada na fita): 0011
  - ID inicial de M: $q_0 0011$
  - A sequência inteira de movimentos de M é:

$$q_0 0011 \vdash X q_1 011 \vdash X 0 q_1 11 \vdash X q_2 0 Y 1 \vdash q_2 X 0 Y 1 \vdash$$
$$X q_0 0 Y 1 \vdash X X q_1 Y 1 \vdash X X Y q_1 1 \vdash X X q_2 Y Y \vdash X q_2 X Y Y \vdash$$
$$X X q_0 Y Y \vdash X X Y q_3 Y \vdash X X Y Y q_3 B \vdash X X Y Y B q_4 B$$

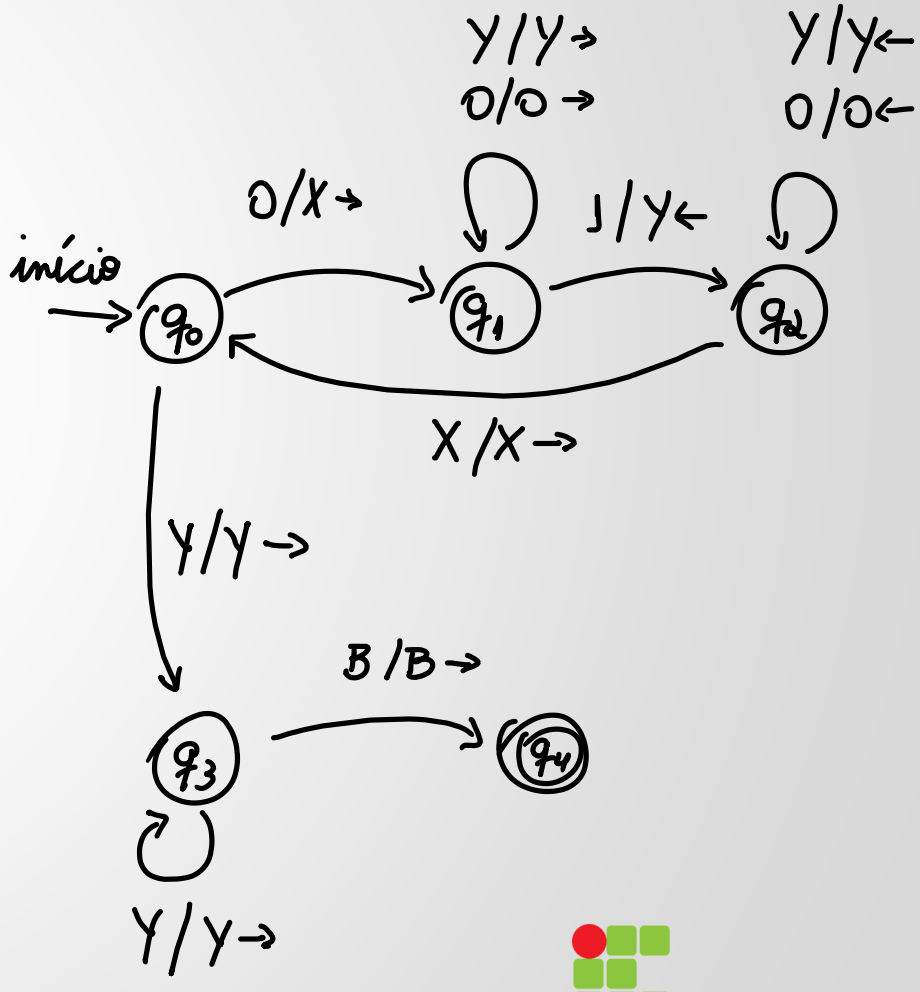


Introdução às Máquinas de Turing

A Máquina de Turing

- **Exemplo:** TM que aceita $L = \{0^n 1^n \mid n \geq 1\}$
- $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0,1, X, Y, B\}, \delta, q_0, B, \{q_4\})$
- Computação de não aceitação por M:
  - Entrada (copiada na fita): 0010
  - ID inicial de M: $q_0 0010$
  - A sequência inteira de movimentos de M é:

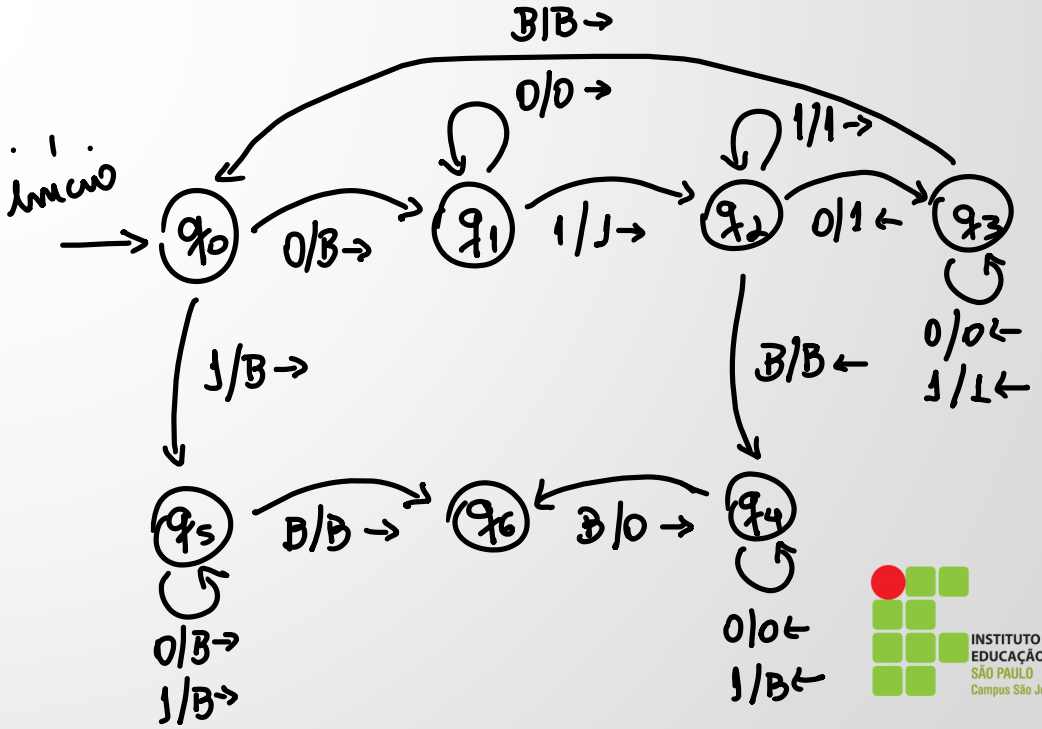
$$q_0 0010 \vdash X q_1 010 \vdash X 0 q_1 10 \vdash X q_2 0 Y 0 \vdash q_2 X 0 Y 0 \vdash$$
$$X q_0 0 Y 0 \vdash X X q_1 Y 0 \vdash X X Y q_1 0 \vdash X X Y 0 q_1 B$$



Introdução às Máquinas de Turing
A Máquina de Turing

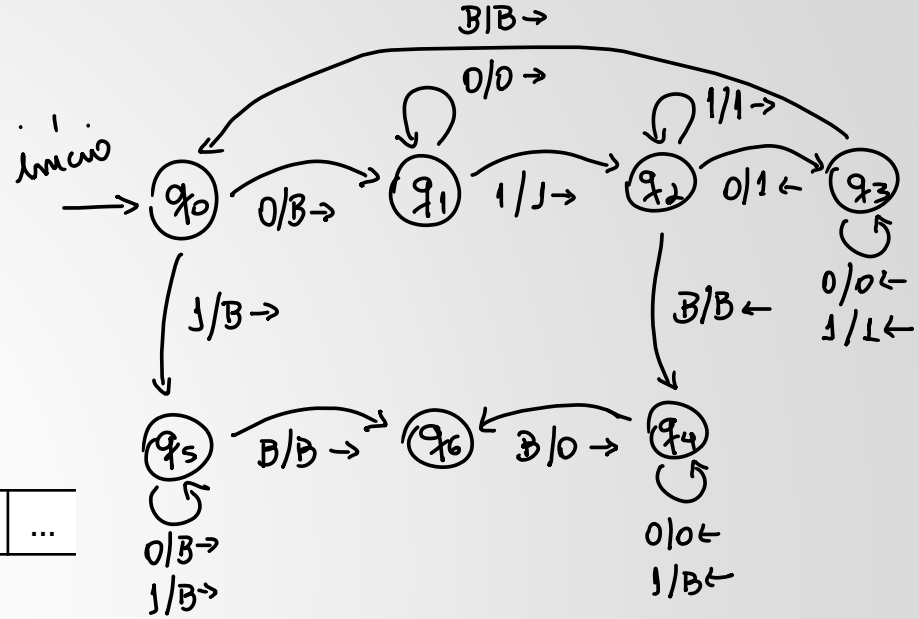
- **Exemplo:** TM que calcula a função $\dot{-}$ (*monus* ou subtração própria) que é definida como:
  - $m \dot{-} n = \max(m - n, 0)$, ou seja $m \dot{-} n$ é $m - n$ se $m \geq n$ e 0 se $m < n$;
- $M = (\{q_0, q_1, \ldots, q_6\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B)$
- Note que não há F, pois essa TM não é usada para aceitar entradas;
- M começará com uma fita que consiste em $0^m 10^n$ cercada por brancos e irá parar com $0^{m \dot{-} n}$ em sua fita, cercada por brancos.
- δ:

	símbolo na fita		
estado	0	1	B
$\rightarrow q_0$	(q_1, B, R)	(q_5, B, R)	–
q_1	$(q_1, 0, R)$	$(q_2, 1, R)$	–
q_2	$(q_3, 1, L)$	$(q_2, 1, R)$	(q_4, B, L)
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_0, B, R)
q_4	$(q_4, 0, L)$	(q_4, B, L)	$(q_6, 0, R)$
q_5	(q_5, B, R)	(q_5, B, R)	(q_6, B, R)
q_6	–	–	–

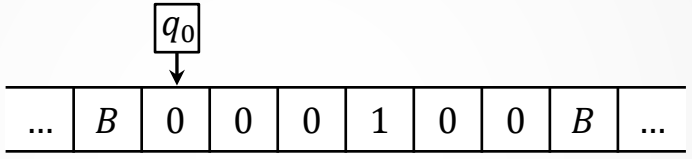


Introdução às Máquinas de Turing

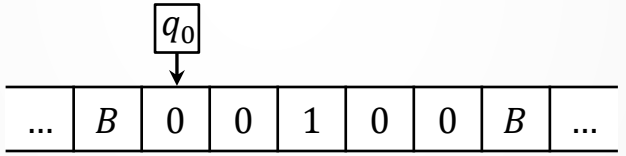
A Máquina de Turing



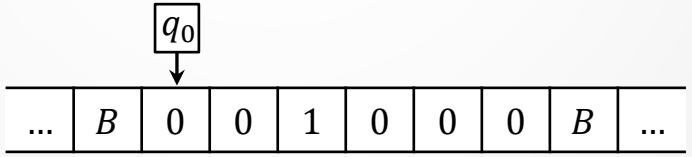
- Vamos testar!
  - $3 \div 2 = 1$
    - ID inicial de M: $q_0000100$



  - $2 \div 2 = 0$
    - ID inicial de M: q_000100



  - $2 \div 3 = 0$
    - ID inicial de M: q_001000



Introdução às Máquinas de Turing

A Máquina de Turing

- **A Linguagem de uma Máquina de Turing:**
  - Seja $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ uma TM, sua linguagem é definida como:

$$L(M) = \{ w \mid q_0 w \overset{*}{\underset{M}{\vdash}} \alpha p \beta \}$$

  - Onde $p \in F$, $\alpha \in \Gamma^*$ e $\beta \in \Gamma^*$

Introdução às Máquinas de Turing
A Máquina de Turing

- **As Máquinas de Turing e sua Parada:**
  - Uma outra noção de aceitação usada comumente para TMs é a aceitação por parada;
  - Uma TM para se ela entra em um estado q, varrendo um símbolo de fita X e não existe mais nenhum movimento nessa situação, ou seja, $\delta(q, X)$ é indefinido;
  - Nosso segundo exemplo de TM realiza a computação de uma função aritmética, parando em q_6;
  - Podemos supor que sempre que uma TM aceita ela irá parar. Para isso podemos tornar $\delta(q, X)$ indefinido quando q for um estado de aceitação;
    - Supomos que uma TM sempre para quando se encontra em um estado de aceitação.

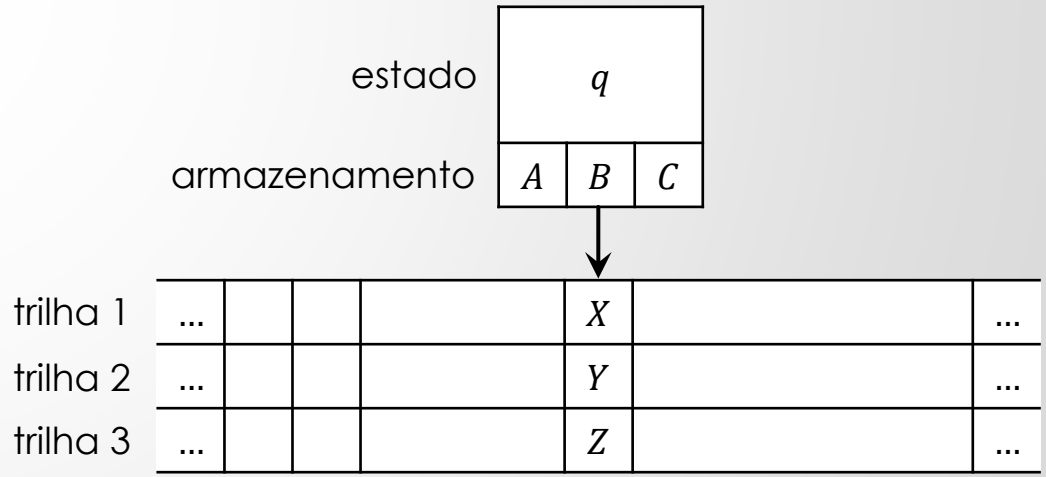
Introdução às Máquinas de Turing
A Máquina de Turing

- **As Máquinas de Turing e sua Parada:**
  - Nem sempre é possível exigir que uma TM pare, mesmo quando não ocorre aceitação;
  - As linguagens que possuem TMs que eventualmente param, independente de aceitação ou não, são chamadas de recursivas;
  - Essas TMs são um bom modelo de um algoritmo e, caso exista um algoritmo para resolver um problema, saberemos que o problema é decidível, sendo assim as TMs que sempre param desempenham um papel importante na teoria da decidibilidade.

Introdução às Máquinas de Turing

A Máquina de Turing

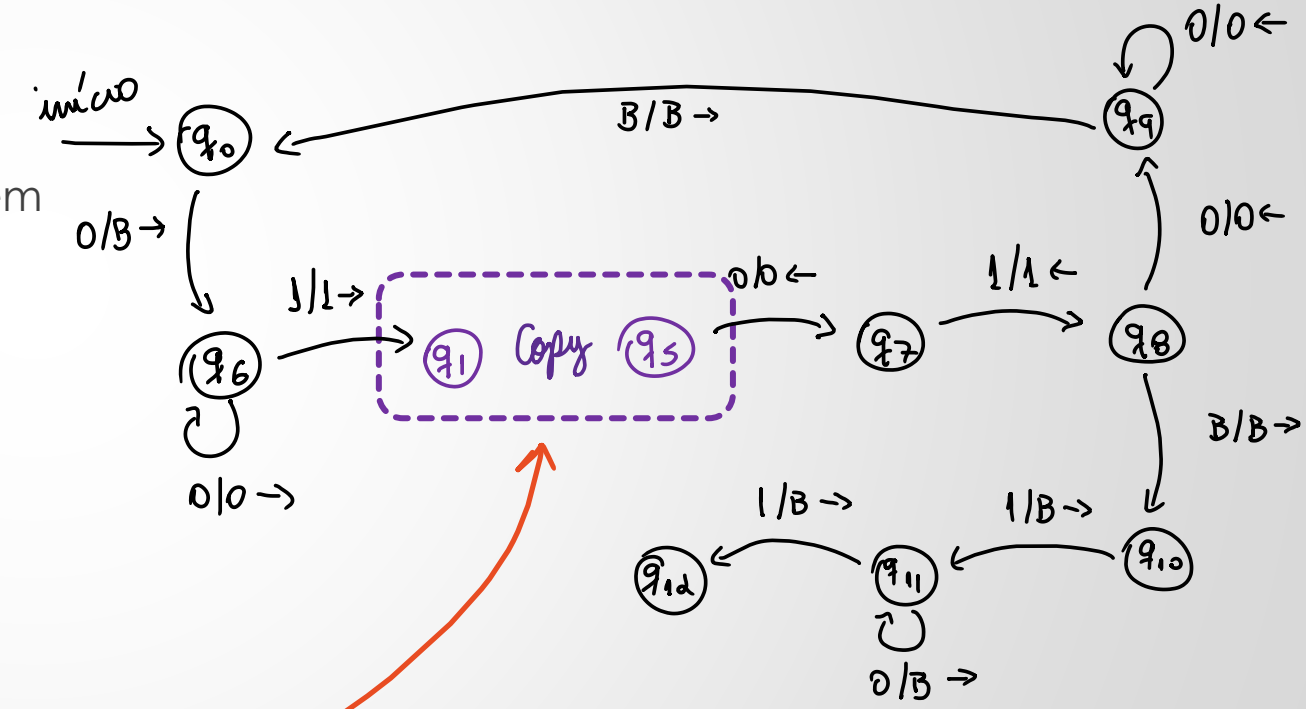
- **Técnicas de Programação de TMs:**
  - Queremos ver como as TMs podem ser usadas para calcular de maneira similar aos computadores convencionais;
  - Nenhum desses artifícios estende o modelo básico da TM, sendo apenas conveniências de notação;
  - **Armazenamento no estado:**
    - Permitimos que o controle finito armazene uma quantidade finita de dados;
  - **Várias trilhas:**
    - O alfabeto da fita consiste em tuplas, com um componente para cada trilha.

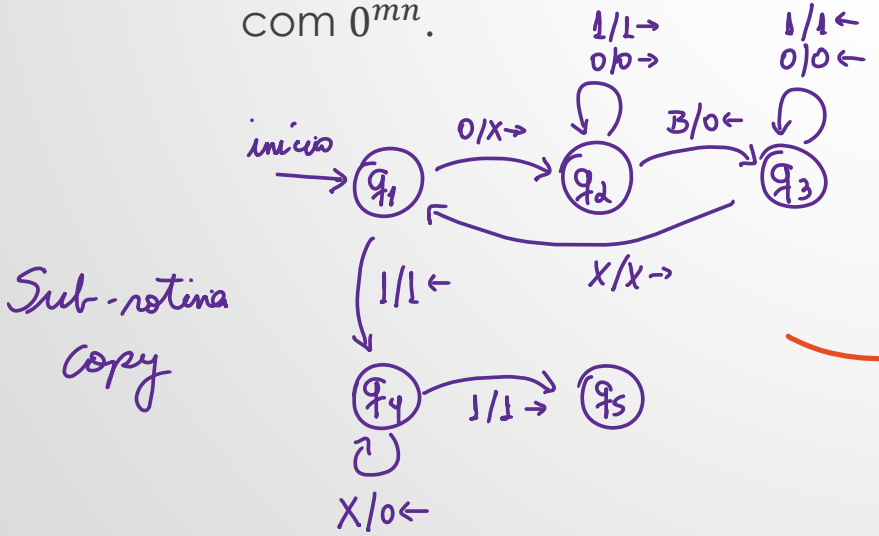


Introdução às Máquinas de Turing

A Máquina de Turing

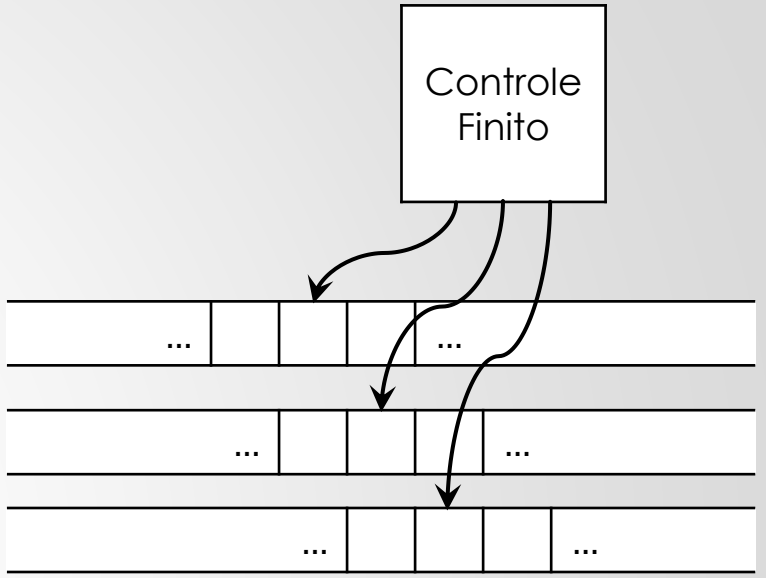
- **Técnicas de Programação de TMs:**
  - **Sub-rotinas:**
    - TMs que usam outras TMs como componente, assim como um programa em Java, C etc. podem ser modularizados;
    - **Exemplo:** TM que implementa a função multiplicação. A TM começa com 0^m10^n1 e termina com 0^{mn}.





Introdução às Máquinas de Turing

A Máquina de Turing



- **Extensões para a TM básica:**
  - Têm o mesmo poder de reconhecimento de linguagens que o modelo básico das TMs;
  - **Várias fitas:**
    - O modelo de várias fitas permitirá que possamos entender de forma mais fácil como TMs desse tipo podem simular computadores reais;

      Inicialmente:

      1. A entrada é colocada na primeira fita;
      2. Todas as outras células de todas as fitas contém brancos;
      3. O controle finito está no estado inicial;
      4. A cabeça da primeira fita está na extremidade esquerda da entrada;
      5. Todas as outras cabeças estão em posições arbitrárias.

      Movimentação:

      1. O controle entra num novo estado;
      2. Em cada fita, um novo símbolo de fita é escrito na célula varrida;
      3. Cada uma das cabeças da fita faz um movimento, que pode ser para esquerda (L), para a direita (R) ou estacionário (S), movendo-se independentemente.

Introdução às Máquinas de Turing

A Máquina de Turing

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

- **Extensões para a TM básica:**
  - NTM: máquina de Turing não-determinística (*Nondeterministic Turing Machine*), uma 7-upla, onde:
    - Q: conjunto finito de estados;
    - Σ: conjunto finito de símbolos de entrada (alfabeto);
    - Γ: conjunto finito de símbolos de fita, tal que $\Sigma \subset \Gamma$;
    - δ: função de transição, na forma:

      $$\delta(q, X) \rightarrow \{(p_1, Y_1, D_1), (p_2, Y_2, D_2), \ldots, (p_k, Y_k, D_k)\},$$

      ou seja,

      $$\delta: Q \times \Gamma \rightarrow P(Q \times \Gamma \times D), \text{ onde:}$$

      - q é um estado;
      - X é um símbolo de fita;
      - $p_1, p_2, \ldots, p_k$ são os próximos estados;
      - $Y_1, Y_2, \ldots, Y_k$ são os símbolos contidos em Γ, gravados nas células que estão sendo varridas, substituindo os símbolos que estavam nessas células;
      - $D_1, D_2, \ldots, D_k$ são uma direção ou sentido, sendo L (esquerda) ou R (direita), informando o sentido em que a cabeça se move.
    - q_0: estado inicial, tal que $q_0 \in Q$;
    - B: o símbolo branco, tal que $B \in \Gamma \wedge B \notin \Sigma$. Ele aparece em todas as células da fita, com exceção das células usadas para conter os símbolos de entrada;
    - F: conjunto de estados finais ou de aceitação, tal que $F \subseteq Q$

Introdução às Máquinas de Turing

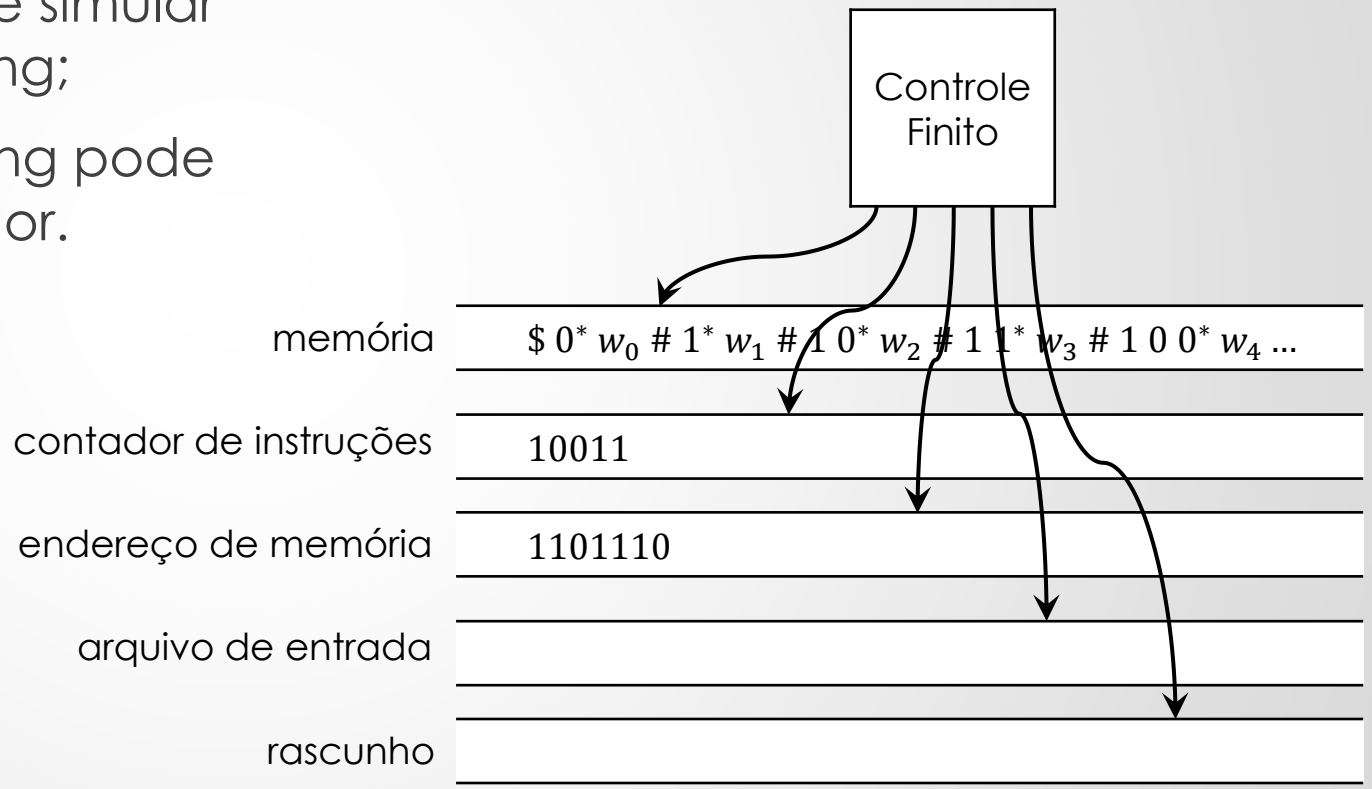
A Máquina de Turing

- **TMs Restritas:**
  - As restrições, assim como as extensões para as TMs, não adicionam qualquer poder de reconhecimento de linguagens;
  - **Fitas semi-infinitas:**
    - A fita é infinita apenas para a direita;
  - **Máquinas de várias pilhas:**
    - PDAs com duas pilhas têm o mesmo poder de reconhecimento que as TMs;
  - **Máquinas de contadores:**
    - Estrutura análoga às máquinas de várias pilhas, mas no lugar que cada pilha há um contador que pode ser adicionado ou subtraído em uma unidade, mas não pode ser negativo, valendo no mínimo, e inicialmente, zero.

Introdução às Máquinas de Turing

A Máquina de Turing

- **Simulação de um Computador por Uma Máquina de Turing:**
  - Um computador pode simular uma máquina de Turing;
  - Uma máquina de Turing pode simular um computador.



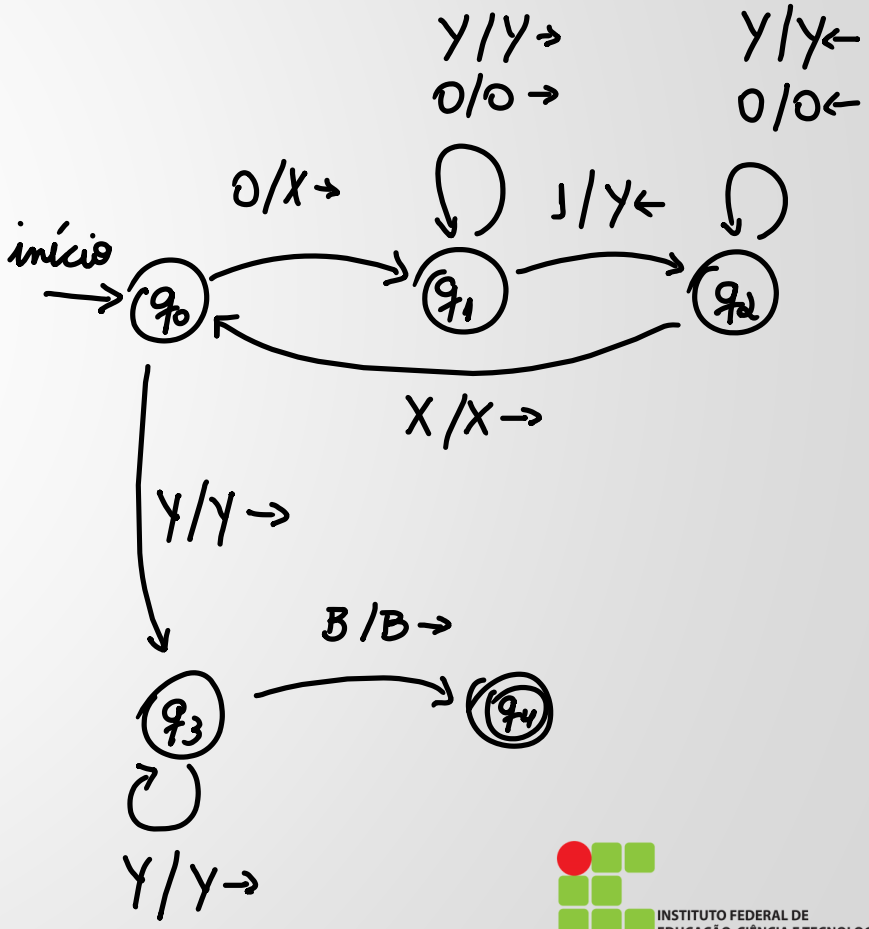
Introdução às Máquinas de Turing
Exercícios Escritos

Exercício e10.1: Mostre as IDs da máquina de Turing apresentada, que aceita $L = \{0^n 1^n \mid n \geq 1\}$, se a fita de entrada contém:

a) 00

b) 000111

c) 00111



Introdução às Máquinas de Turing
Exercícios Escritos

Exercício e10.2: Projete TMs para as seguintes linguagens:

a) O 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) $\{ ww^R \mid w \text{ é qualquer string de } 0's \text{ e } 1's \}$

Introdução às Máquinas de Turing
Exercícios Escritos

Exercício e10.3: Projete uma TM que tome como entrada um número n e adicione 1 a esse número em binário. Para ser preciso, inicialmente a fita contém um \$ seguido por n em binário. A cabeça da fita está inicialmente varrendo o \$ no estado q_0. Sua TM deve parar com $n+1$, em binário, na sua fita, varrendo o símbolo mais à esquerda de $n+1$, no estado q_f. Você pode destruir o \$ ao criar $n+1$, se necessário. Por exemplo, $q_0\$10011 \overset{*}{\vdash} \$q_f10100$ e $q_0\$11111 \overset{*}{\vdash} q_f100000$. Após projetar a TM, mostre a sequência de IDs quando são dadas as entradas 111, 10011 e 11111.

Introdução às Máquinas de Turing

Exercícios Escritos

Exercício e10.4: Uma operação comum em programas de TMs envolve um "deslocamento". No caso ideal, gostaríamos de criar uma célula extra na posição atual da cabeça, na qual poderíamos armazenar algum caractere. Porém, não podemos editar a fita desse modo. Em vez disso, precisamos mover o conteúdo de cada uma das células à direita da posição atual da cabeça, uma célula para a direita, e depois voltar à posição atual da cabeça. Mostre como executar essa operação. **Sugestão:** deixe um símbolo especial para marcar a posição a qual a cabeça deve retornar.

Introdução às Máquinas de Turing

Exercícios Escritos

Exercício e10.5: Projete uma sub-rotina para mover a cabeça de uma TM de sua posição atual para a direita, saltando sobre todos os 0's, até chegar a um 1 ou branco. Se a posição atual não contiver 0, a TM deve parar. Você poderá supor que não existem símbolos de fita além de 0, 1 e B (branco). Assim, use essa sub-rotina para projetar uma TM que aceite todas as strings de 0's e 1's que não têm dois 1's seguidos.

Introdução às Máquinas de Turing

Exercícios Escritos

Exercício e10.6: Projete uma TM que implementa a função adição. A TM começa com 0^m10^n1 e termina com 0^{m+n}. Considere que $m > 0$ e $n > 0$.

Introdução às Máquinas de Turing

Exercícios Escritos

Exercício e10.7: Projete uma TM que implementa a função divisão de inteiros. A TM começa com 0^m10^n1 e termina com $0^{m/n}$. Considere que $m > 0, n > 0$ e $m \geq n$.

Bibliografia

HOPCROFT, J. E.; ULLMAN, J. D.; MOTWANI, R. **Introdução à Teoria de Autômatos, Linguagens e Computação**. 2. ed. Rio de Janeiro: Elsevier, 2002. 560 p.

RAMOS, M. V. M.; JOSÉ NETO, J.; VEGA, I. S. **Linguagens Formais: Teoria, Modelagem e Implementação**. Porto Alegre: Bookman, 2009. 656 p.

SIPSER, M. **Introdução à Teoria da Computação**. 2. ed. São Paulo: Cengage Learning, 2017. 459 p.