Modelos Universais de Computação

Equivalência entre Variantes de TM

- TM's definem naturalmente uma classe.
- Toda variante razoável de TM define a mesma classe de linguagens. (reforça a Tese Church-Turing)

Recursivo e Recursivamente Enumerável

DEF: Uma linguagem é ***recursivamente enumerável*** se ela pode ser *reconhecida* por uma Máquina de Turing. Uma linguagem é ***recursiva*** se ela pode ser *decidida* por uma Máquina de Turing.

A terminologia "recursivamente enumerável" vem de outro modelo de TM: uma TM que nunca pára (quando descreve uma linguagem infinita) e escreve cada um dos strings da linguagem em uma fita auxiliar (impressora).

Enumerador Recursivo
Exemplo

Por exemplo, a saída de um enumerador para a linguagem **pal** seria, em um dado instante, como a seguir:

1. (*string vazio*)
2. a
3. b
4. aa
5. bb
6. aba
7. bab
8. aaa
9. bbb

Enumerador Recursivo
Exemplo

Enumeradores Recursivos são equivalentes a TM reconhecedoras, em razão do seguinte argumento:

1) Enumerador E → TM M:
    1) Sobre a entrada w:
        1) Rode E. Toda vez que E dá como saída uma cadeia, compare-a com w.
        2) Se w em algum momento aparece na saída de E, aceite.
2) TM M → Enumerador E:
    1) Ignore a entrada.
    2) Repita o seguinte para i=1, 2, 3, …
        1) Rode M por i passos sobre cada cadeia s_1, s_2, s_3, …
        2) Se quaisquer computações aceitam, imprima a s_j correspondente.

Equivalência de *k*-fitas vs. 1-fita

O argumento de que TM's com *k*-fitas são universais é bastante convincente. Afinal, todos os computadores atuais têm a memória dividida em diversos locais: RAM, Hard-Disc, Floppy, CD-ROM. É plausível que cada unidade de memória possa ser simulada por uma fita separada. O teorema a seguir implica em mostrar que o computador poderia manter a mesma funcionalidade (embora com alguma perda de eficiência) se toda a memória fosse juntada em uma única fita:

Equivalência de *k*-fitas vs. 1-fita

THM: Toda TM de *k*-fitas pode ser simulada por uma TM de 1-fita, e vice versa.

Prova. É claro que toda TM de 1-fita pode ser simulada por uma TM de *k*-fitas. Basta ignorar as fitas de no. 2 a *k*.

Por outro lado, para mostrar que qualquer TM de *k*-fitas pode ser simulada por uma TM de 1-fita TM, vamos provar 2 resultados:

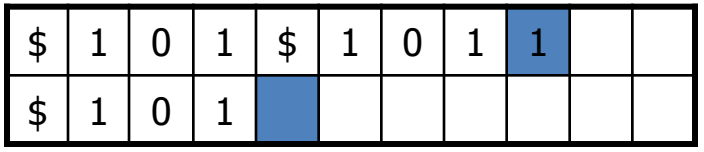
A) Qualquer TM de *k*-fitas pode ser simulada por uma TM de 2*k*-***trilhas***.

B) Qualquer TM de *k*-trilhas pode ser simulada por uma TM de 1-fita.

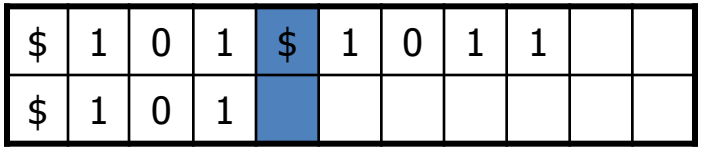
TM's de *k*-trilhas

Uma máquina de *k*-trilhas tem uma fita cujas células são divididas em *k* sub-células. Diferentemente de uma TM de *k*-fitas, ela possui apenas uma cabeça de leitura[1], que lê simultaneamente as *k* trilhas, baseando sua ação nesses valores.

EX. máquina 2-fitas:



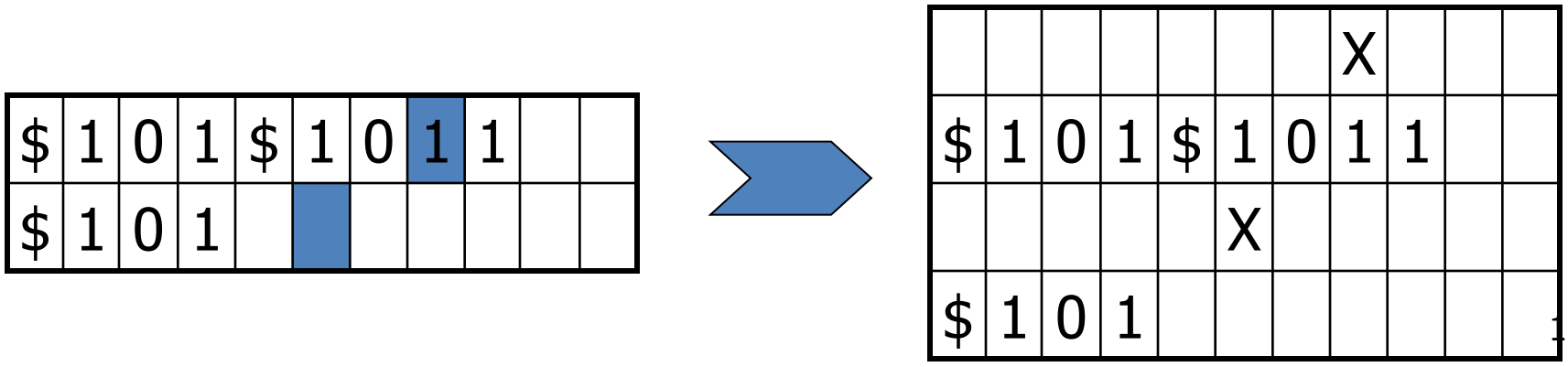
vs. máquina 2-trilhas:



2*k*-trilhas Simulando *k*-fitas

Podemos simulas qualquer TM de *k*-fitas por uma TM de *2k*-trilhas do seguinte modo:

- Cada fita da TM de *k*-fitas é representada por 2 trilhas:
  - A primeira contém o próprio conteúdo da fita.
  - A segunda é uma trilha em que todas as posições são "branco", exceto por um único X que indica qual é a célula corrente nessa fita.



2*k*-trilhas Simulando *k*-fitas

Para cada “move” da TM de *k*-fitas, a TM de 2*k*-trilhas anda p/ direita e depois p/ esquerda.

- Enquanto anda p/ direita, a TM de 2*k*-trilhas obtém a posição da cabeça de leitura de cada uma das *k*-fitas, assim como o símbolo na posição correspondente em cada fita, mantendo essa informação por meio de seus estados. Retorna depois de ler todos os X’s.
- Enquanto anda p/ esquerda, para cada X encontrado a máquina modifica o conteúdo da trilha correspondente, usando o que memorizou sobre a configuração das *k*-fitas.

2*k*-trilhas Simulando *k*-fitas

simulação por
TM de 2k-trilhas

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

TM *k*-trilha
transição

$	1	0	1	$	1	0	3	1		
$	1	0	1		2					

							X			
$	1	0	1	$	1	0	1	1		
					X					
$	1	0	1							

2*k*-trilhas Simulando *k*-fitas

simulação por
TM de 2k-trilhas

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

TM *k*-trilha
transição

$	1	0	1	$	1	0	3	1		
$	1	0	1		2					

							X			
$	1	0	1	$	1	0	1	1		
					X					
$	1	0	1							

2*k*-trilhas Simulando *k*-fitas

simulação por
TM de 2k-trilhas

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

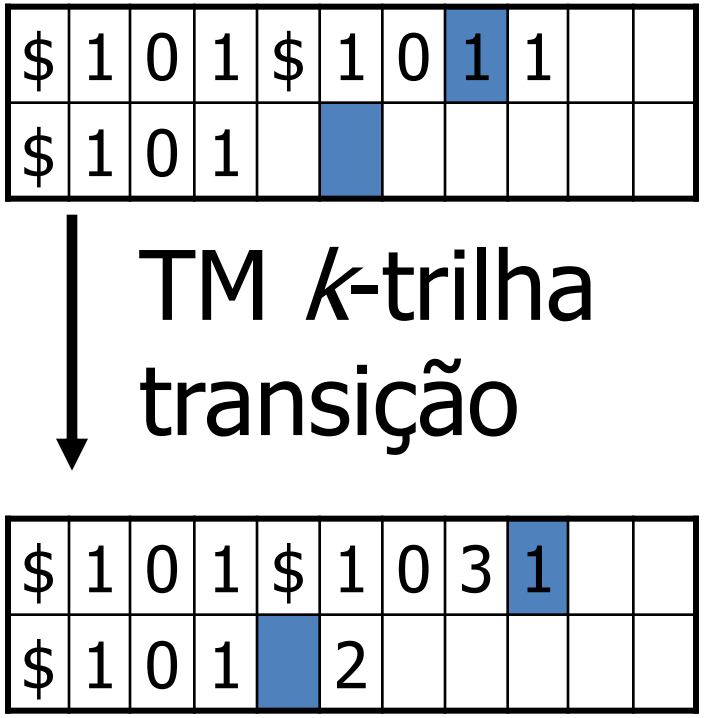
TM *k*-trilha
transição

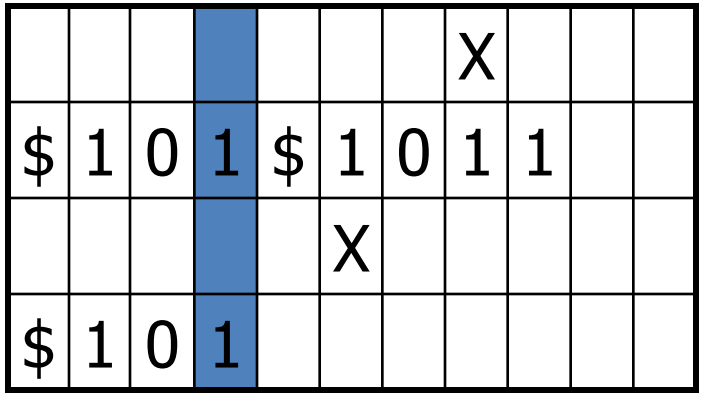
$	1	0	1	$	1	0	3	1		
$	1	0	1		2					

							X			
$	1	0	1	$	1	0	1	1		
					X					
$	1	0	1							

2*k*-trilhas Simulando *k*-fitas

simulação por
TM de 2k-trilhas





2*k*-trilhas Simulando *k*-fitas

simulação por
TM de 2k-trilhas

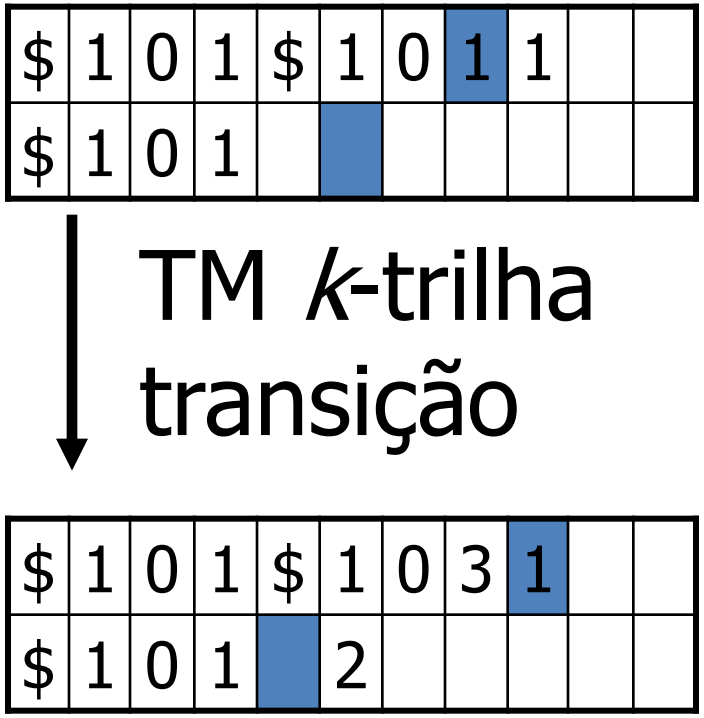
$	1	0	1	$	1	0	1	1		
$	1	0	1							

TM *k*-trilha
transição

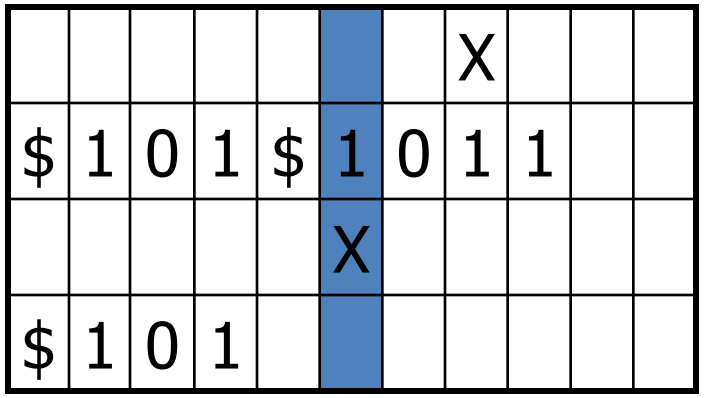
$	1	0	1	$	1	0	3	1		
$	1	0	1		2					

							X			
$	1	0	1	$	1	0	1	1		
					X					
$	1	0	1							

2*k*-trilhas Simulando *k*-fitas



simulação por
TM de 2k-trilhas



lê 2ª. fita

2*k*-trilhas Simulando *k*-fitas

simulação por
TM de 2k-trilhas

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

TM *k*-trilha
transição

$	1	0	1	$	1	0	3	1		
$	1	0	1		2					

							X			
$	1	0	1	$	1	0	1	1		
					X					
$	1	0	1							

2*k*-trilhas Simulando *k*-fitas

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

TM *k*-trilha
transição

$	1	0	1	$	1	0	3	1		
$	1	0	1		2					

simulação por
TM de 2k-trilhas

							X			
$	1	0	1	$	1	0	1	1		
					X					
$	1	0	1							

lê 1ª. Fita
passou todos X's

2*k*-trilhas Simulando *k*-fitas

simulação por
TM de 2k-trilhas

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

TM *k*-trilha
transição

$	1	0	1	$	1	0	3	1		
$	1	0	1		2					

$	1	0	1	$	1	0	3	1		
					X					
$	1	0	1							

2ª. fase, na 1ª.
fita: 1→3,R

2*k*-trilhas Simulando *k*-fitas

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

TM *k*-trilha
transição

$	1	0	1	$	1	0	3	1		
$	1	0	1		2					

simulação por
TM de 2k-trilhas

								X		
$	1	0	1	$	1	0	3	1		
					X					
$	1	0	1							

2ª. fase, na 1ª.
fita: 1→3,R

2*k*-trilhas Simulando *k*-fitas

simulação por
TM de 2k-trilhas

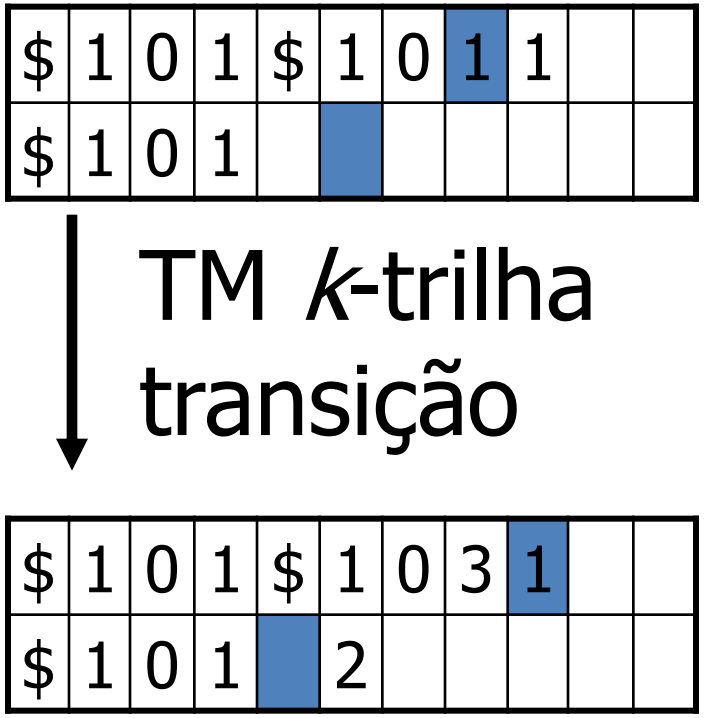
$	1	0	1	$	1	0	1	1		
$	1	0	1							

TM *k*-trilha
transição

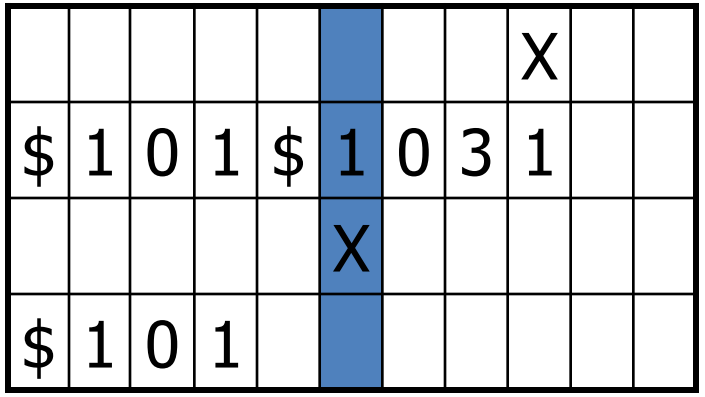
$	1	0	1	$	1	0	3	1		
$	1	0	1		2					

								X		
$	1	0	1	$	1	0	3	1		
					X					
$	1	0	1							

2*k*-trilhas Simulando *k*-fitas

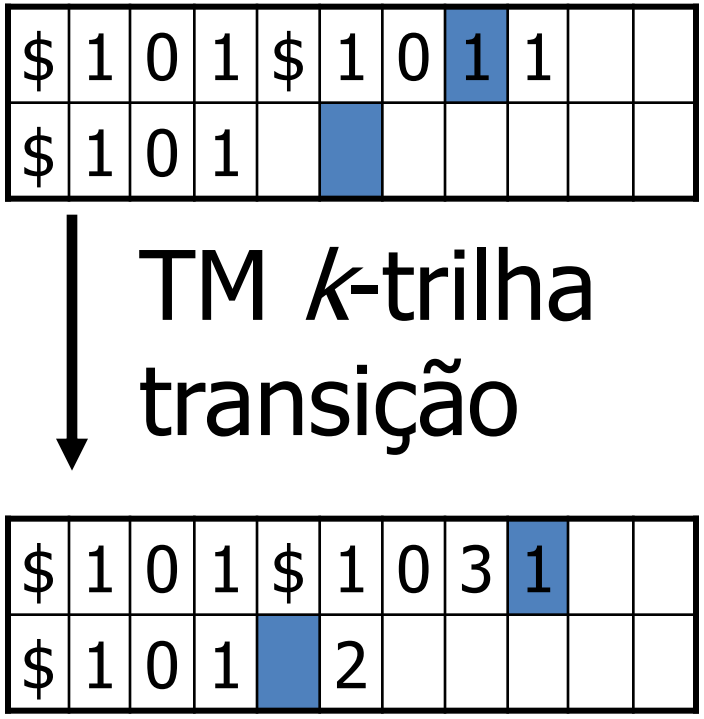


simulação por
TM de 2k-trilhas

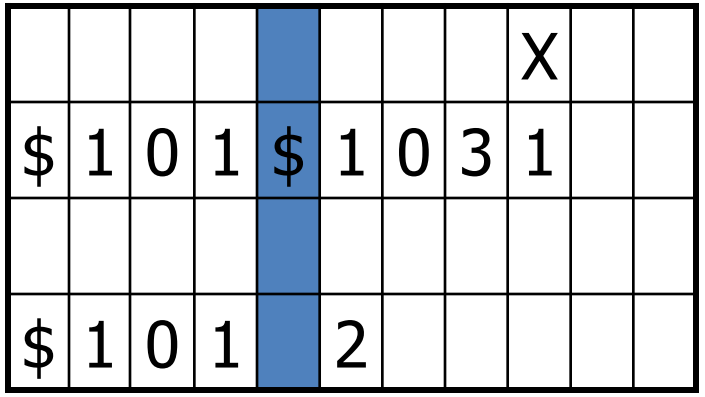


na 2nd fita: ⊟⊠2,L

2*k*-trilhas Simulando *k*-fitas

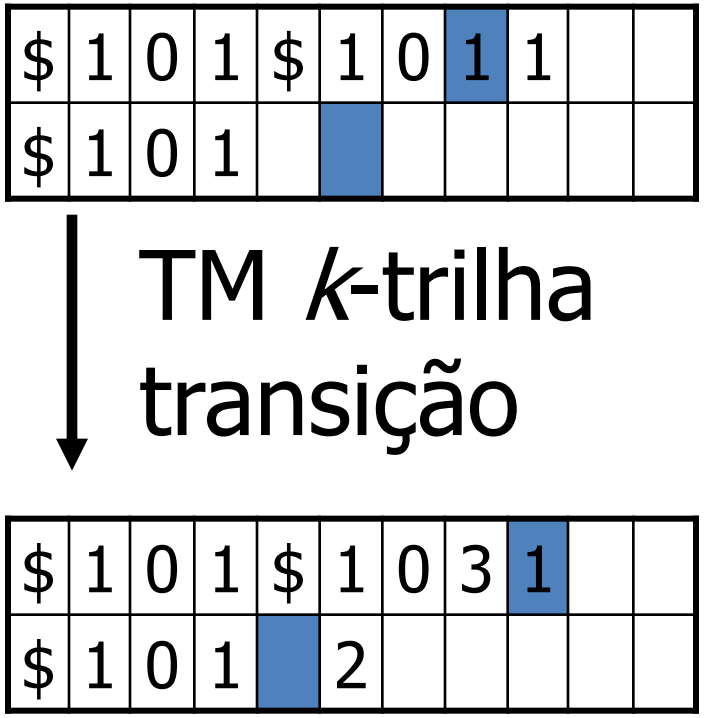


simulação por
TM de 2k-trilhas

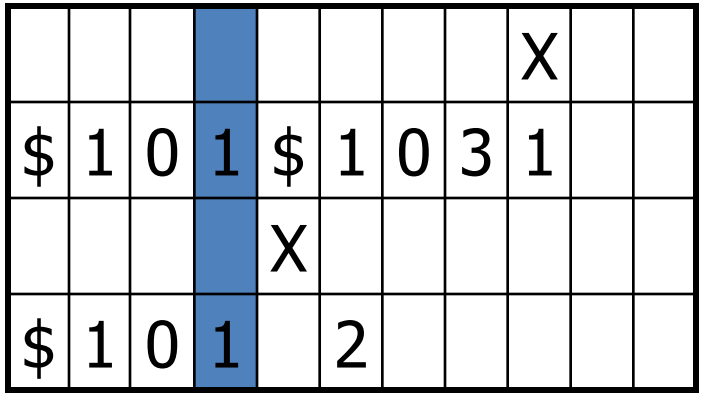


na 2nd fita: ⊟⊠2,L

2*k*-trilhas Simulando *k*-fitas



simulação por
TM de 2k-trilhas



na 2nd fita: ⊟⊠2,L

2*k*-trilhas Simulando *k*-fitas

simulação por
TM de 2k-trilhas

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

TM *k*-trilha
transição

$	1	0	1	$	1	0	3	1		
$	1	0	1		2					

								X		
$	1	0	1	$	1	0	3	1		
				X						
$	1	0	1		2					

2*k*-trilhas Simulando *k*-fitas

simulação por
TM de 2k-trilhas

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

TM *k*-trilha
transição

$	1	0	1	$	1	0	3	1		
$	1	0	1		2					

								X		
$	1	0	1	$	1	0	3	1		
				X						
$	1	0	1		2					

2*k*-trilhas Simulando *k*-fitas

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

TM *k*-trilhas
transição

$	1	0	1	$	1	0	3	1	
$	1	0	1		2				

simulação por
TM de 2k-trilhas

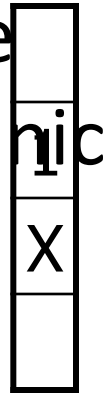
								X	
$	1	0	1	$	1	0	3	1	
				X					
$	1	0	1		2				

pronta p/ próxima
transição *k*-fita

1 fita simulando *k*-trilhas

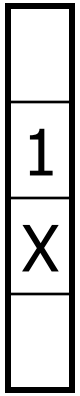
Finalmente, converter a máquina 2*k*-trilhas para uma máquina com uma única fita. De certo modo, já concluímos, já que há apenas uma cabeça de leitura.

Q: Como a informação de uma coluna de trilhas pode ser armazenada em uma única célula?



1 fita simulando *k*-trilhas

R: Basta expandir o alfabeto da fita Γ! Esse caso, o novo alfabeto consistiria de 4-tuplas, e a 4-tupla (□,1,X,□) representaria a coluna à direita.



Portanto, uma TM *k*-trilhas torna-se uma *TM 1-fita* se a enxergamos da maneira apropriada. Nenhuma modificação adicional é necessária.

Isso completa a simulação da M *k*-trilhas por uma TM 1-fita. □

Convertendo uma NTM para Determinista

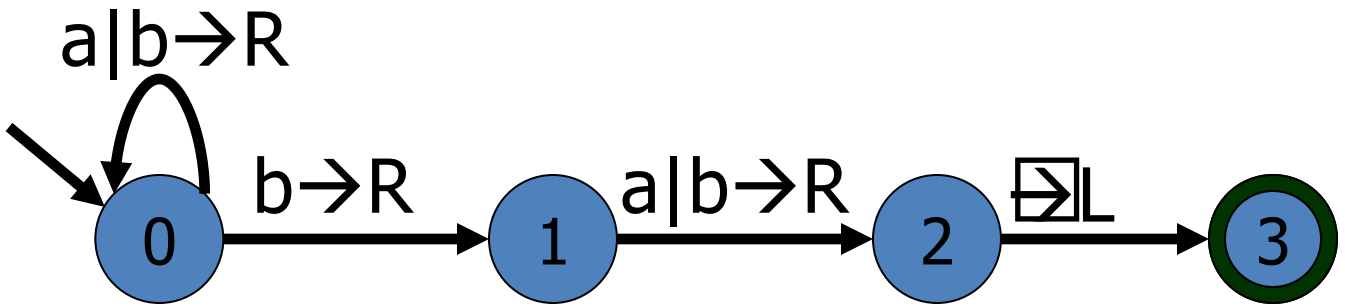
THM: Toda não determinista pode ser convertida em uma TM determinista equivalent.

A idéia da prova é a seguinte:

1) Simule a NTM por meio de uma TM 3-fitas determinista.
2) Os resultados anteriores garantem que podemos converter essa máquina em uma TM determinista de 1-fita.

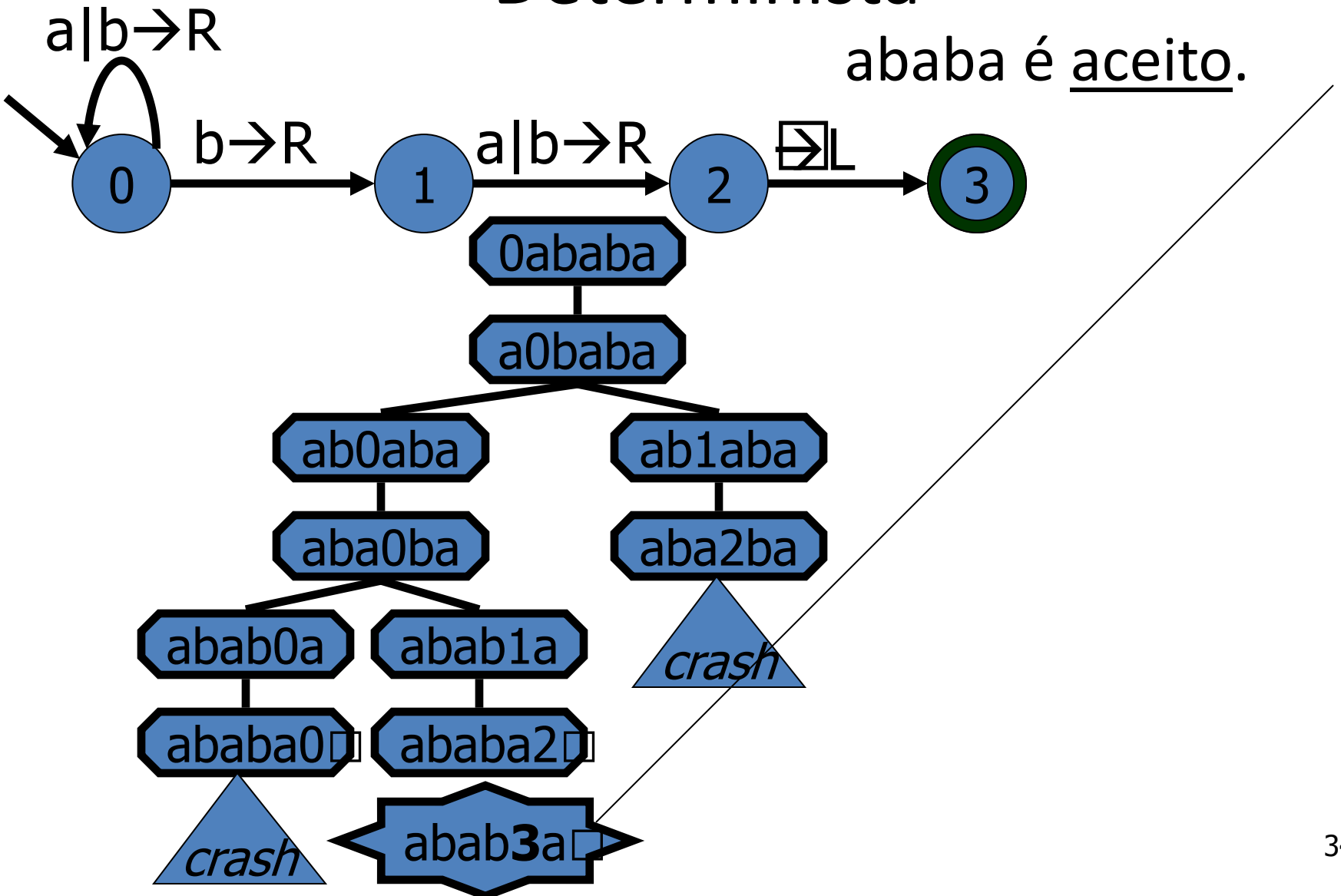
Convertendo uma NTM para Determinista

Considere a seguinte TM decisora para (a∪b)*b(a∪b):



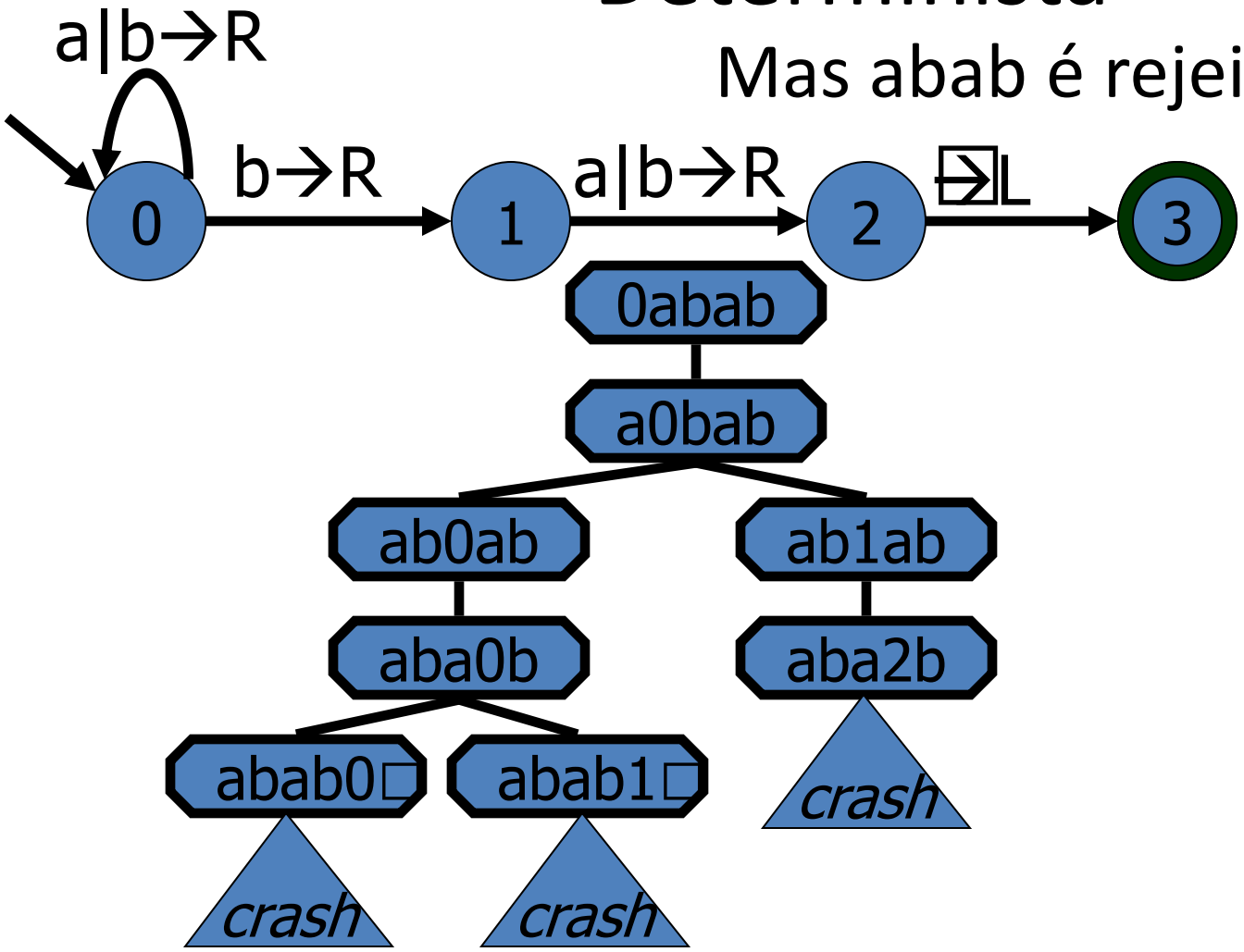
Para verificar se um string é aceito, precisamos ver se algum ramo da computação leva a uma configuração que contenha o estado “3”:

Convertendo uma NTM para Determinista



Convertendo uma NTM para Determinista

Mas abab é rejeitado:



Q: Como converter isso em um algoritmo?

Convertendo uma NTM para Determinista

R: Construa a árvore de computação da TM a partir da configuração inicial. E então verifique se ocorre alguma configuração de aceitação em algum ramo.

Q: Pesquisa em largura ou pesquisa em profundidade?

Convertendo uma NTM para Determinista

R: Pesquisa em largura! Se a NTM tem algum loop infinito, podemos prosseguir em um ramo infinito, se usarmos pesquisa em profundidade. Por outro lado, como cada nodo tem número finito de filhos, uma BFS garante que atingiremos uma configuração de aceitação, caso exista.

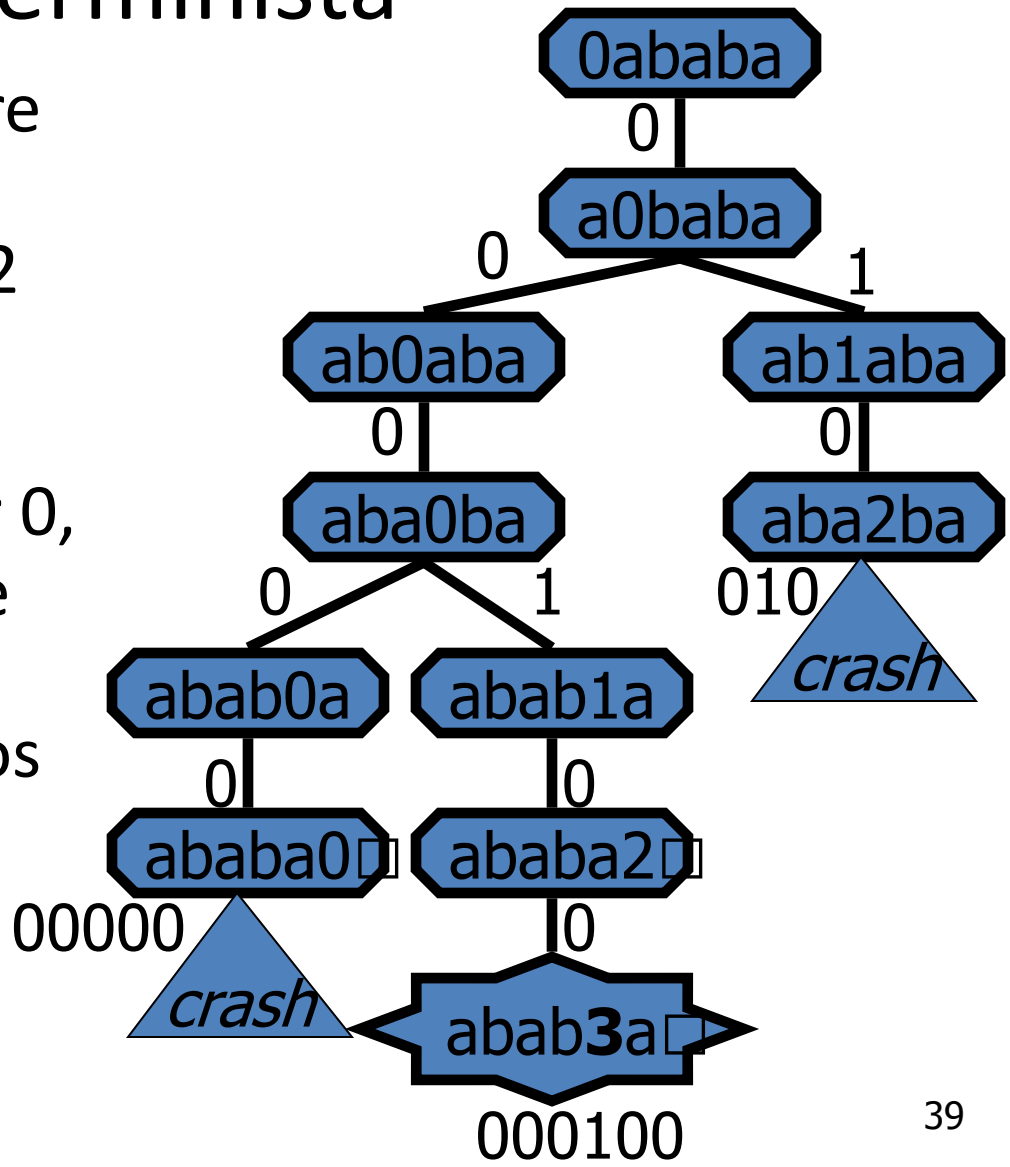
Convertendo uma NTM para Determinista

Construindo a árvore de computação BFS:

- Usamos uma TM 3-fitas
- A primeira fita apenas contém a entrada
- A segunda fita simula a computação em um ramo particular até determinada profundidade
- A última fita contém as instruções de desvio para o ramo de computação que está sendo simulado

Convertendo uma NTM para Determinista

Neste exemplo, a árvore de computação é binária: no máximo 2 escolhas em cada passo. Denote a primeira escolha por 0, e a segunda por 1 de modo a obter identificadores únicos para os ramos:



Convertendo uma NTM para Determinista

BFS prosseguiria do seguinte modo:

1. Tente 0, se configuração de aceitação, aceita.
2. Tente 1, se configuração de aceitação, aceita.
3. Tente 00, se configuração de aceitação, aceita.
4. Tente 01, se configuração de aceitação, aceita.
5. Tente 10, se configuração de aceitação, aceita.
6. Tente 11, se configuração de aceitação, aceita.
7. Tente 000, se configuração de aceitação, aceita.
8. Tente 001, se configuração de aceitação, aceita.

… No nosso caso, finalmente o string seria aceito em:

67. Tente 000100, ACEITA!

Convertendo uma NTM para Determinista

Para máquinas reconhecedoras, a conversão é desse modo.

Para máquinas decisoras, o lema de König implica que a árvore de computação é de fato finita, para qualquer entrada. Portanto poderíamos construir toda a árvore e dizer quando um string é rejeitado depois de examinar a árvore toda.

Máquina FIFO Simuladora
Formalização

Função de transição de estados:

$$\delta_{\text{queue}}(x,y) = \begin{cases} (y,x) \text{ se } y \in \Gamma \\ (x',z) \text{ se } y \in Q \text{ e } \delta(y,x) = (z, x', \text{L}) \\ (z_{copy},x') \text{ se } y \in Q \text{ e } \delta(y,x) = (z, x', \text{R}) \\ (z, y) \text{ se } x = z_{copy} \in Q_{copy} \end{cases}$$