Teoria da Computação

Prof. Sergio D Zorzo
Departamento de Computação - UFSCar
2023/2
6

Não-determinismo

Em Máquinas de Turing

Definição formal da função de transição de uma MT

- Função de transição
  - $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{E,D\}$
  - Os argumentos de $\delta(q,X)$ são:
    - um estado q e um símbolo de fita X
  - O valor de $\delta(q,X)$ (se definido) é uma tripla (p,Y,S), onde:
    - p é o próximo estado em Q
    - Y é o símbolo, em Γ, gravado na célula que está sendo varrida, substituindo o símbolo que estava nessa célula
    - S é um sentido, ou direção, em que a cabeça se move
      - E = esquerda, D = direita
      - L = left, R = right

Não-determinismo em MTs

- “a definição de MTs que vimos até agora é determinística ou não-determinística?”
  - Resposta: determinística – a todo momento, sempre se sabe o que fazer
  - Em autômatos finitos, não-determinismo não aumenta a capacidade de reconhecimento de linguagens!
  - Em autômatos de pilha, não-determinismo aumenta a capacidade de reconhecimento de linguagens!
  - E em máquinas de Turing???

Não-determinismo em MTs

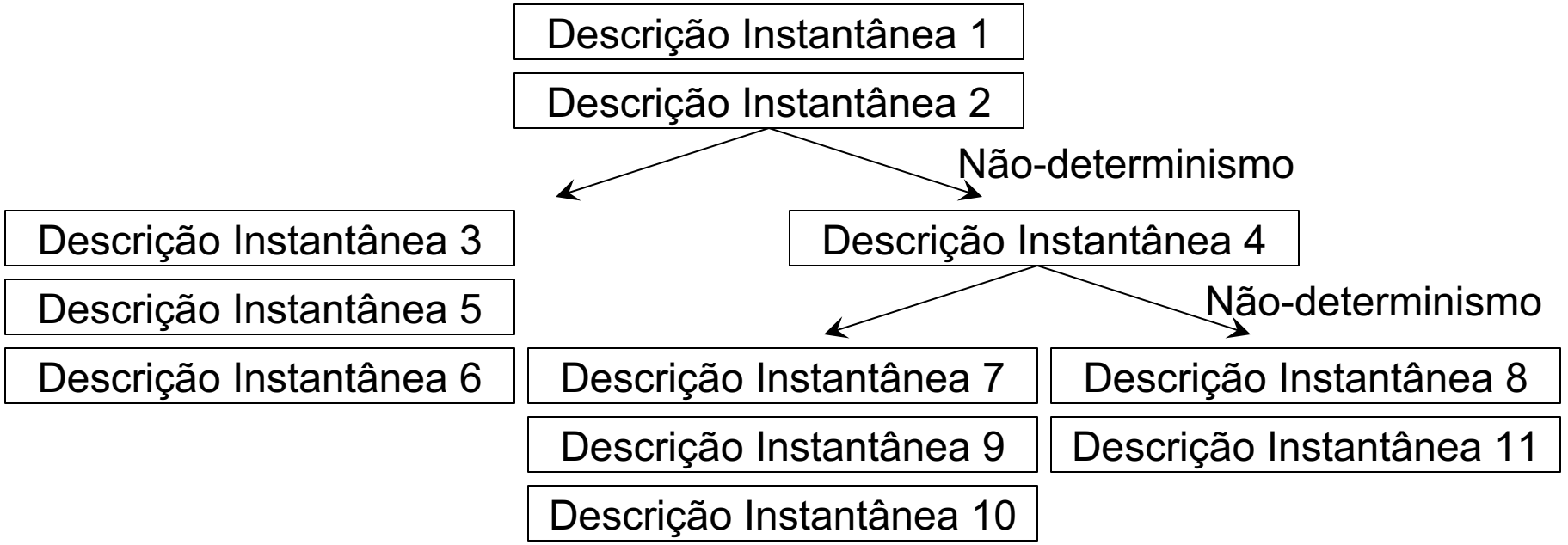
- Antes, veremos o que significa não-determinismo em MTs
- A única diferença está na função de transição:
  - $\delta: Q \times \Gamma \rightarrow \mathbf{P(Q \times \Gamma \times \{E,D\})}$
  - Os argumentos de $\delta(q,X)$ são:
    - um estado q e um símbolo de fita X
  - O valor de $\delta(q,X)$ (se definido) é um **conjunto de triplas** (p,Y,S)

MTs não-determinísticas

- A linguagem aceita por uma NTM (Nondeterministic Turing Machine) é definida de forma similar a outros autômatos
  - A cada ponto de dúvida, analisam-se todas as possibilidades, se nenhuma levar à aceitação, a cadeia não é aceita. Se ao menos uma levar à aceitação, a cadeia é aceita
- É possível provar que, para toda NTM que reconhece uma linguagem L, é possível construir uma DTM que reconheça L
  - Ou seja: NTMs e DTMs são equivalentes em termos de capacidade de reconhecimento de linguagens!!

Conversão NTM → DTM

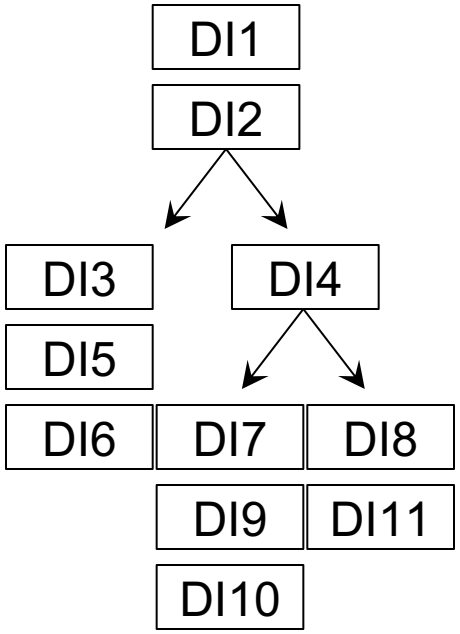
- A idéia é construir uma DTM que explora os diferentes pontos de não-determinismo, testando todas as possibilidades
- Como faríamos no papel?

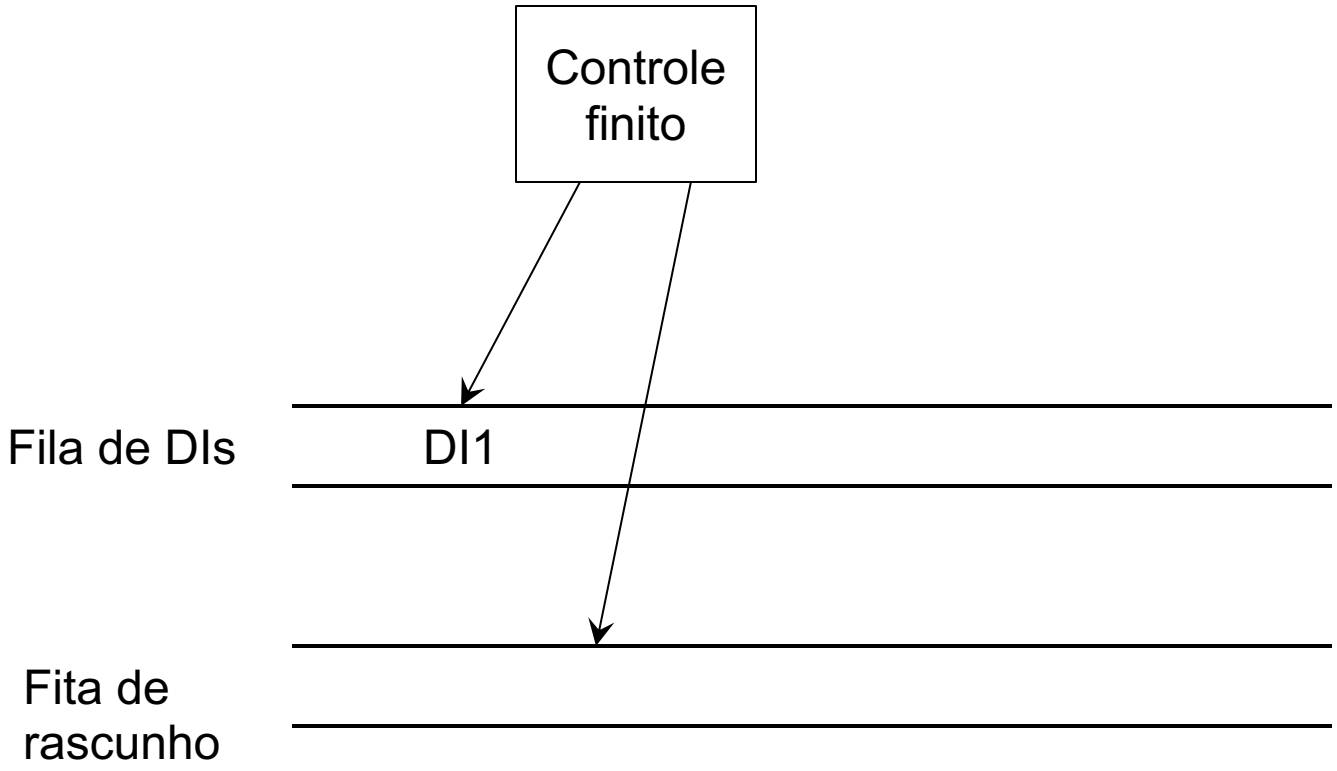


Conversão NTM → DTM

- A DTM fará isso
  - Usará uma fita para armazenar, numa fila, as DIs a serem testadas
    - Nessa fita, uma trilha adicional é usada para “marcar” qual a DI sendo atualmente testada
  - Usará outra fita para ajudar a fazer “cópias” de uma DI imediatamente antes de um ponto de não-determinismo
    - Se, em uma determinada DI, existem k possibilidades de escolha, serão criadas k cópias daquela DI
    - Essas cópias serão armazenadas no fim da fila (fita 1)

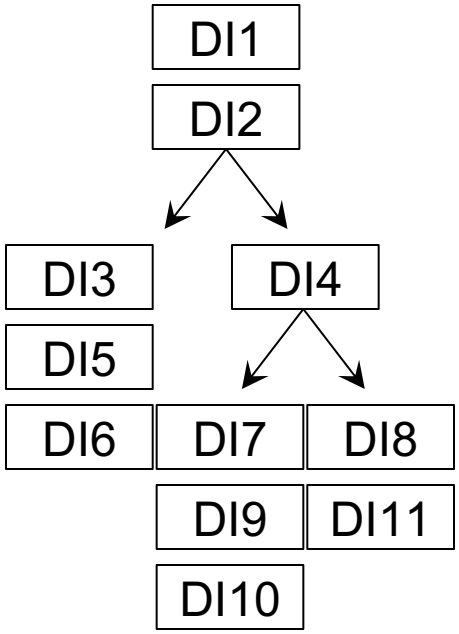
Conversão NTM → DTM

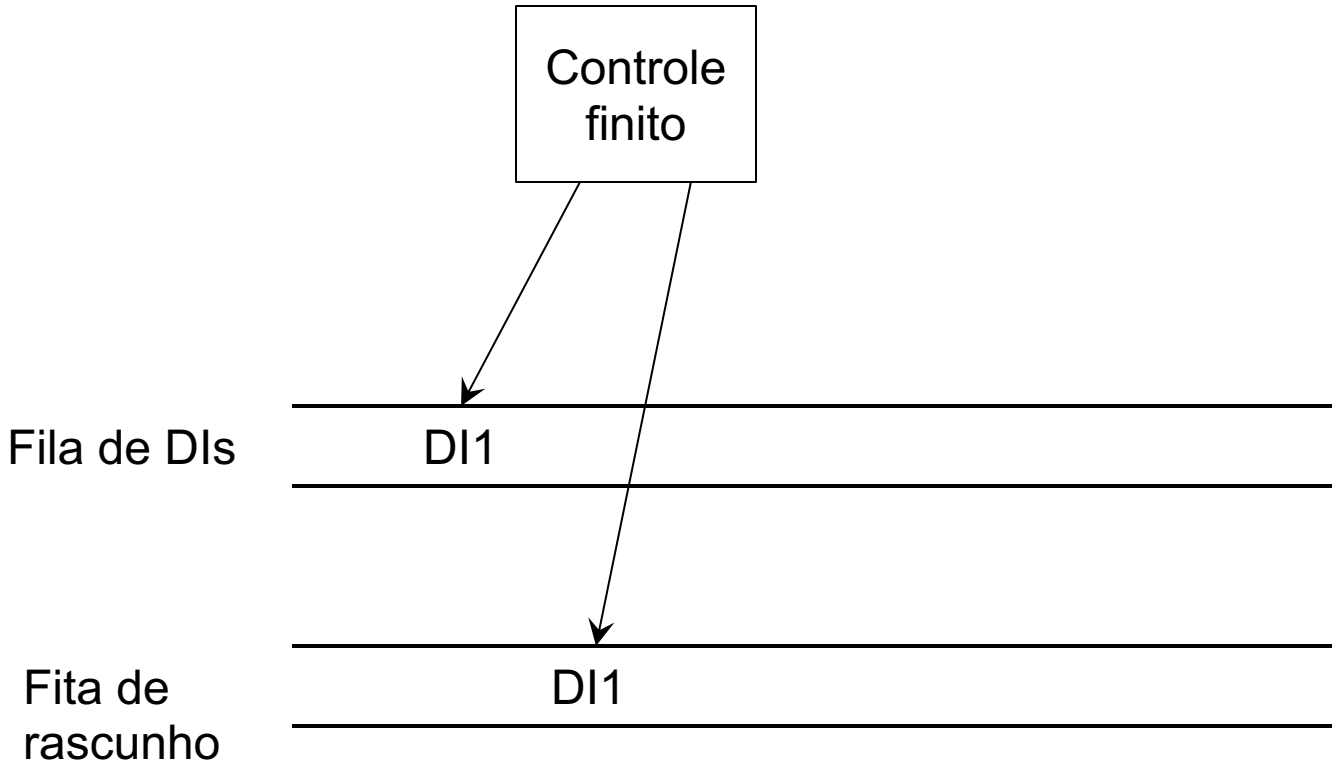




A execução começa. DI1 está na primeira fita. DI1 não é uma DI de aceitação, portanto DTM precisa seguir em frente.

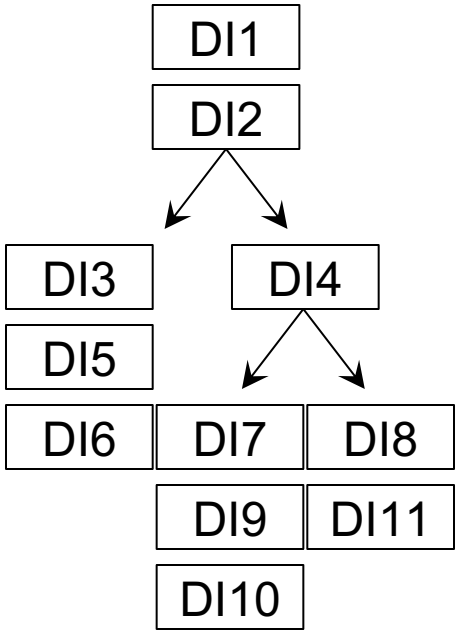
Conversão NTM → DTM

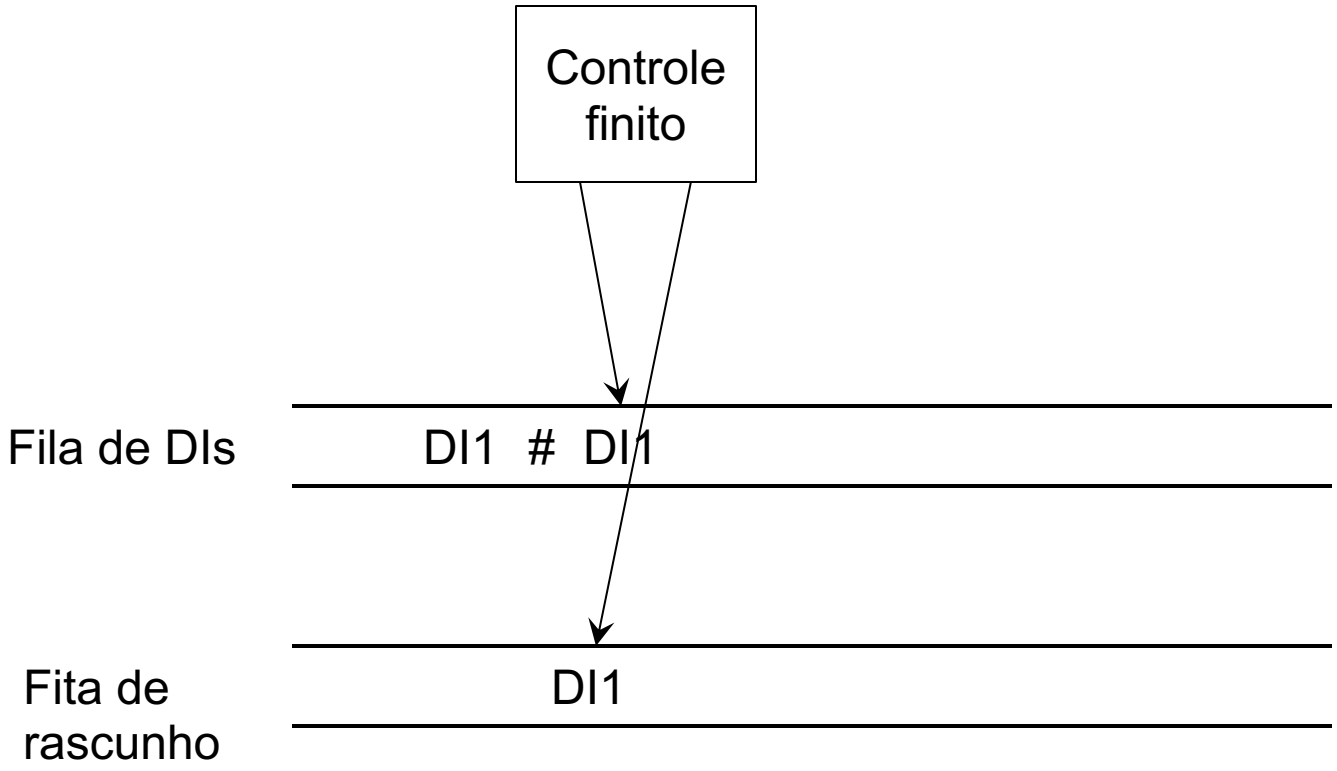




DTM avalia o número de possibilidades a partir de DI1. Só existe 1 alternativa, e portanto DTM vai criar uma cópia de DI1. Para isso, ela insere DI1 temporariamente na fita de rascunho, que irá servir de “memória” durante a cópia

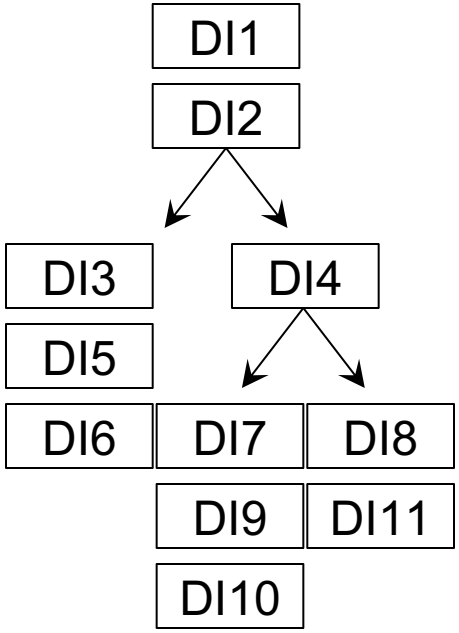
Conversão NTM → DTM

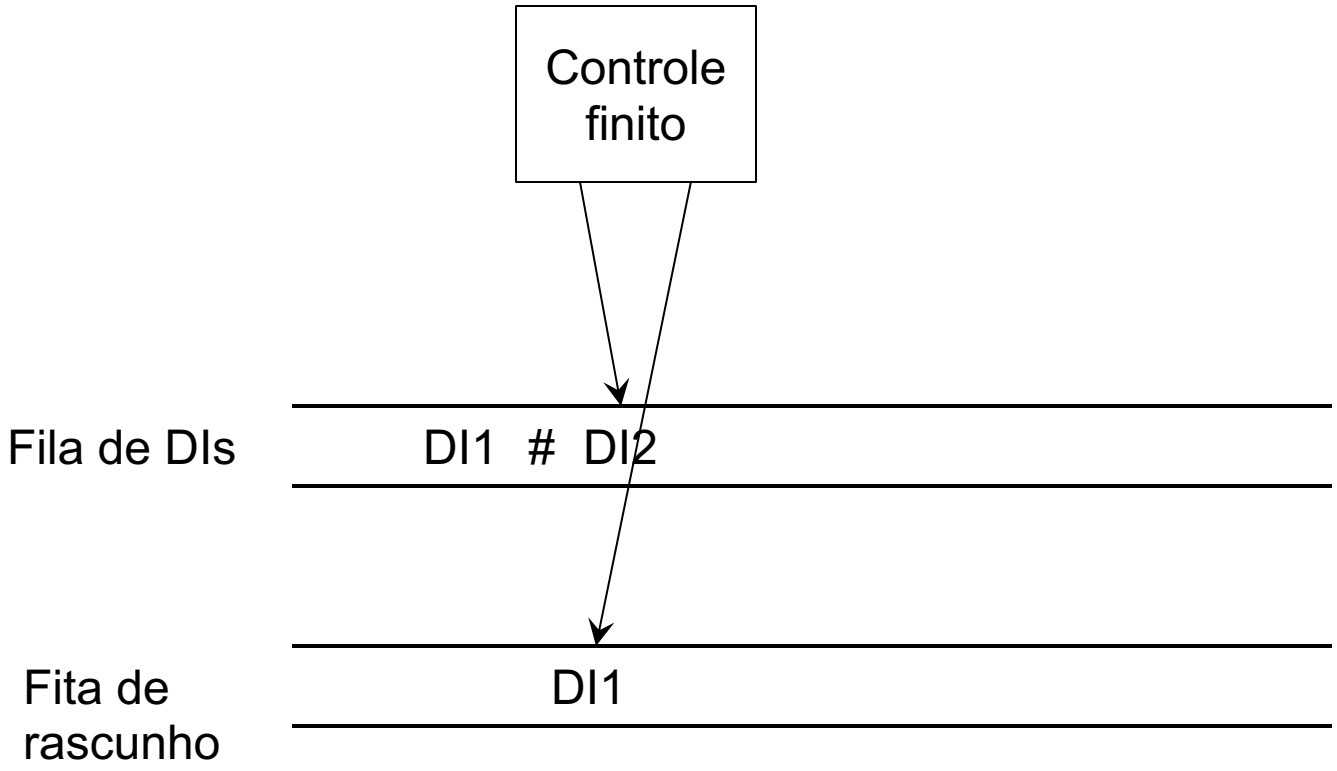




Tendo armazenado DI1 na fita de rascunho, a DTM faz uma cópia, inserindo-a no final da fila. Um símbolo especial (#) serve de separador entre as DIs na fila

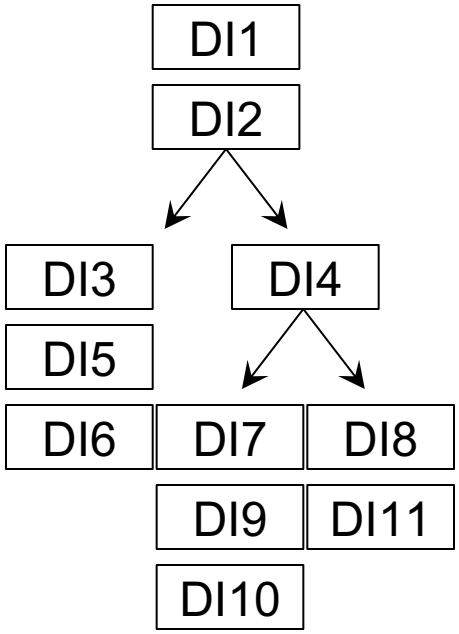
Conversão NTM → DTM

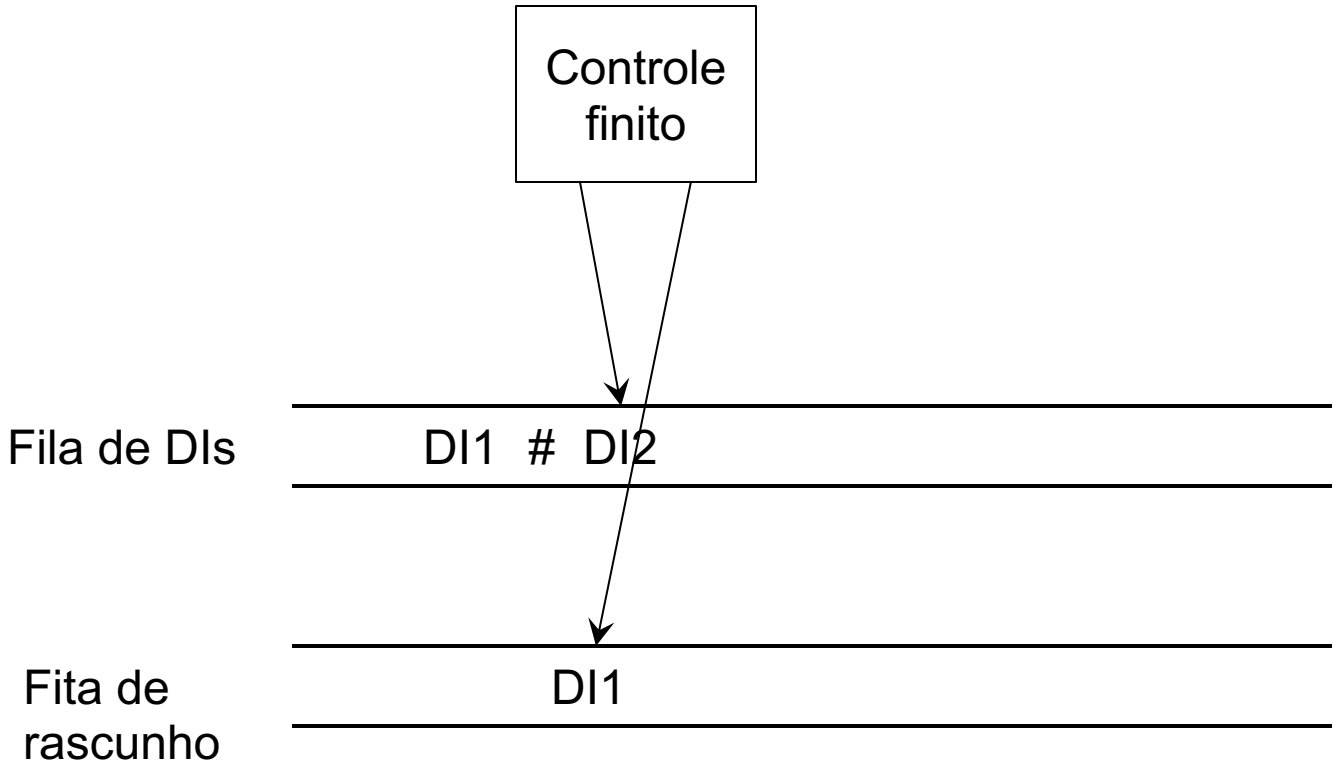




DTM então modifica cada cópia de acordo com as possibilidades de movimento.
Nesse caso, só há uma cópia, e uma possibilidade de movimento (DI2).

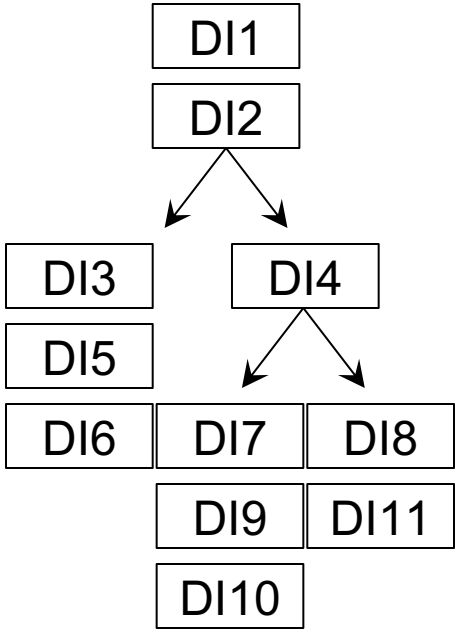
Conversão NTM → DTM

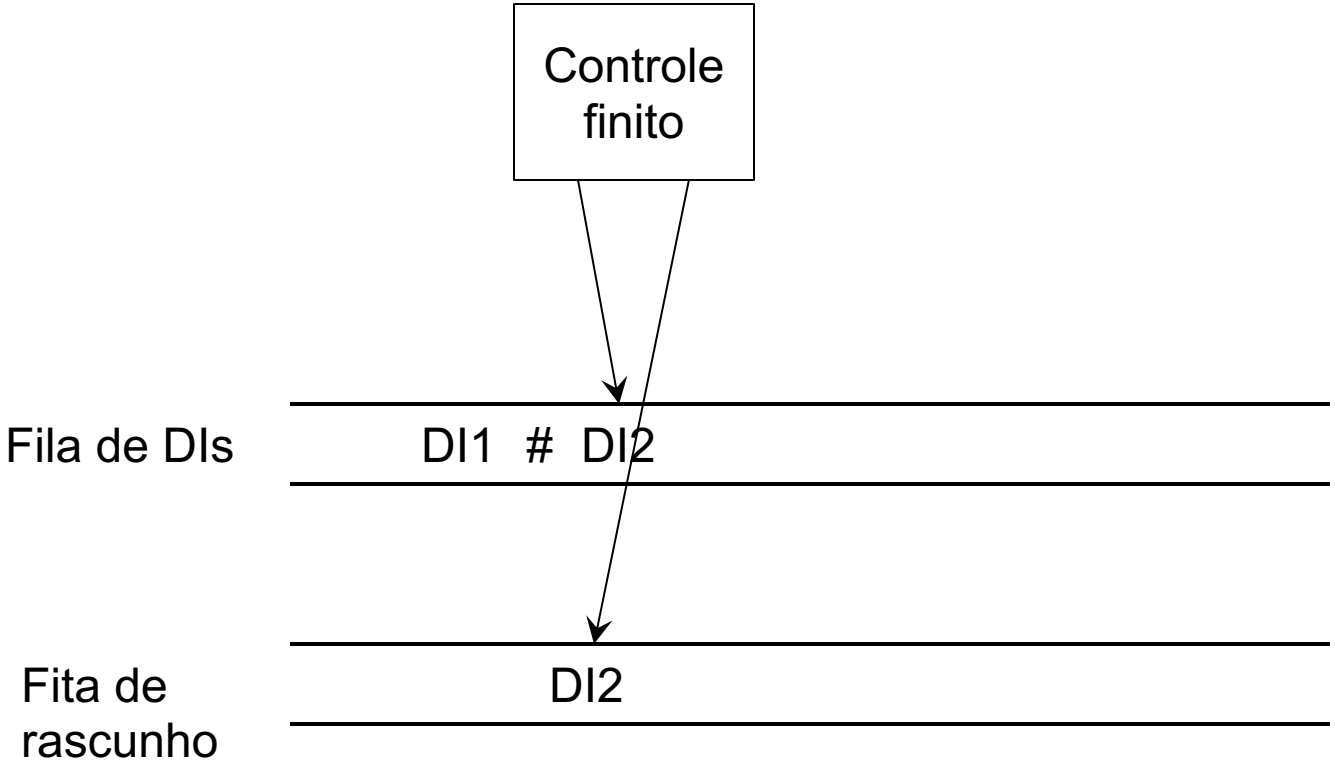




Avaliando DI2, DTM descobre que esta não é uma DI de aceitação. Portanto, ela precisa ir em frente. Avaliando as possibilidades, ela identifica 2 alternativas, portanto DI2 deverá ser copiada 2 vezes.

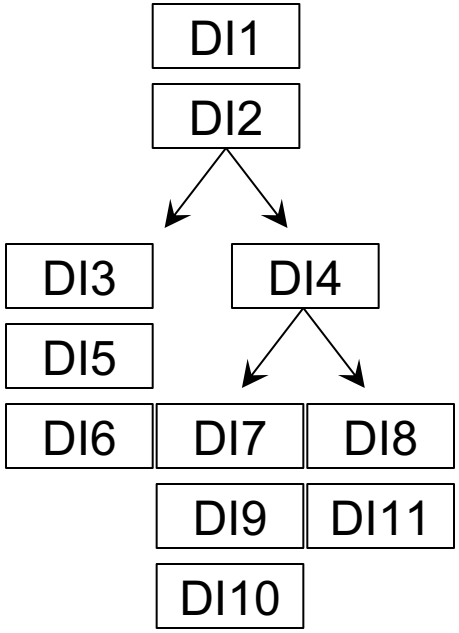
Conversão NTM → DTM

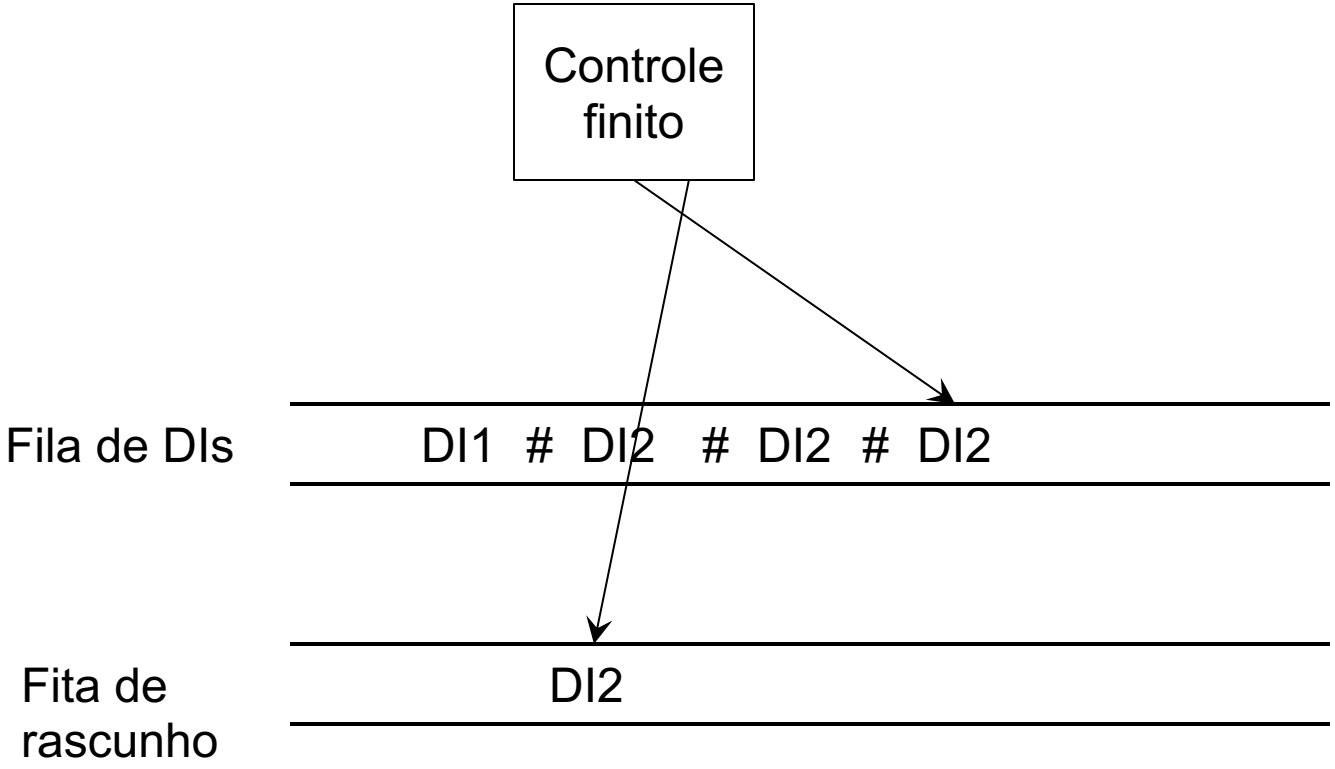




Para fazer a cópia, DTM usa a fita de rascunho como memória, inserindo DI2 no lugar do valor anterior.

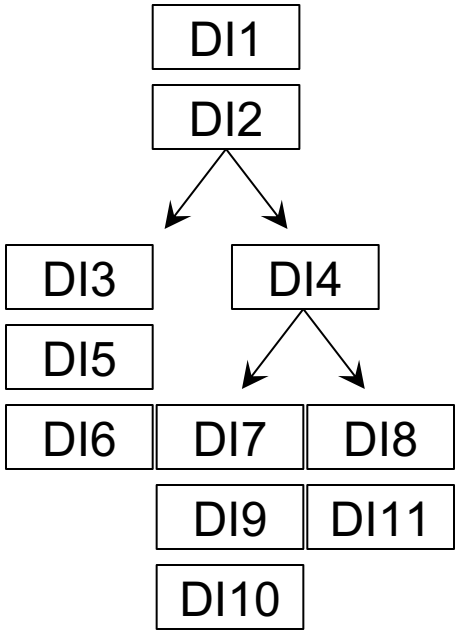
Conversão NTM → DTM

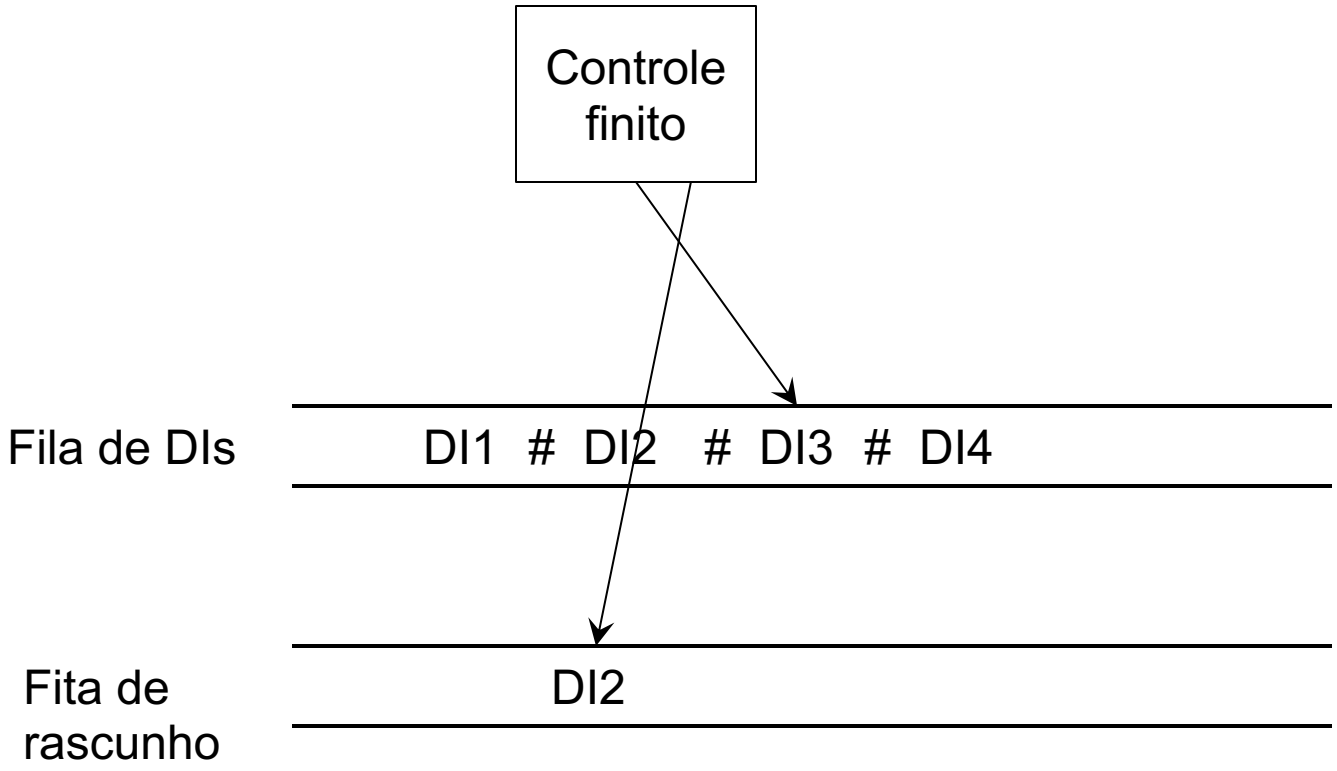




Em seguida, DTM insere 2 cópias de DI2 no final da fila, copiando a partir do rascunho.

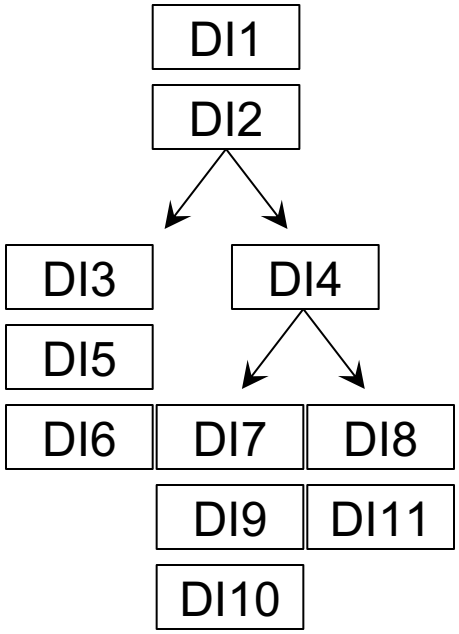
Conversão NTM → DTM

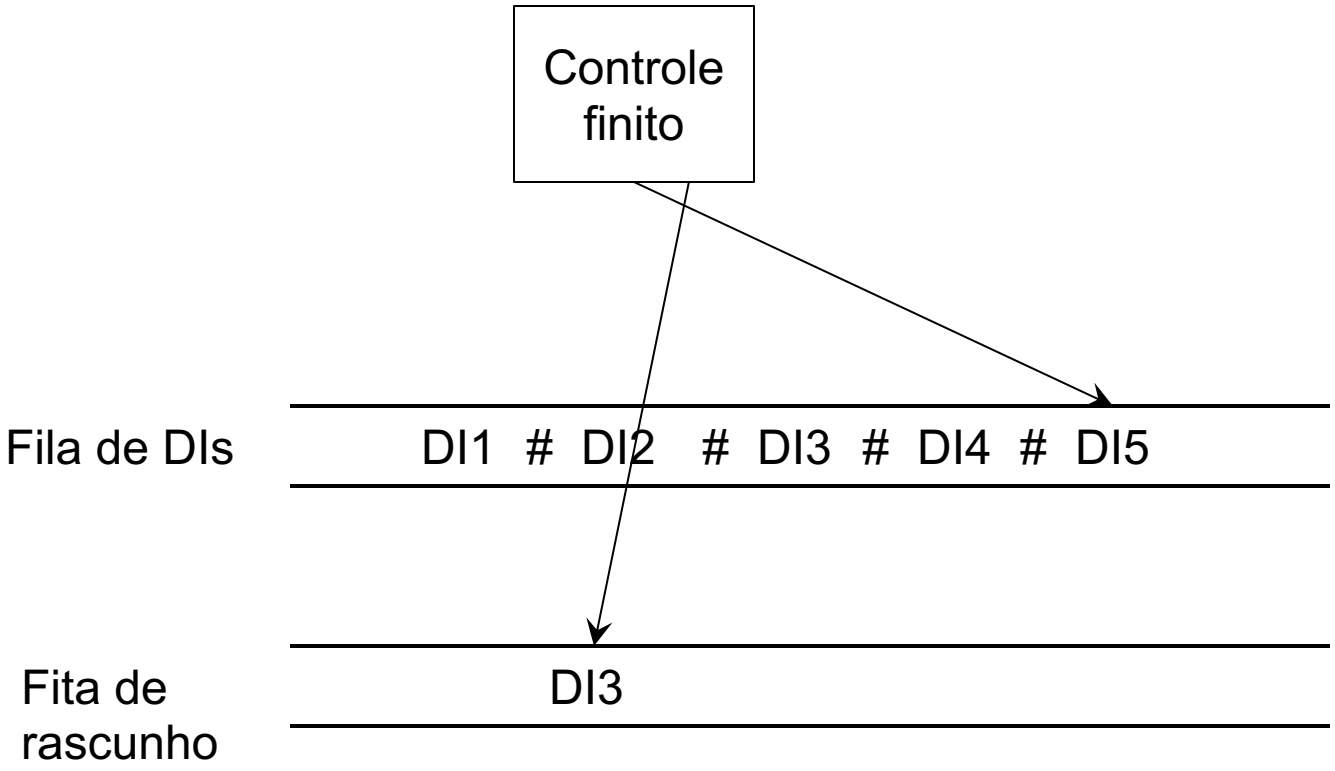




DTM faz o retorno, substituindo cada cópia de DI2 por uma alternativa diferente, no caso, DI3 e DI4

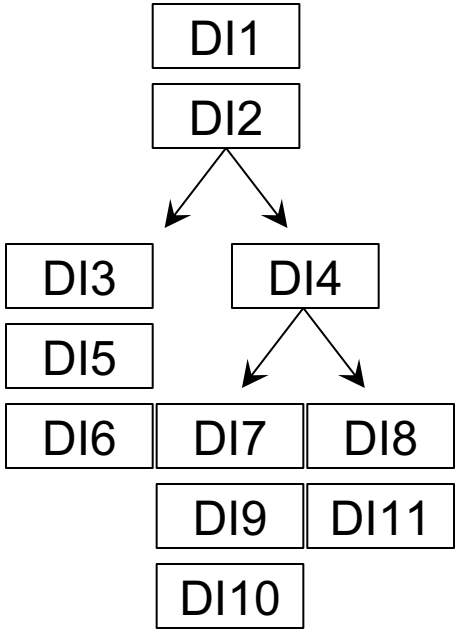
Conversão NTM → DTM

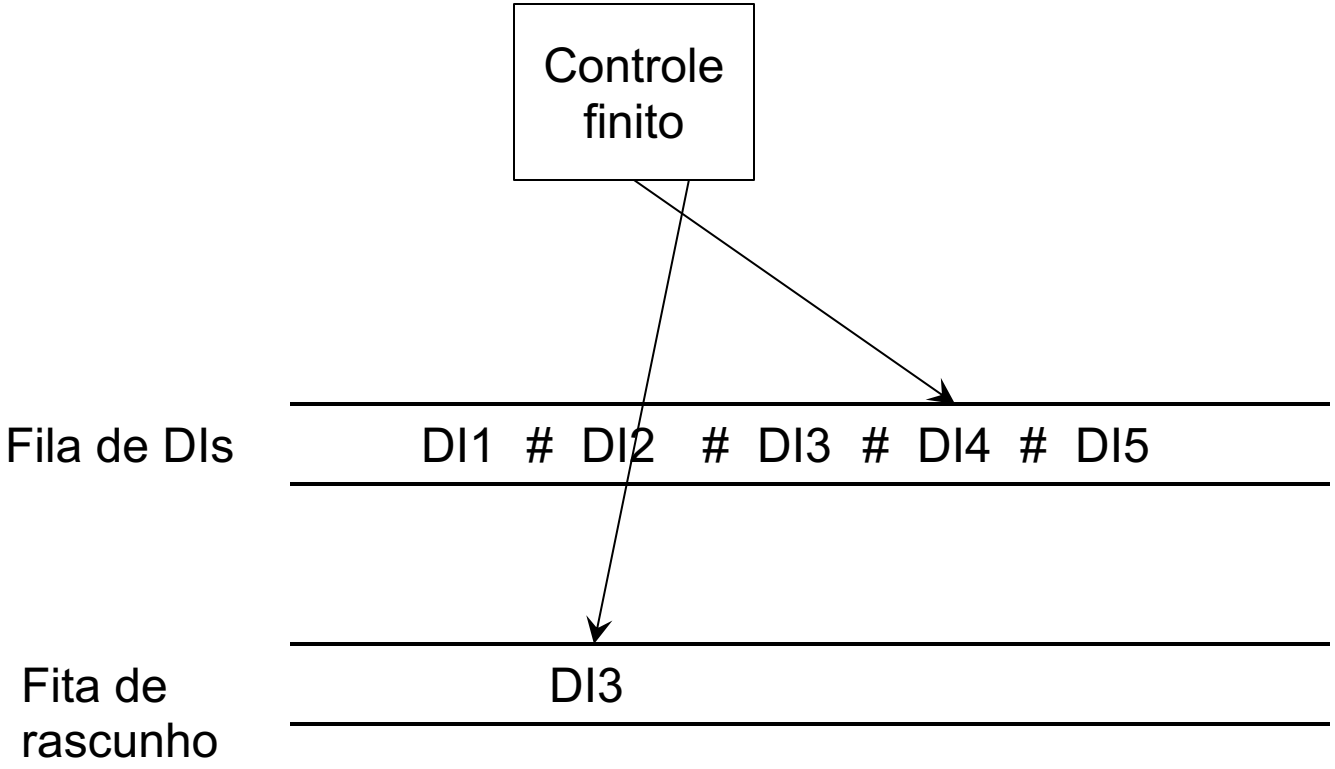




Não sendo uma DI de aceitação, DTM faz a análise de alternativas a partir de DI3, criando cópias no final da fila e alterando-as para o próximo movimento possível. Nesse caso, só existe a possibilidade de ir para DI5

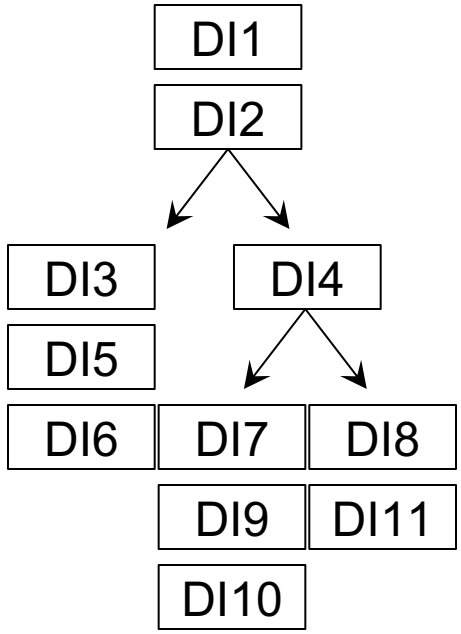
Conversão NTM → DTM

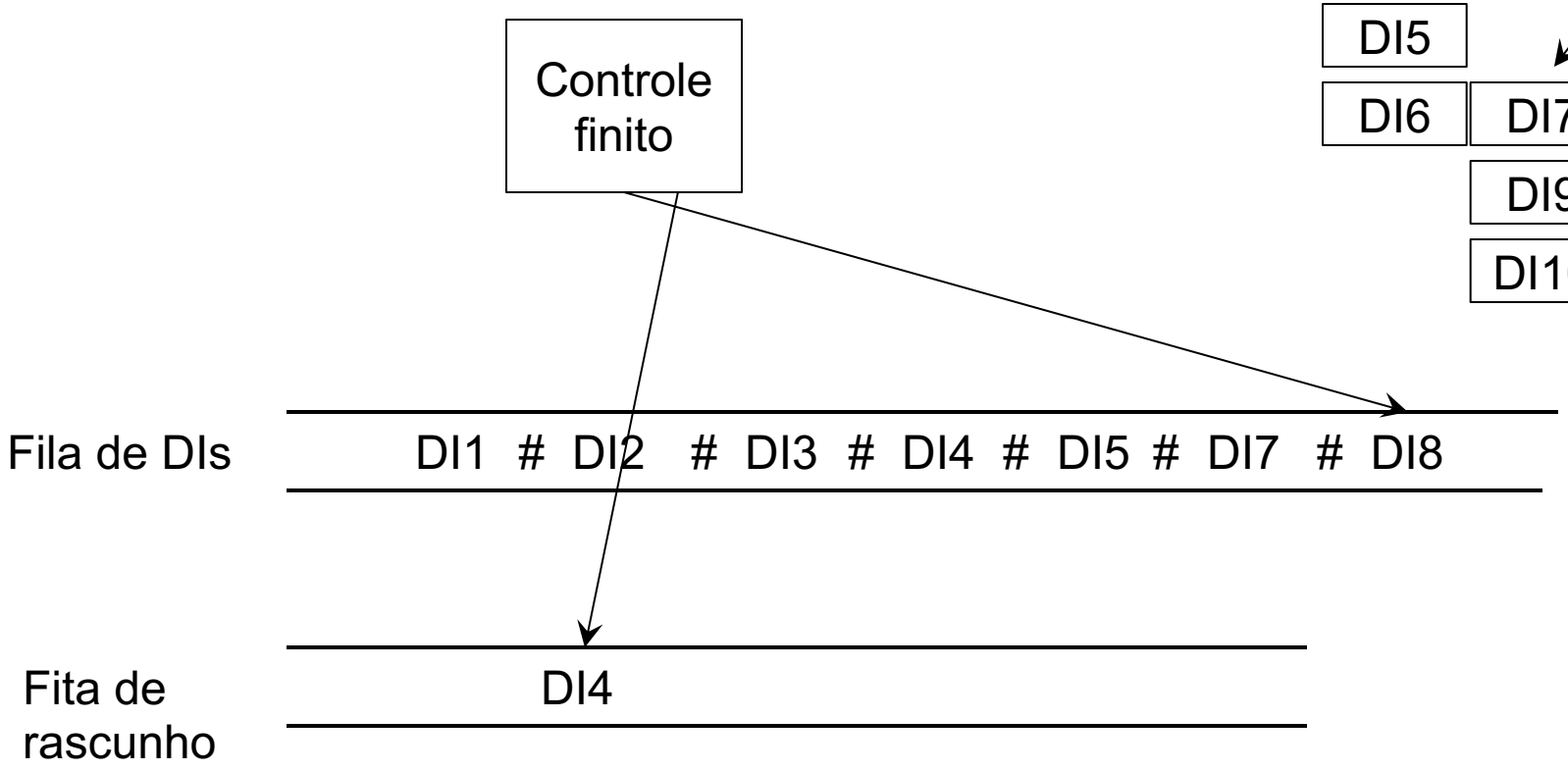




Antes de avaliar DI5, porém, DTM volta para a próxima DI não analisada, ou seja, DI4

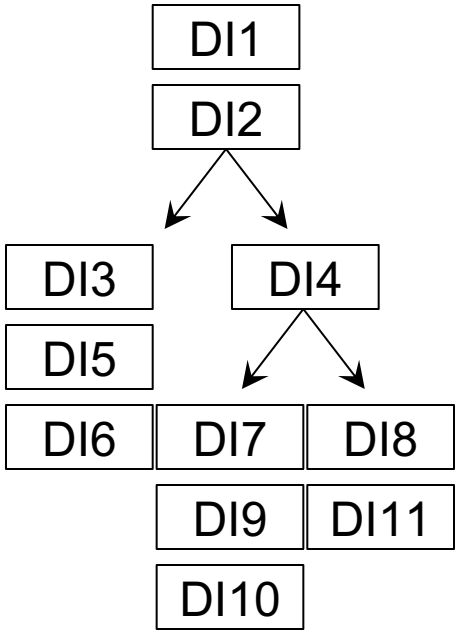
Conversão NTM → DTM

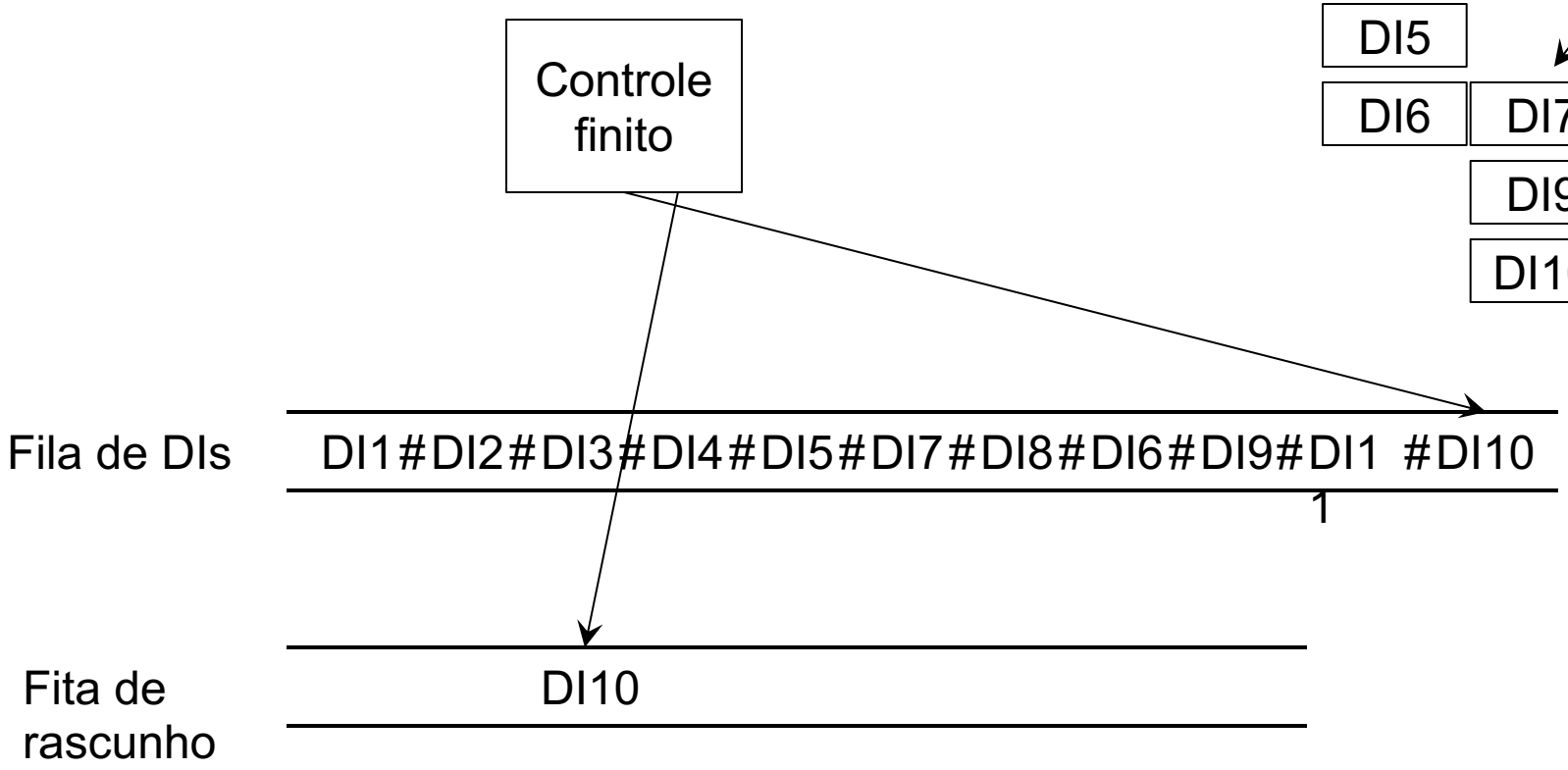




DI4 também não é uma DI de aceitação, e portanto novas DIs serão inseridas no final da fila, representando as possibilidades, DI7 e DI8

Conversão NTM → DTM





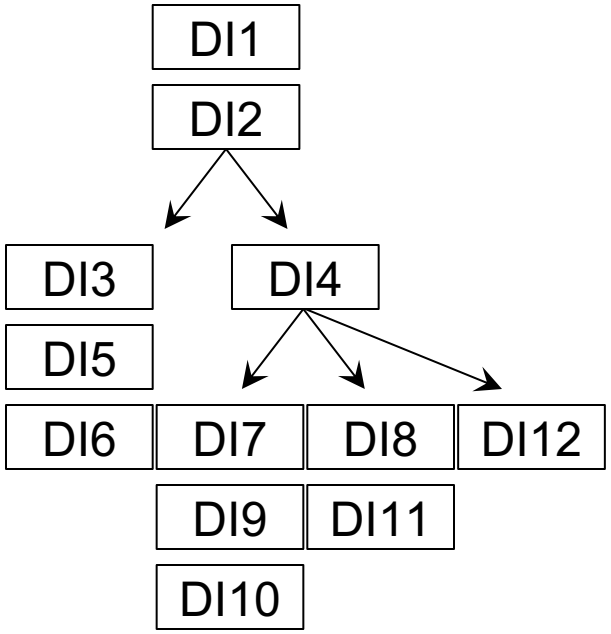
Esse procedimento continua até serem esgotadas todas as possibilidades.
Analise a fita resultante e note que se trata de uma busca “primeiro em amplitude”

Conversão NTM → DTM

- Durante esse percurso, caso a máquina entre em uma DI de aceitação, ela aceita e encerra a execução
  - Caso encontre uma DI de rejeição, ela continua na próxima DI da fila
- Essa simulação é precisa
  - DTM só aceita se NTM pode entrar em uma DI de aceitação

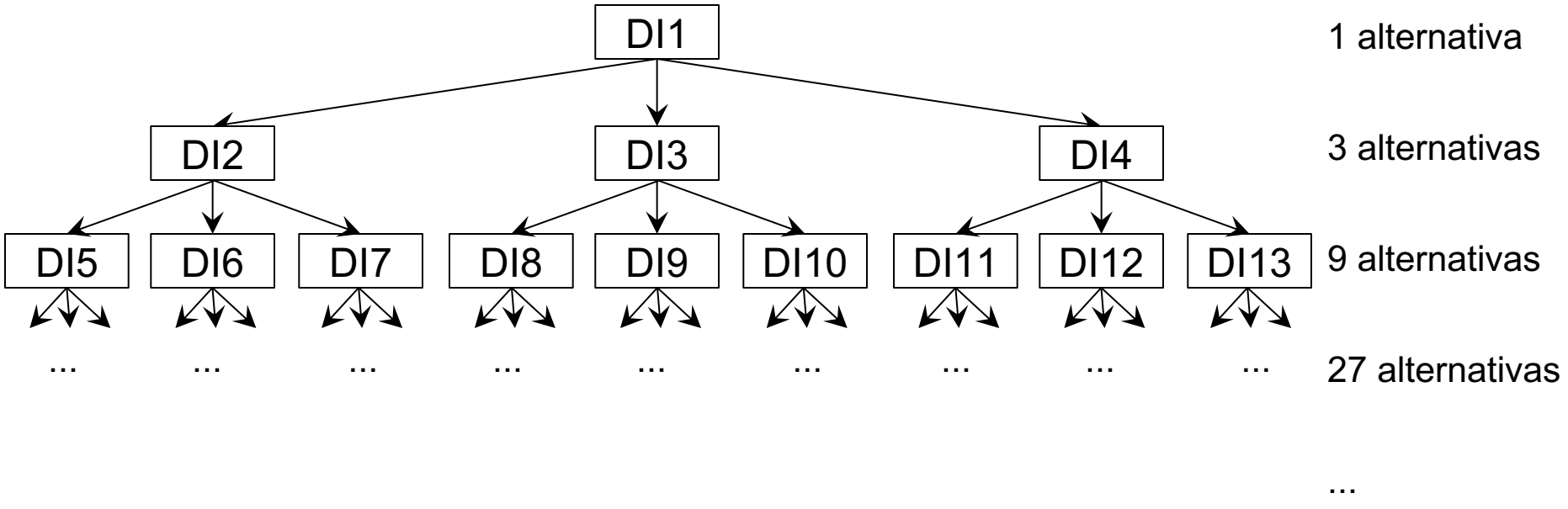
Conversão NTM → DTM

- Seja m o número máximo de escolhas que NTM tem em qualquer configuração
  - Nesse exemplo à direita, m=3 (DI4 → DI7, DI8, DI12)
- Vamos então analisar um caso extremo!
  - Em toda DI, há 3 escolhas!



Conversão NTM → DTM

- Essa figura ilustra um caso extremo de não-determinismo, onde, a cada DI, temos 3 escolhas



Conversão NTM → DTM

- Ou seja, vemos uma progressão:
  - 1,3,9,27,...
  - Ou: 3^0, 3^1, 3^2, 3^3, 3^4, ...
- Generalizando: m^0, m^1, m^2, m^3, m^4, ...
  - Onde m é o número máximo de escolhas que a NTM pode fazer em todas as DIs
- Pergunta: por quantas DIs a NTM pode passar, no máximo, em:
  - 1 movimento? Resp: 1 DI
  - 2 movimentos? Resp: 1+3 DIs
  - 3 movimentos? Resp: 1+3+9 DIs
  - 4 movimentos? Resp: 1+3+9+27 DIs

Conversão NTM → DTM

- Generalizando: em n movimentos, a NTM pode alcançar no máximo:
  - $1 + m + m^2 + m^3 + ... + m^n$ DIs
  - Esse número é no máximo nm^n DIs
- A DTM equivalente faz uma busca primeiro em amplitude, isto é:
  - Primeiro ela testa as DIs que seriam alcançadas em 0 movimentos da NTM
  - Depois ela testa as DIs que seriam alcançadas em 1 movimento da NTM
  - Depois as DIs que seriam alcançadas em 2 movimentos da NTM
  - E assim por diante

Conversão NTM → DTM

- Generalizando:
  - Se NTM leva n movimentos para chegar à aceitação...
  - A DTM levará no máximo nm^n movimentos!!
- Eventualmente a NTM chegará na DI de aceitação
  - Ou seja, se a NTM aceita, a DTM também o fará (mesmo que demorando mais)
  - Concluímos que L(NTM) = L(DTM)

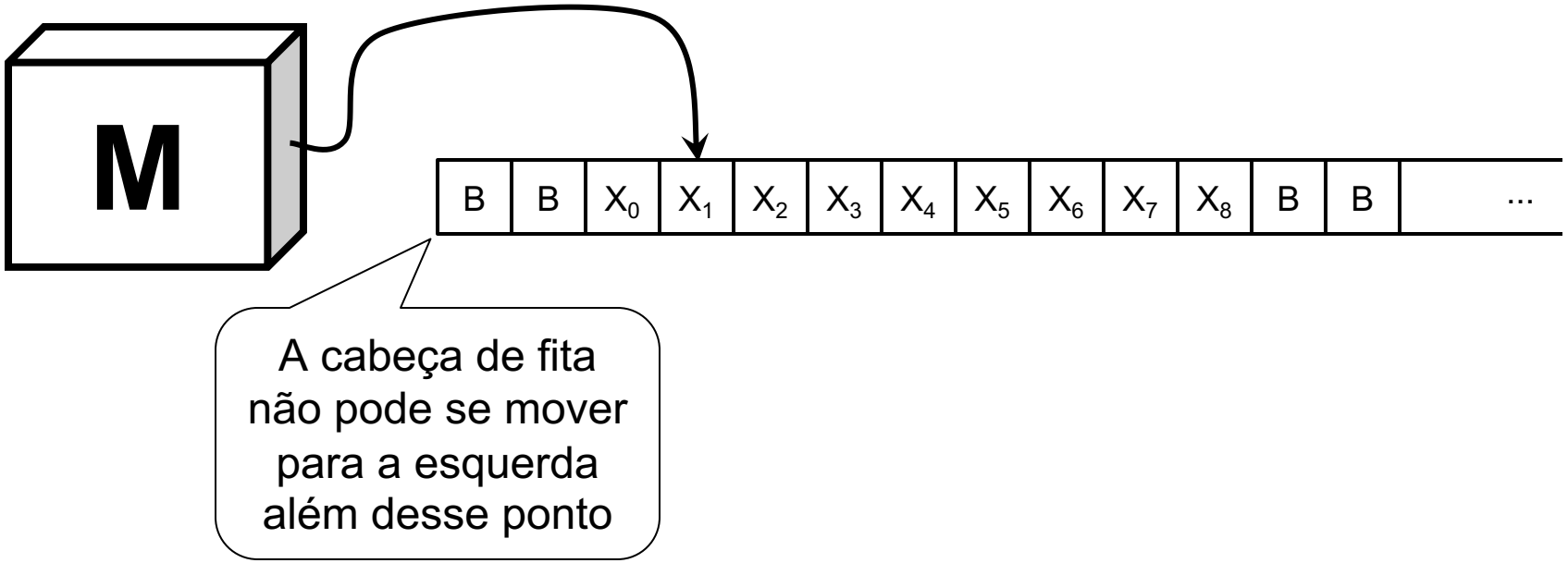
Conversão NTM → DTM

- Vimos que a DTM demora exponencialmente mais tempo que a NTM
  - Essa diferença tem importantes consequências práticas
- Mas não se sabe se essa lentidão é realmente necessária
  - Existe um modo melhor de simular uma NTM de forma determinística?
  - A resposta, e mais detalhes sobre isso, veremos mais adiante

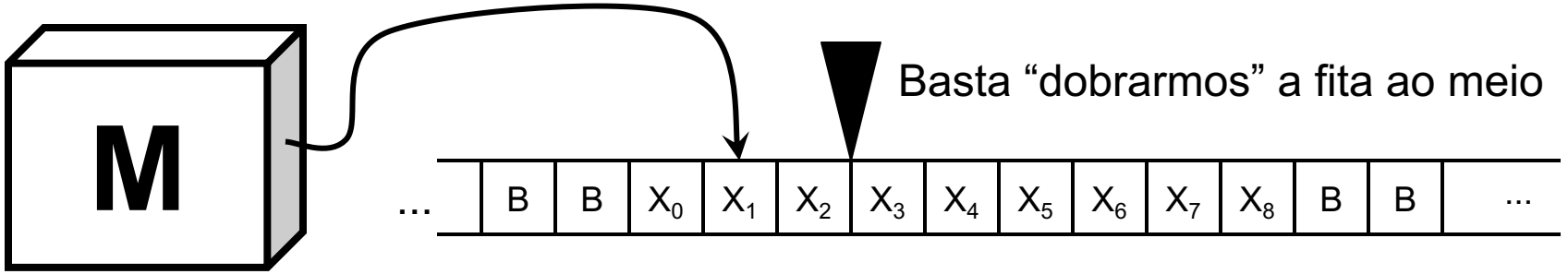
MTs restritas

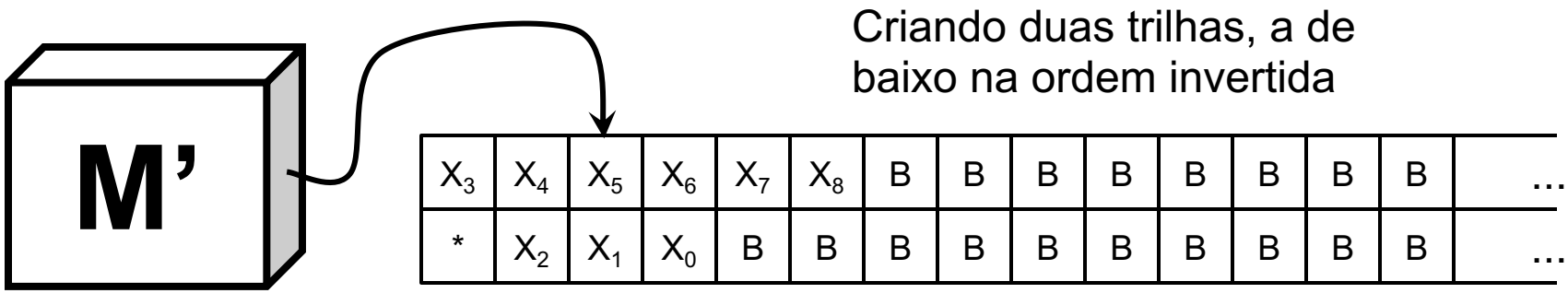
MT com fita semi-infinita

- Ou seja, a fita tem um início
  - A cabeça não pode se mover além desse ponto
- Veremos que a capacidade de processamento de linguagens é a mesma que as MTs com fita duplamente infinita

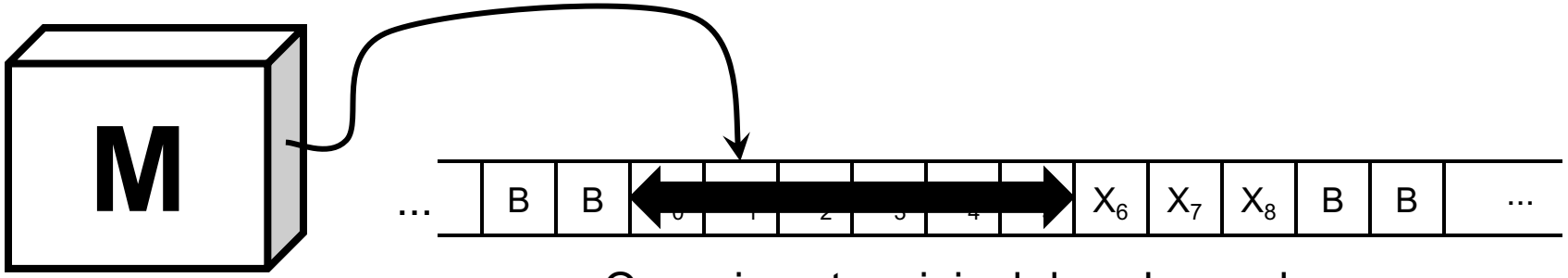


MT com fita semi-infinita

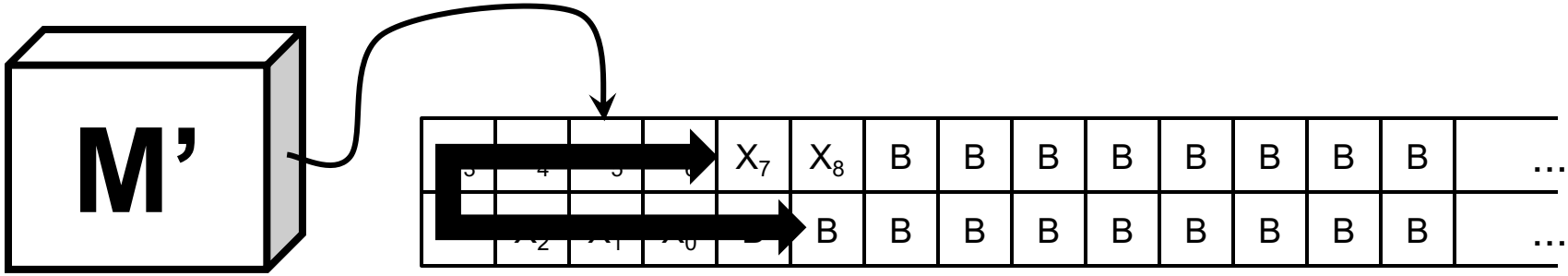




MT com fita semi-infinita



O movimento original da cabeça, da esquerda para a direita, pode ser simulado

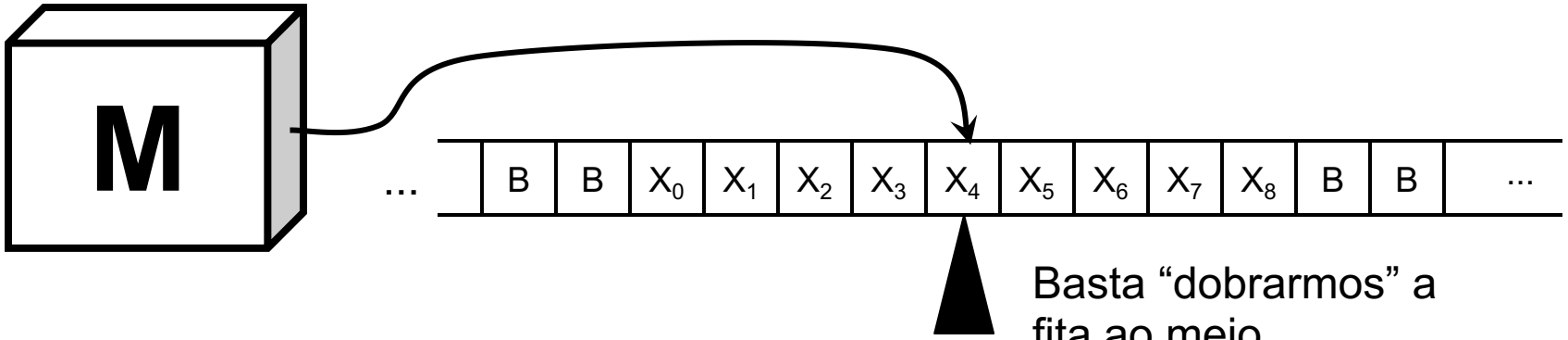


Sempre que o início da fita é alcançado, inverte-se a ordem de movimento, e troca-se de trilha

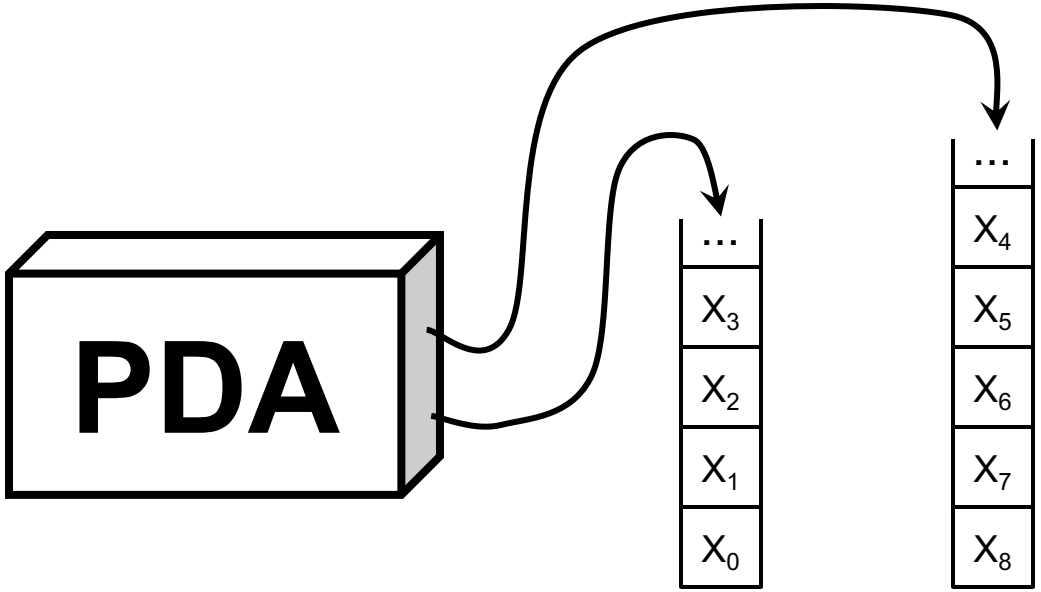
PDA com duas pilhas

- Um PDA com uma única pilha aceita as linguagens livres de contexto (Tipo-2)
- Mas acrescentando uma segunda pilha, é possível simular completamente uma Máquina de Turing!
  - Passando a aceitar as linguagens recursivamente enumeráveis (Tipo-0)
- Veremos essa construção a seguir

PDA com duas pilhas



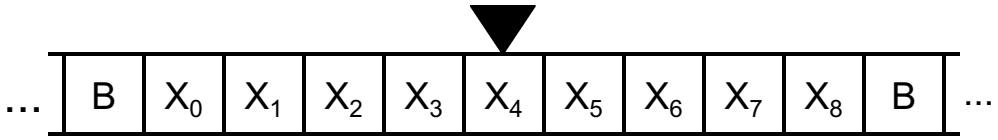
Basta “dobrarmos” a fita ao meio

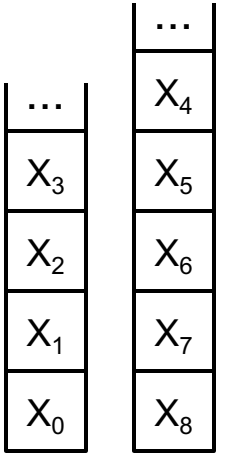


E criar duas pilhas:
- Uma contendo os símbolos à esquerda da cabeça
- Outra contendo os símbolos à direita da cabeça

PDA com duas pilhas

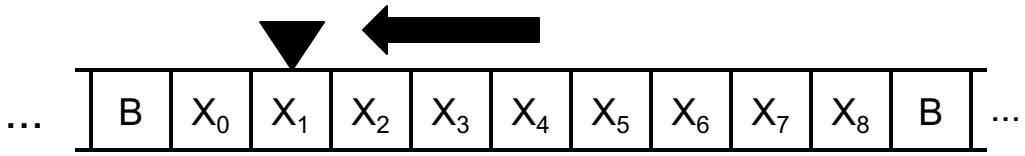
Como simular o movimento da cabeça na fita?

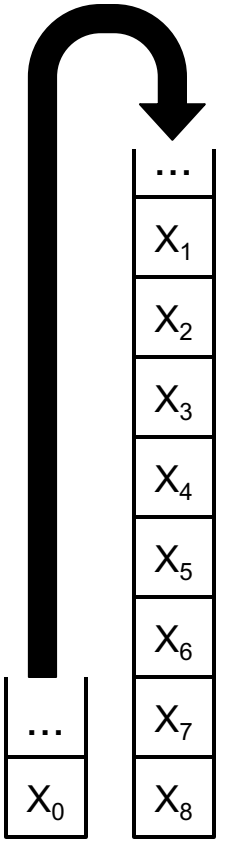




PDA com duas pilhas

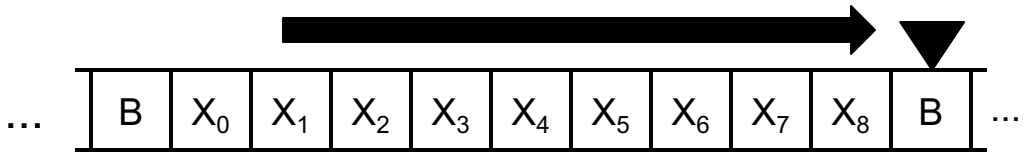
À medida que a cabeça se move para a esquerda, deslocamos os símbolos da pilha da esquerda para a pilha da direita

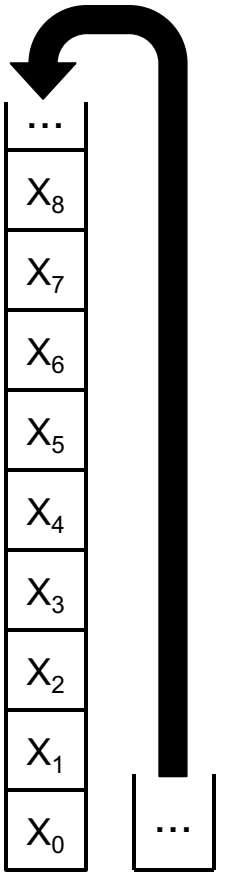




PDA com duas pilhas

À medida que a cabeça se move para direita, deslocamos os símbolos da pilha da direita para a pilha da esquerda





O fundo das pilhas representa a posição em que os infinitos brancos começam

Máquinas de Turing e computadores reais

Máquinas de Turing e computadores reais

- Agora iremos comparar a máquina de Turing e computadores reais, que usamos no dia-a-dia
  - Iremos mostrar que reconhecem as mesmas linguagens: as linguagens recursivamente enumeráveis
- Ou seja:
  - Um computador pode simular uma máquina de Turing
  - Uma máquina de Turing pode simular um computador
    - E mais: pode fazê-lo em um tempo polinomial em relação ao tempo de execução do computador
    - * Lembrando que a noção de tempo = movimentos

Simulação de uma MT por um computador

- Deve ser bastante óbvio que é relativamente fácil implementar uma MT usando uma linguagem de programação:
  - Uma variável armazena estados
  - Tabela de transições para os movimentos
  - A fita pode ser armazenada em um array
  - Um ponteiro aponta o local da cabeça de leitura
  - Existem, de fato, muitos simuladores de MT disponíveis, o que demonstra essa possibilidade

Simulação de uma MT por um computador

- Mas existe um aspecto no qual a MT “ganha”
  - Sua fita é infinita!
  - A memória do computador (memória principal, disco rígido, pen drive, etc) é finita!!
- Alguém pode argumentar que é sempre possível implementar um sistema de troca de discos em tempo de execução
  - Sempre que for necessário um novo pedaço de fita, troque o disco mais distante da cabeça por um disco novo
  - Ou seja, assumimos que conseguimos fabricar quantos discos o computador necessita
  - Mas aí estamos assumindo que a quantidade de matéria no Universo é infinita
    - E isso é outra discussão...
- Na prática, podemos confiar que a quantidade de memória não é infinita, mas é suficientemente grande para armazenar dados sobre todos os possíveis problemas que iremos encontrar

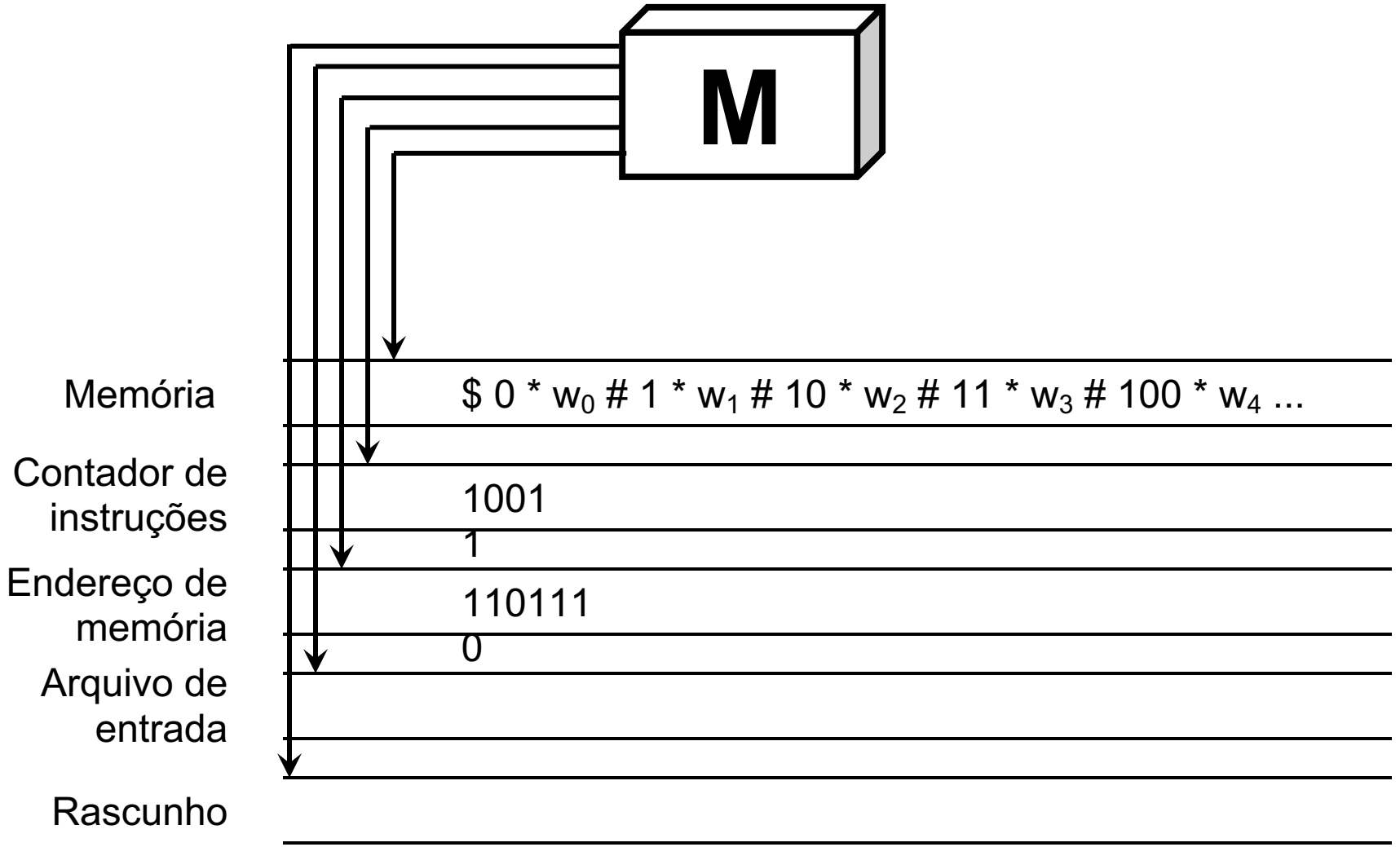
Simulação de um computador por uma MT

- A demonstração anterior não é tão surpreendente
- Mais interessante é o fato de que uma MT consegue fazer tudo que um computador faz (mais lentamente, claro)
- Iremos demonstrar como é possível simular a execução de um programa de computador em uma MT
- Para isso, vamos primeiro definir um modelo realista de um computador típico

Modelo de um computador

- Memória
  - Sequência indefinidamente longa de palavras
  - Cada uma com um endereço
- Programa
  - Está armazenado em algumas das palavras
  - Cada palavra representa uma instrução simples
    - Como linguagem de máquina, por exemplo
      - Movimentam uma palavra para outra
      - Adicionam uma palavra a outra
      - Endereçamento indireto (ponteiros)
  - Cada instrução envolve um número limitado (finito) de palavras
  - Cada instrução altera o valor de no máximo uma palavra

MT que simula um computador



MT que simula um computador

- A cada instrução:
  - A MT procura na fita 1 o local apontado pelo conteúdo da fita 2
  - A MT interpreta o conteúdo da instrução
  - Se a instrução exigir o valor de um endereço (ponteiro), a MT o copia para a fita 3. Em seguida, ela busca na fita 1 o endereço apontado pela fita 3, e copia seu conteúdo na fita 3
  - A MT então executa a instrução: cópia, soma, “salto”, etc, usando o rascunho opcionalmente como auxílio
  - A MT então incrementa o conteúdo da fita 2 e recomeça o ciclo

Tempo de simulação

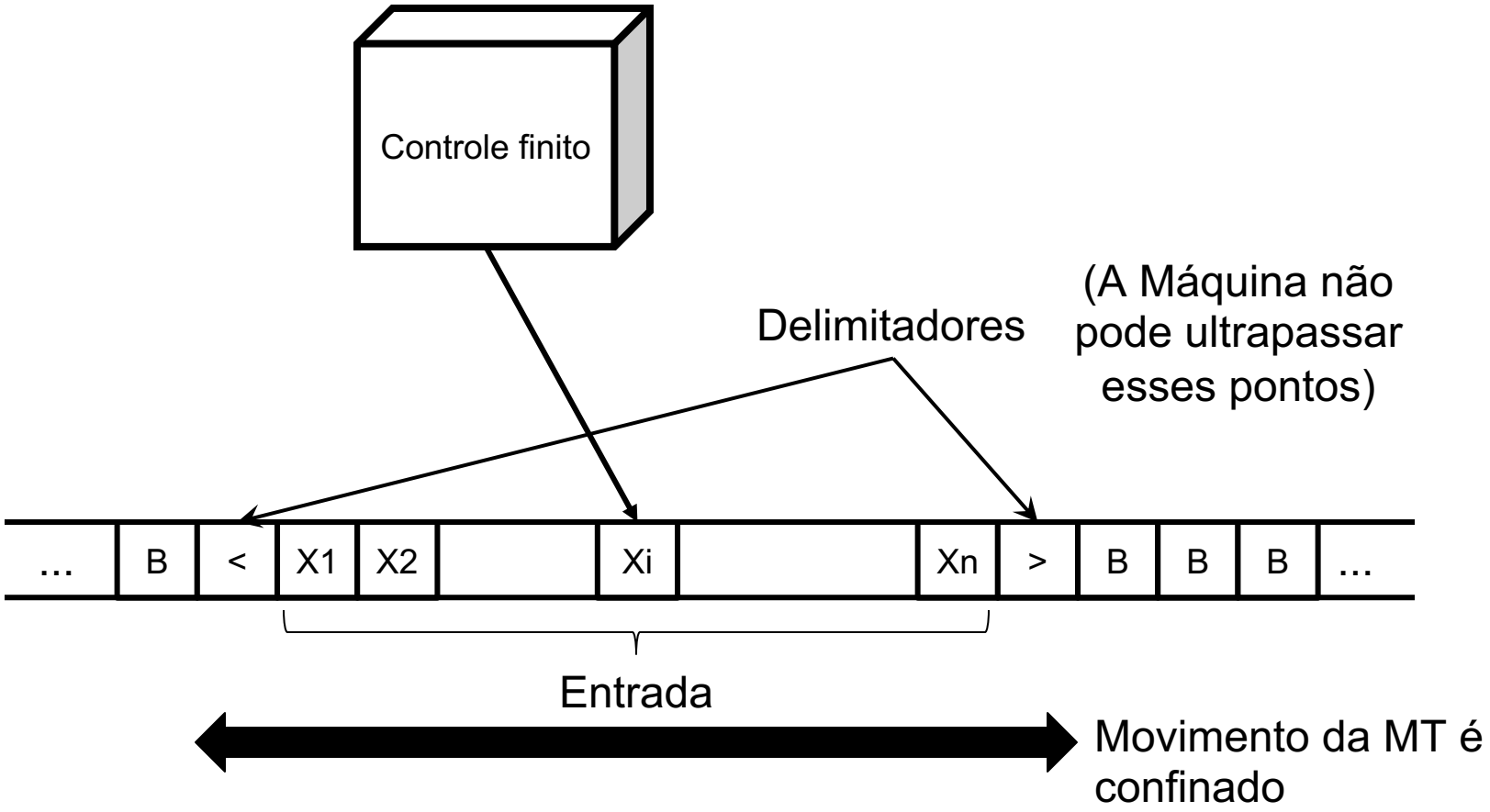
- Estamos em busca de um modelo matemático para analisar a capacidade de um computador em resolver problemas
  - Encontramos um: a MT, que tem a mesma capacidade de reconhecimento que um computador comum
  - Mas ainda falta analisar se ela é eficiente
    - Ou seja, ela executa em tempo “similar”?
    - Ou seja (2), se eu estudar o comportamento e eficiência das MTs para um determinado problema, os estudos serão também válidos para os computadores reais?
- A linha divisória é polinomial X não-polinomial!
  - Polinomial = tratável
  - Não-polinomial (exponencial, por exemplo) = intratável

Tempo de simulação

- Primeiro. A MT que simula um computador tem várias fitas, ou seja, iremos no mínimo “gastar” um tempo quadrático para converter para uma fita
  - Conforme vimos anteriormente
  - Quadrático = polinômio, então até agora, OK!
- Uma instrução do computador leva no máximo $O(n^2)$ movimentos da MT
  - Incluindo a busca de endereços, execução de instruções, etc
- Portanto: n instruções do computador leva no máximo $O(n^3)$ movimentos da MT
- Combinado à conversão para uma única fita, uma MT de uma fita pode simular n etapas de um computador em no máximo $O(n^6)$ movimentos

Máquinas de Turing com Fita Limitada

Máquinas de Turing com Fita Limitada



Máquinas de Turing com Fita Limitada

- Funcionamento é igual à MT com fita infinita
- Porém existem dois delimitadores, um à esquerda e um à direita
  - Imediatamente ao redor da entrada
- Ou seja, se a entrada tem n símbolos
  - A fita terá n+2 posições
- A MT pode ler/escrever na fita
  - Mas não pode se mover além dos delimitadores

A linguagem de uma MT com fita limitada

- O conjunto de linguagens aceitas pelas máquinas de Turing com fita limitada é chamado de
  - Linguagens sensíveis ao contexto
- Na hierarquia de Chomsky

Hierarquia	Gramáticas	Linguagens	Autômato mínimo
Tipo-0	Recursivamente Enumeráveis	Recursivamente Enumeráveis	Máquinas de Turing
Tipo-1	**Sensíveis ao contexto**	**Sensíveis ao contexto**	**MT com fita limitada**
Tipo-2	Livres de contexto	Livres de contexto	Autômatos de pilha
Tipo-3	Regulares (Expressões regulares)	Regulares	Autômatos finitos

Fim