Variantes de Máquinas de Turing

Máquinas de Turing Input-Output

TM's ***Input/output*** (ou ***IO*** ou ***transdutoras***) diferem de TM's reconhecedoras por terem um estado de parada neutro q_{halt}, ao invés de estados de aceitação e rejeição. Esse tipo de TM é então vista como uma função (string -> string), que mapeia o conteúdo inicial da fita *u* ao conteúdo (porção não vazia) da fita no instante em que atinge o estado q_{halt}. Se *v* é o conteúdo da fita quando a máquina pára, escrevemos $f_M(u) = v$

Se *M* falha durante a computação, ou entra em um loop infinito, *M* é dita ***indefinida*** sobre *u*.

Máquinas de Turing Input-Output

Quando f_M falha ou entra em um loop infinito para algum string de entrada, f_M é dita uma ***função parcial.***

Se *M* sempre pára (halt) para qualquer possível string de entrada, a função f_M é ***total*** (i.e. sempre definida).

TM Notação

Sipser adota três maneiras de descrever algoritmos de TM's.

1) Pseudocódigo de alto nível, que explica como o algoritmo funciona, sem entrar em detalhes técnicos da notação formal de TM's
2) Nível de implementação – descreve como a TM opera sobre sua fita, sem mencionar estados explicitamente.
3) Descrição de baixo nível:
   - Conjunto completo de instruções em estilo "goto"
   - Diagrama de Estados
   - Descrição Formal (na forma de 7-tuple)

TM: Alto Nível
Exemplo

Vamos descrever, por exemplo, uma Máquina de Turing *M* que multiplica um número por 2, em notação unária:

M = "Sobre a entrada $w = 1^n$

Para cada caractere *c* em *w*

copie *c* no próximo "espaço branco'' disponível

TM: Nível de Implementação Exemplo

A idéia é mostrar como funciona o processo de copiar cada caractere no final da fita, mencionado na descrição de alto nível. Devemos manter informação sobre que caracteres já foram copiados, distinguindo esses caracteres. Uma maneira de fazer isso é usar um caractre distinto, digamos X.

EX: Vejamos como 11111 é transformado.

TM: Nível de Implementação Exemplo

Então, passo a passo, o conteúdo da entrada é transformado como a seguir:

- 11111
- X1111X
- XX111XX
- XXX11XXX
- XXXX1XXXX
- XXXXXXXXXX
- 1111111111

TM: Nível de Implementação Exemplo

O ***nível de implementação*** descreve como o algoritmo opera de fato na máquina de Turing, facilitando a obtenção do diagrama de estados

Alguns métodos úteis:

- fast forward
  - move p/ a dir. enquanto uma condição é satisfeita
- rewind
  - move p/ esq. enquanto uma condição é satisfeita
- Pode requerer funcionalidade extra:
  - Adicione $ se for necessário detectar fim de fita

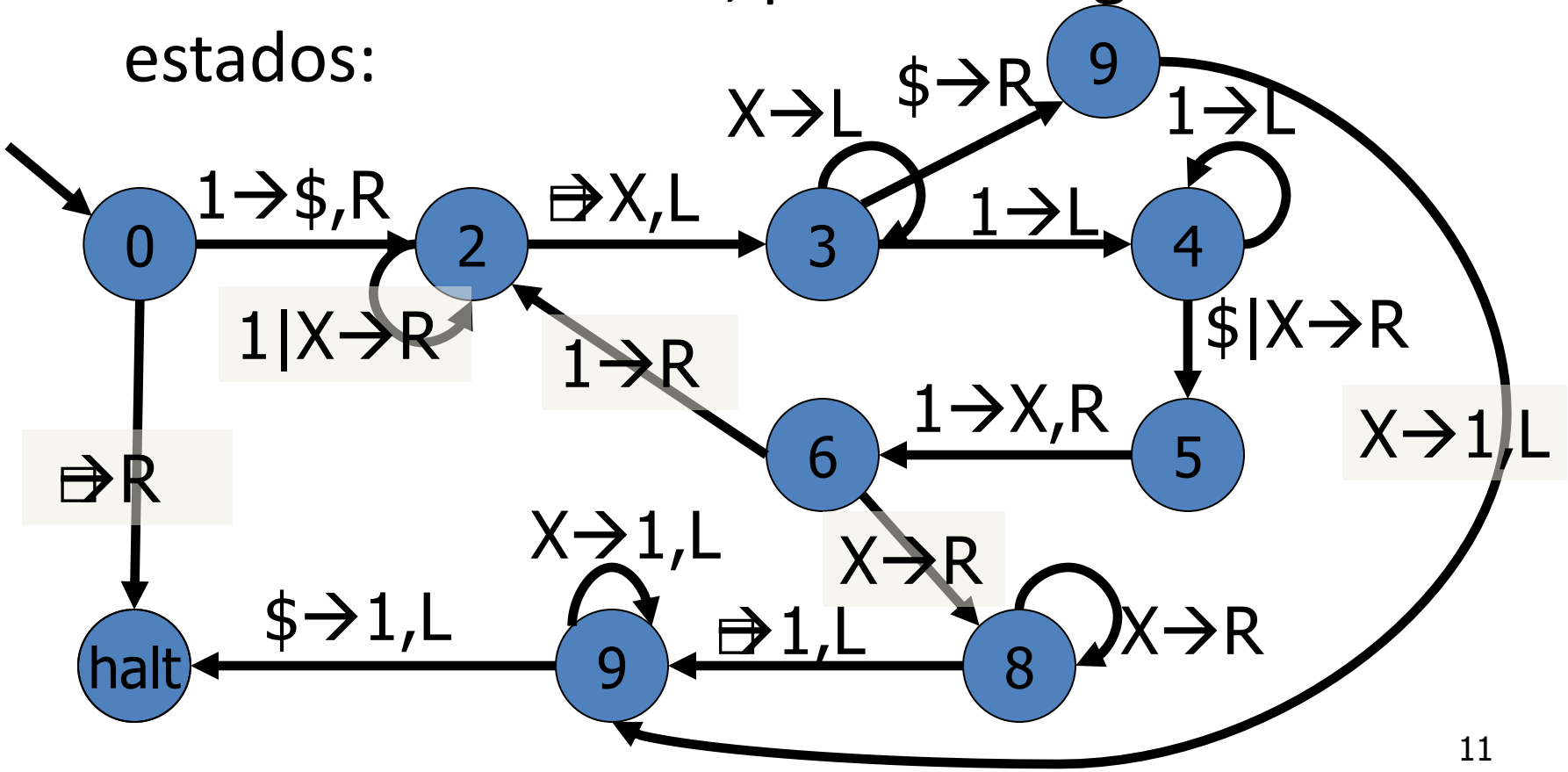
TM: Nível de Implementação Exemplo

M = “Sobre entrada *w* = 1^n

1. HALT se entrada vazia
2. Escreva $ na posição mais à esquerda
3. Ande p/ dir. e escreva X no primeiro branco
4. Ande p/ esq. enquanto encontrar X ou 1
5. Mova para direita
6. Se ler X, mova p/ direita e vá para 9. Senão, substitua 1 por X, mova p/ direita.
7. Se lê X [[*w* original terminado]] vá para 8 Senão, vá para 3
8. Ande p/ dir. até encontrar branco e o substitua p/ X
9. Ande p/ esq. substituindo todo não branco por 1
10. HALT

TM: Baixo Nível
Exemplo

No ***baixo nível*** a Máquina de Turing é descrita em detalhe, por um diagrama de estados:



TM's Não Deterministas

Uma Máquina de Turing não Determinista *N* permite mais de uma possível ação para um dado par (estado, símbolo na fita).

Um string *w* é ***aceito*** por *N* se algum *ramo da computação* de *N* sobre esse string entra eventualmente no estado q_{acc}.

Se, por outro lado, em qq ramo da computação, *N* eventualmente entra no estado q_{rej} **ou** falha **ou** entra em um loop infinito, *w* não é aceito por aquele ramo.

Simbolicamente:

$$L(N) = \{ x \in \Sigma^* \mid \exists \text{ config.de aceitação } y,\ q_0 x \Rightarrow^* y \}$$

TM's Não Deterministas Reconhecedores vs. Decisores

N é dita um ***reconhecedor não determinista*** e diz-se que *reconhece L*(*N*); além disso, se para qualquer string de entrada e todos os ramos de computação, *N* sempre pára, então *N* é dita um ***decisor não determinista*** e diz-se que *decide L*(*N*).

TM Não Determinista
Exemplo

Considere o seguinte método não determinista:

```
void nonDeterministicCrossOut(char c)
        while()
            if (read blank) go left
            else
                        if (read c)
                                    cross out, go right, return
                       OR go right
                       OR go left
```

Non-Deterministic TM Example

Usando nonDeterministicCrossOut() construimos o seguinte programa:

1. while(some character not crossed out)
   nonDeterministicCrossOut(‘0’)
   nonDeterministicCrossOut(‘1’)
   nonDeterministicCrossOut(‘2’)
2. ACEITA

Q: Qual é a linguagem que esse programa não determinista reconhece ?

Non-Deterministic TM Example

R: $\{x \in \{0,1,2\}^* \mid x$ tem igual no. de 0's e 1's e 2's $\}$

Q: Suponha que q seja o estado em que permanece a TM enquanto executa nonDeterministicCrossOut('1') e q' é o estado em que executa nonDeterministicCrossOut('2').

Suponha que a configuração corrente é

$$u = 0XX1Xq12X2$$

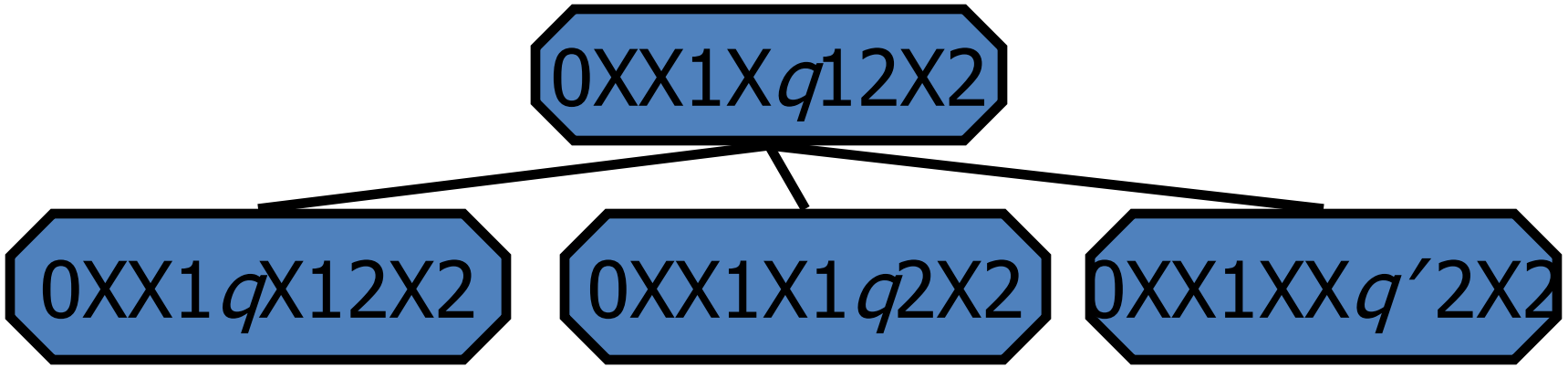
Para qual v temos que $u \Rightarrow v$?

Non-Deterministic TM Example

R: 0XX1Xq12X2 $\Rightarrow$

0XX1qX12X2 | 0XX1X1q2X2 | 0XX1XXq' 2X2

Isso define 3 ramos da árvore de computação



Q: Essa TM não determinista TM é um *decisor*?

Non-Deterministic TM Example

R: Não. Essa TM é um reconhecedor, e não um decisor. nonDeterministicCrossOut() pode entrar em um ramo de computação infinito, já que pode alternar move esq-dir ad infinitum, sem nunca marcar um símbolo. I.e., a árvore de computação é infinita!

Nota: Se você desenhar o diagrama de estados, verá a NTM é mais compacta que a versão TM, portanto há vantagens no não determinismo! Mais tarde, você verá exemplos de programas não deterministas "eficientes" para problemas práticos importantes, sem correspondentes deterministas eficientes: O problema **P** vs. **NP.**

NTM's
Lema de Konig

Para o Problema 3.3 (Sipser) o seguinte fato é importante:

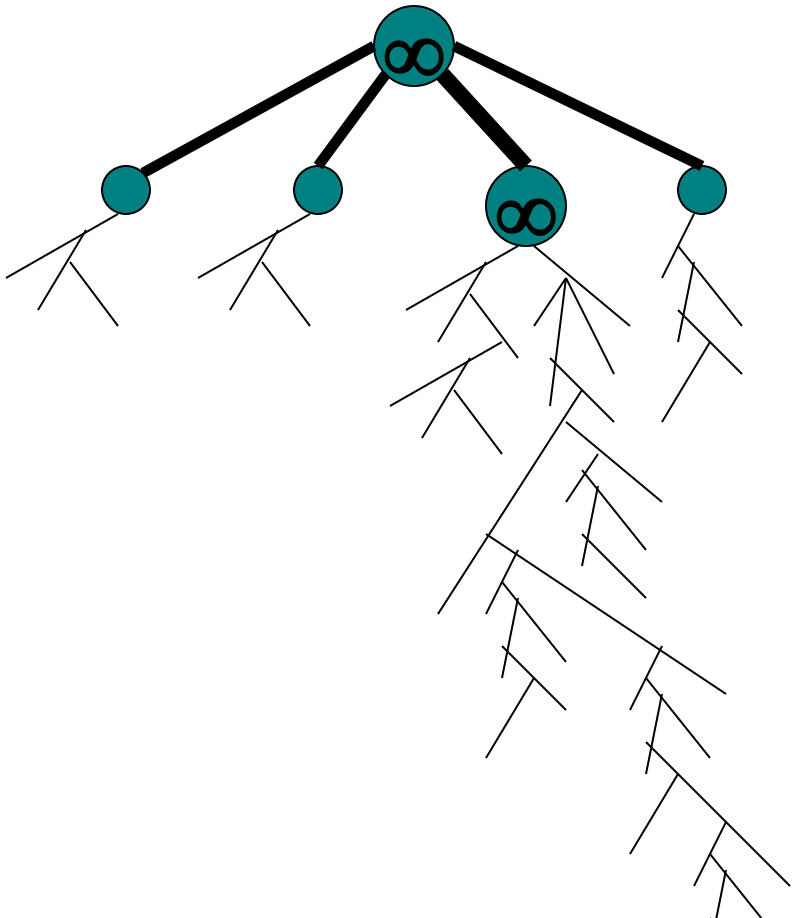
Se uma NTM é um decisor, então dada uma entrada qualquer, existe um número *h* tal que todo ramo de computação envolve no máximo *h* passos básicos, ou seja, a árvore de computação tem altura *h*. Isso segue de:

Lema de Konig: Uma árvore infinita, com número finito de ramos em cada nodo, deve conter algum caminho infinitamente longo.

De fato, usamos o contrapositivo: uma árvore sem caminho infinito e com no. finito de ramos em cada nodo deve ser finita.

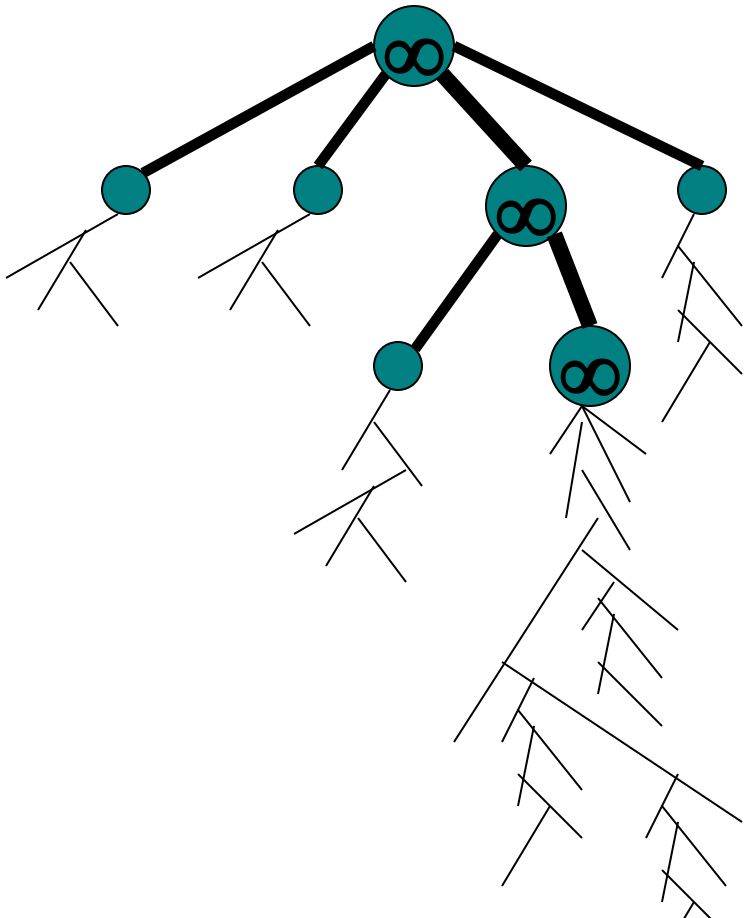
Lema de Konig
Idéia da Prova

A idéia é “advinhar” onde é a parte infinita da árvore e ir nessa direção:



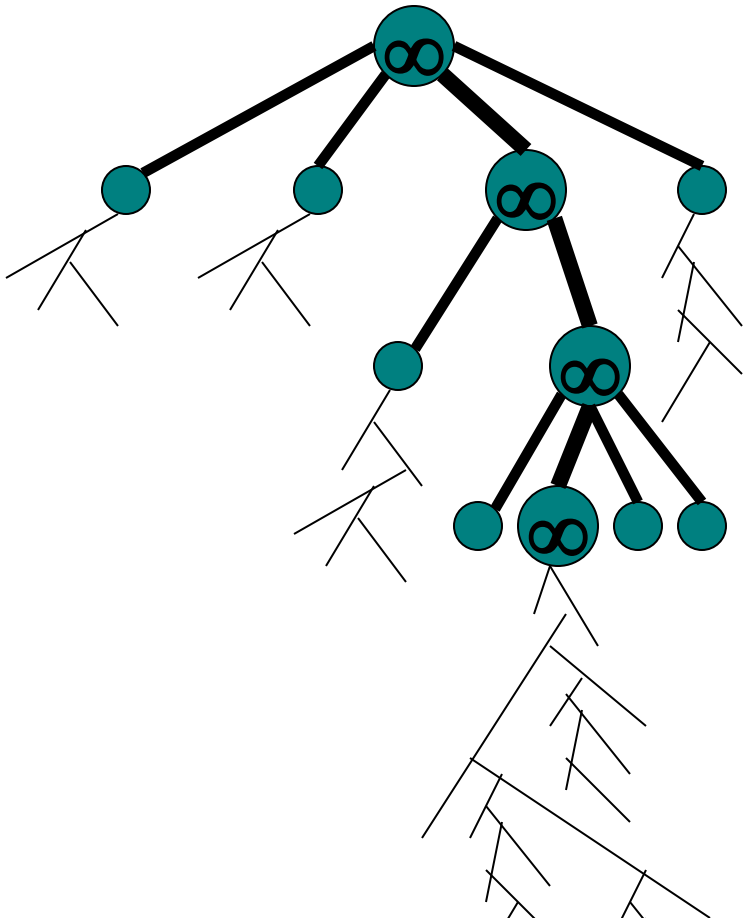
Lema de Konig
Idéia da Prova

A idéia é “advinhar” onde é a parte infinita da árvore e ir nessa direção:



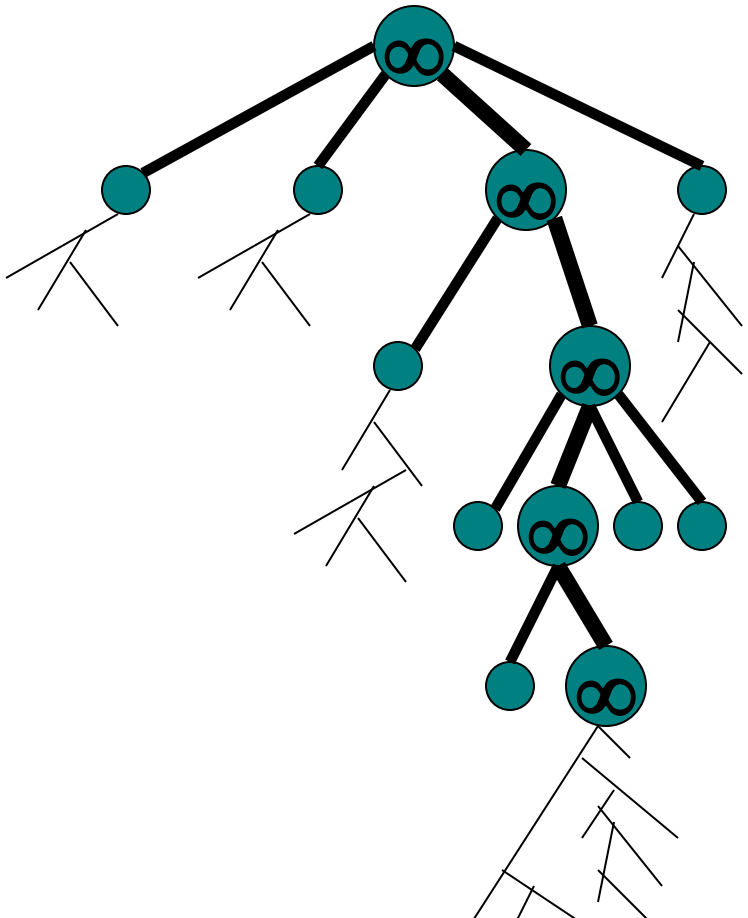
Lema de Konig
Idéia da Prova

A idéia é “advinhar” onde é a parte infinita da árvore e ir nessa direção:



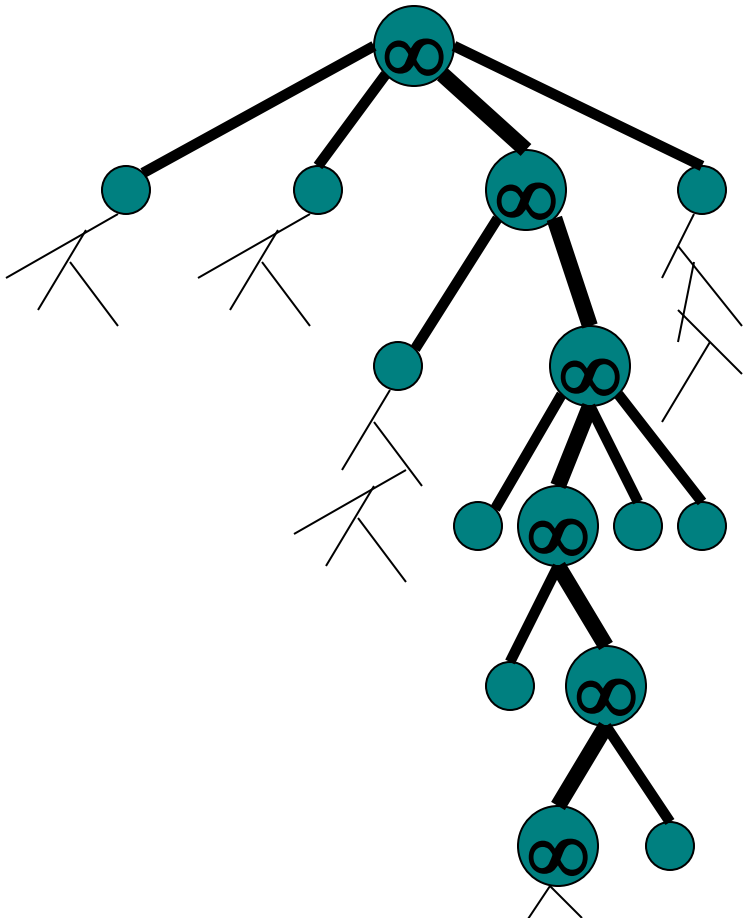
Lema de Konig
Idéia da Prova

A idéia é “advinhar” onde é a parte infinita da árvore e ir nessa direção:



Lema de Konig
Idéia da Prova

A idéia é “advinhar” onde é a parte infinita da árvore e ir nessa direção:



Lema de Konig
Prova

Prova. Dada uma árvore infinita com número finito de ramos em cada nodo, construímos um caminho infinito indutivamente:

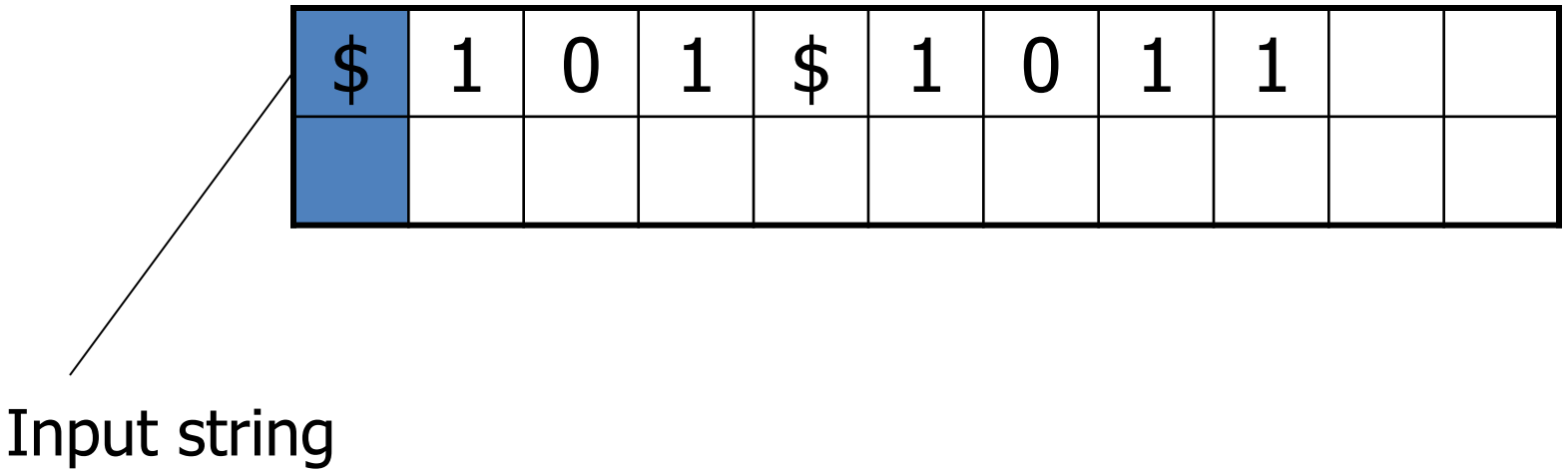
O vértice inicial v_0 é a raiz.

Arco $v_n \rightarrow v_{n+1}$: Suponha $v_0 \rightarrow v_1 \ldots v_{n-1} \rightarrow v_n$ tenha sido construído e que a subárvore a partir de v_n é infinita. Então um dos filhos (em no. finito) de v_n, seja v_{n+1}, deve ter uma subárvore infinita; então adicione o arco $v_n \rightarrow v_{n+1}$. □

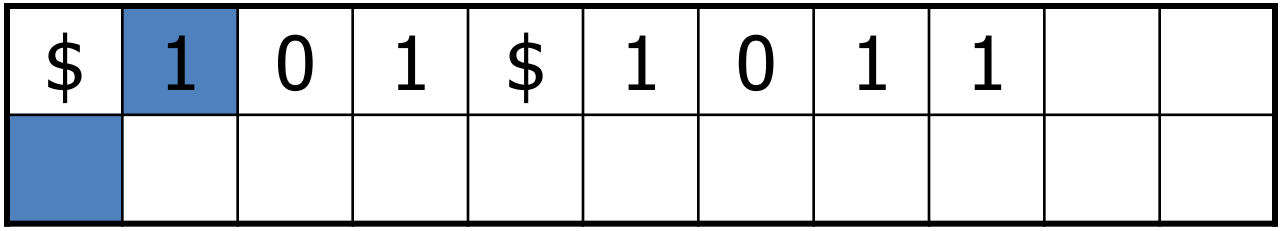
TM com Múltiplas Fitas

Frequentemente é útil considerar que temos várias fitas ao realizar uma computação. Por exemplo, considere uma TM com duas fitas I/O para adicionar números (vamos apenas mostrar como ela atua sobre uma entrada típica)

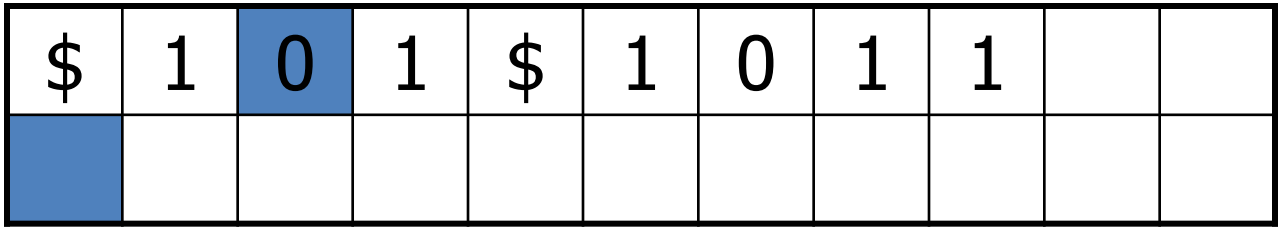
TM com Múltiplas Fitas
Exemplo: Adição



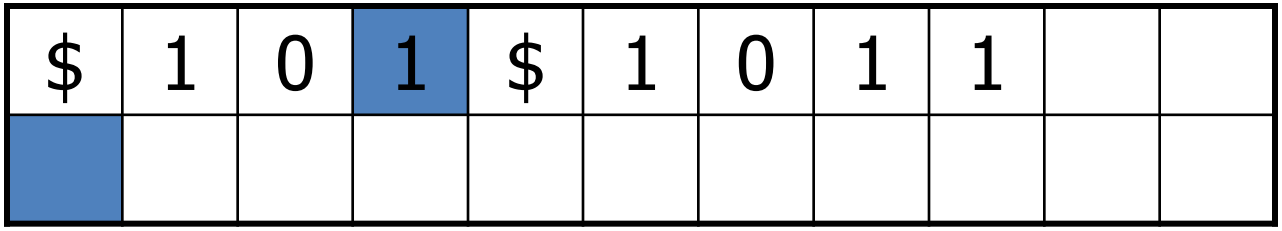
TM com Múltiplas Fitas
Exemplo: Adição



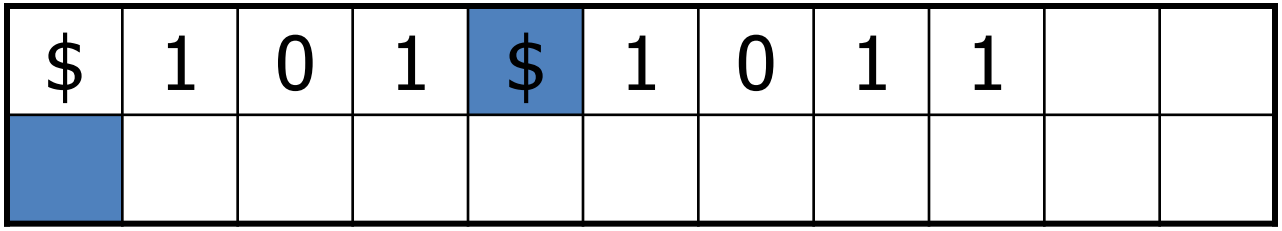
TM com Múltiplas Fitas
Exemplo: Adição



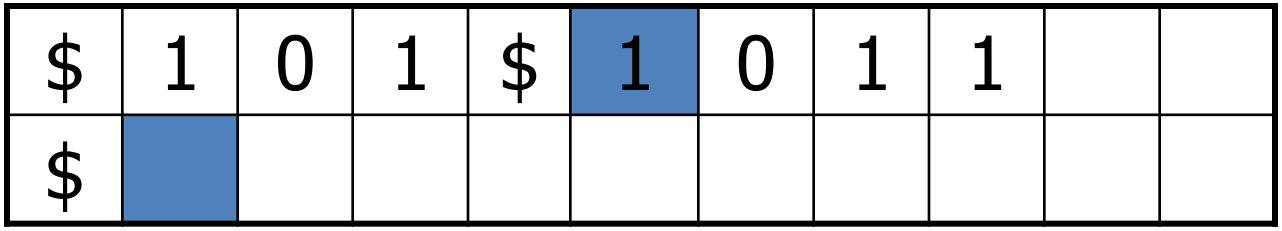
TM com Múltiplas Fitas
Exemplo: Adição



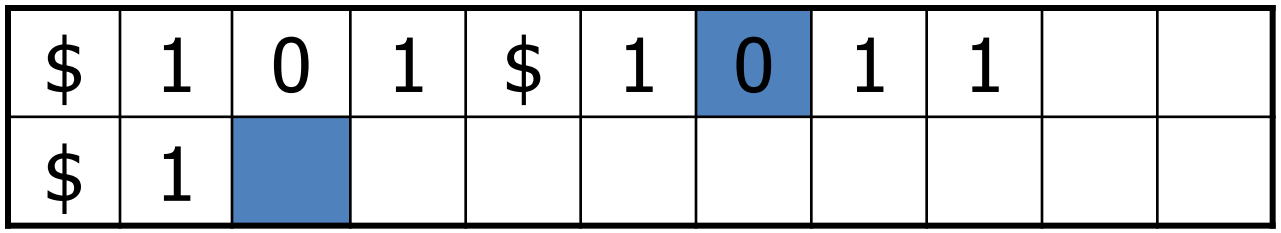
TM com Múltiplas Fitas
Exemplo: Adição



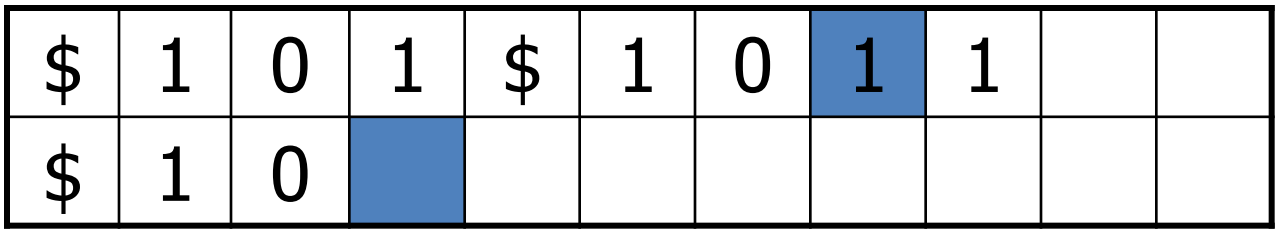
TM com Múltiplas Fitas
Exemplo: Adição



TM com Múltiplas Fitas
Exemplo: Adição



TM com Múltiplas Fitas
Exemplo: Adição



TM com Múltiplas Fitas
Exemplo: Adição

$	1	0	1	$	1	0	1	1		
$	1	0	1							

TM com Múltiplas Fitas
Exemplo: Adição

$	1	0	1	$	1	0	1	1		
$	1	0	1	1						

TM com Múltiplas Fitas
Exemplo: Adição

$	1	0	1	$	1	0	1	1		
$	1	0	1	1						

TM com Múltiplas Fitas
Exemplo: Adição

$	1	0	1	$	1	0	1			
$	1	0	1	1						

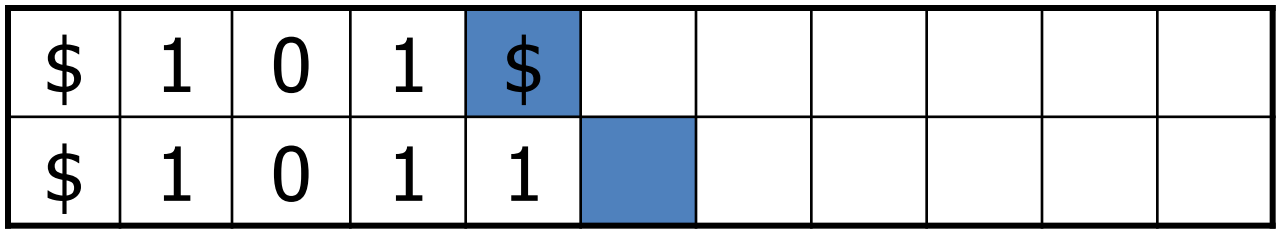
TM com Múltiplas Fitas
Exemplo: Adição

$	1	0	1	$	1	0				
$	1	0	1	1						

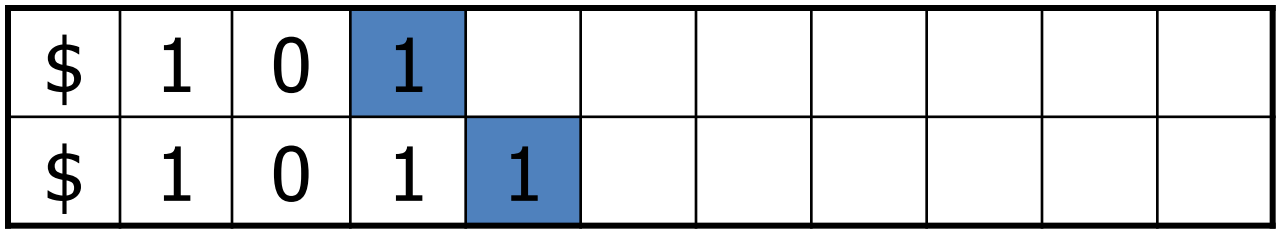
TM com Múltiplas Fitas
Exemplo: Adição

$	1	0	1	$	1					
$	1	0	1	1						

TM com Múltiplas Fitas
Exemplo: Adição



TM com Múltiplas Fitas
Exemplo: Adição



TM com Múltiplas Fitas
Exemplo: Adição

$	1	0	0							
$	1	0	1	0						

TM com Múltiplas Fitas
Exemplo: Adição

$	1	0	0							
$	1	0	0	0						

TM com Múltiplas Fitas
Exemplo: Adição

$	0	0	0							
$	1	0	0	0						

TM com Múltiplas Fitas
Exemplo: Adição

0	0	0	0							
$	0	0	0	0						

TM com Múltiplas Fitas
Exemplo: Adição

1	0	0	0							
1	0	0	0	0						

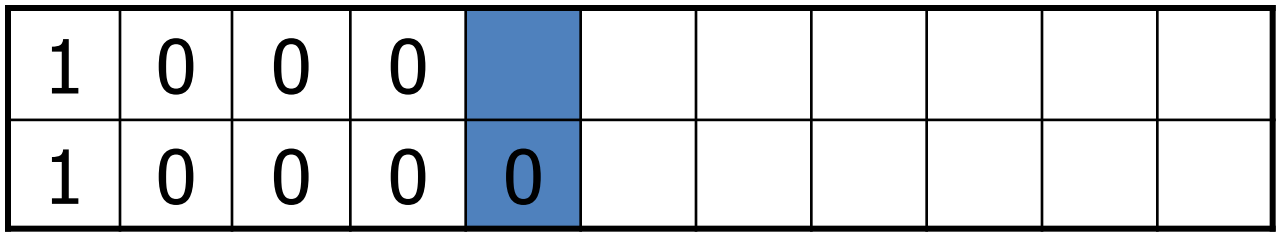
TM com Múltiplas Fitas
Exemplo: Adição

1	0	0	0							
1	0	0	0	0						

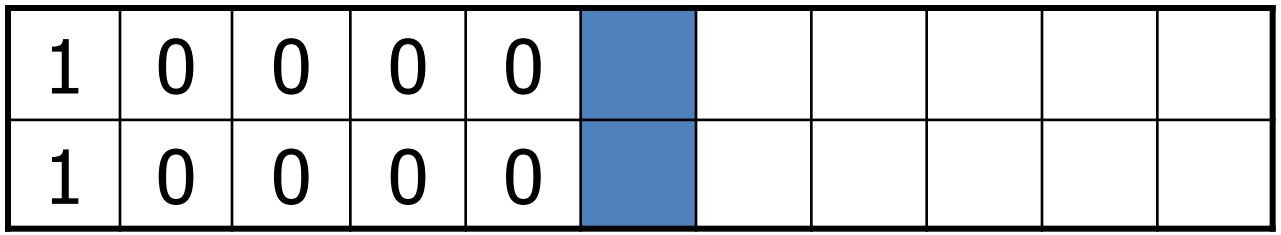
TM com Múltiplas Fitas
Exemplo: Adição

1	0	0	0							
1	0	0	0	0						

TM com Múltiplas Fitas
Exemplo: Adição



TM com Múltiplas Fitas
Exemplo: Adição



HALT!

Output string

TM com Múltiplas Fitas
Notação Formal

NOTA: A TM multifita de Sipser não pode pausar em uma das fitas como faz a do exemplo anterior. Isso não é problema pois uma pausa pode ser simulada por um "move L-R".

Formalmente, a função de transição δ de uma máquina *k*-fita é:

$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{\mathrm{L}, \mathrm{R}\}^k$$

TM com Múltiplas Fitas Convenção

- Entrada e saída sempre na 1$^{a.}$ fita
- Se for máquina I/O, a saída é também na primeira fita
- Essas máquinas podem ser consideradas como geradores de “vetores de string”. E.x., poderíamos supor que uma máquina 4-fita produz como saída um valor em $(\Sigma^*)^4$

Exercício

- Descreva como construir máquinas de Turing Multifita para reconhecer:
  - $\{w\ w^R\ w \mid w \in \{0,1\}^*\}$
  - $\{w\ w \mid w \in \{0,1\}^*\}$
- Descreva como construir máquinas de Turing não Determinísticas para reconhecer:
  - $\{w\ w \mid w \in \{0,1\}^*\}$
  - $\{x\ x^R\ y \mid x, y \in \{0,1\}^*$ e $|x| > |y|\}$