Aspectos Formais da Computação

Prof. Sergio D Zorzo
Departamento de Computação - UFSCar
1º semestre / 2017
Aula 14

Máquina de Turing

Hierarquia de Chomsky

Hierarquia	Gramáticas	Linguagens	Autômato mínimo
Tipo-0	?	?	?
Tipo-1	?	?	?
Tipo-2	Livres de contexto	Livres de contexto	Autômatos de pilha
Tipo-3	Regulares (Expressões regulares)	Regulares	Autômatos finitos

Hierarquia de Chomsky

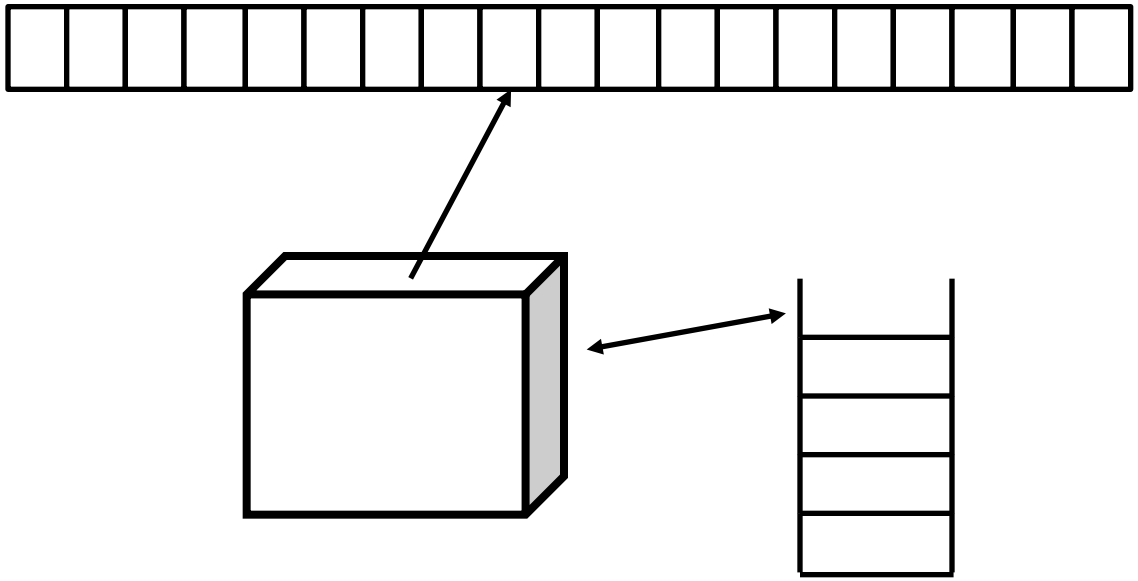
- Tipo-3
  - Problemas mais simples (analisar protocolos, buscas textuais, análise léxica)
- Tipo-2
  - Problemas “médios” (analisar programas, linguagens)
- Tipo-0
  - Problemas que podem ser resolvidos por um dispositivo computacional qualquer

Hierarquia de Chomsky

- Tipo-0:
  - Linguagem
  - Gramática
  - Autômato
- Se esse autômato não conseguir resolver um determinado problema
  - (Lembram da definição de problema? Decidir se uma cadeia w pertence a uma linguagem L)
  - Nenhum computador irá conseguir!!

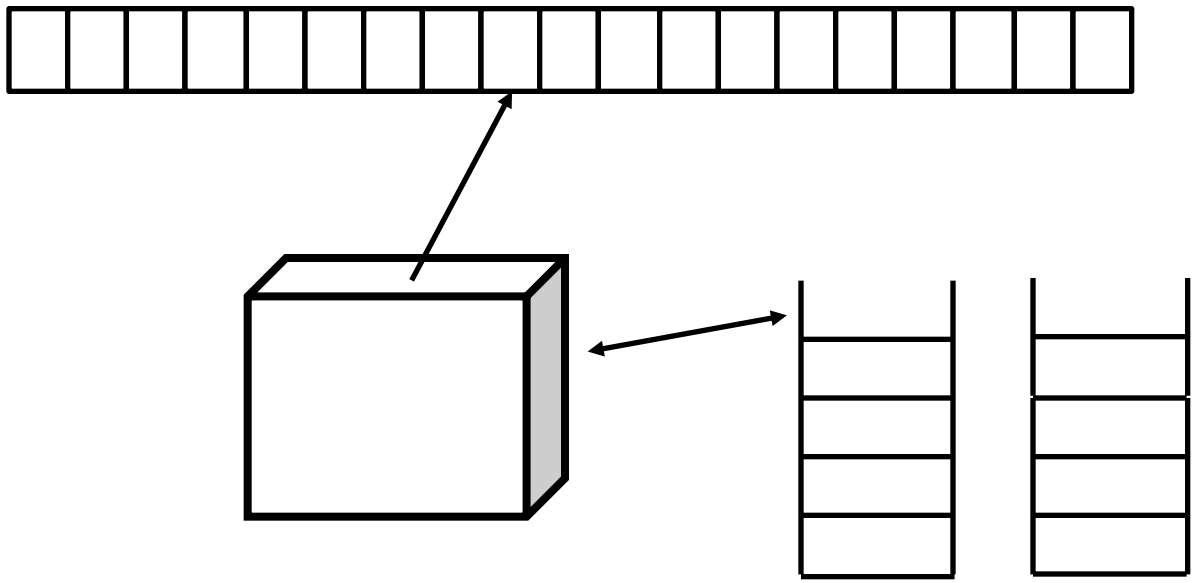
Conclusão

- Esse autômato deve ser muito poderoso mesmo!
- E se eu disser que basta pegar um autômato de pilha....



Conclusão

- ... e adicionar mais uma pilha???
- Você acredita que é possível resolver todo e qualquer problema computacional com esse autômato?



Conclusão

- Bem-vindo ao estudo dos limites da computação...

- Ou: a tese de Church-Turing

A tese de Church-Turing

Um pouco de história

- Século III
  - Diofanto de Alexandria
  - Alguns o consideram como o “pai da álgebra”
  - Não resolvia problemas com um método geral
  - Arithmetica
- Equações diofantinas
  - equações polinomiais indeterminadas cujas variáveis só podem assumir valores inteiros
- Século XVII
  - Último teorema de Pierre de Fermat
  - A equação $a^n+b^n=c^n$ não possui solução para $n > 2$
    - Obs: só foi provado na década de 1990, mais de 3 séculos depois

Os problemas de Hilbert

- 1900:
  - O matemático David Hilbert identificou 23 problemas matemáticos
  - Desafios para o século vindouro
    - Décimo problema: Encontrar um processo com o qual é possível determinar se uma equação diofantina tem uma raiz inteira, usando um número finito de operações
- Hilbert pedia que um algoritmo fosse concebido
  - Aparentemente, ele assumiu que tal algoritmo tinha de existir – alguém só precisava encontrá-lo

Definição de algoritmo

- Você sabe o que é algoritmo?
  - Uma sequência de instruções simples para realizar alguma tarefa
  - Procedimento
  - Receita
- Essa noção é intuitiva
  - Não serve para a matemática
  - Mais precisamente, não serve para responder à pergunta:
    - “Existe um algoritmo para um dado problema?”

Definição de algoritmo

- 1936
  - Alonzo Church usou um sistema notacional denominado λ-cálculo para definir algoritmos
  - Alan Turing usou "máquinas" abstratas – autômatos – para definir algoritmos
- Conexão entre noção informal de algoritmo e definição precisa
  - ***Tese de Church-Turing***
- 1970
  - Yuri Matijasevic mostrou que não existe algoritmo para determinar se uma equação diofantina tem solução

Na nossa terminologia

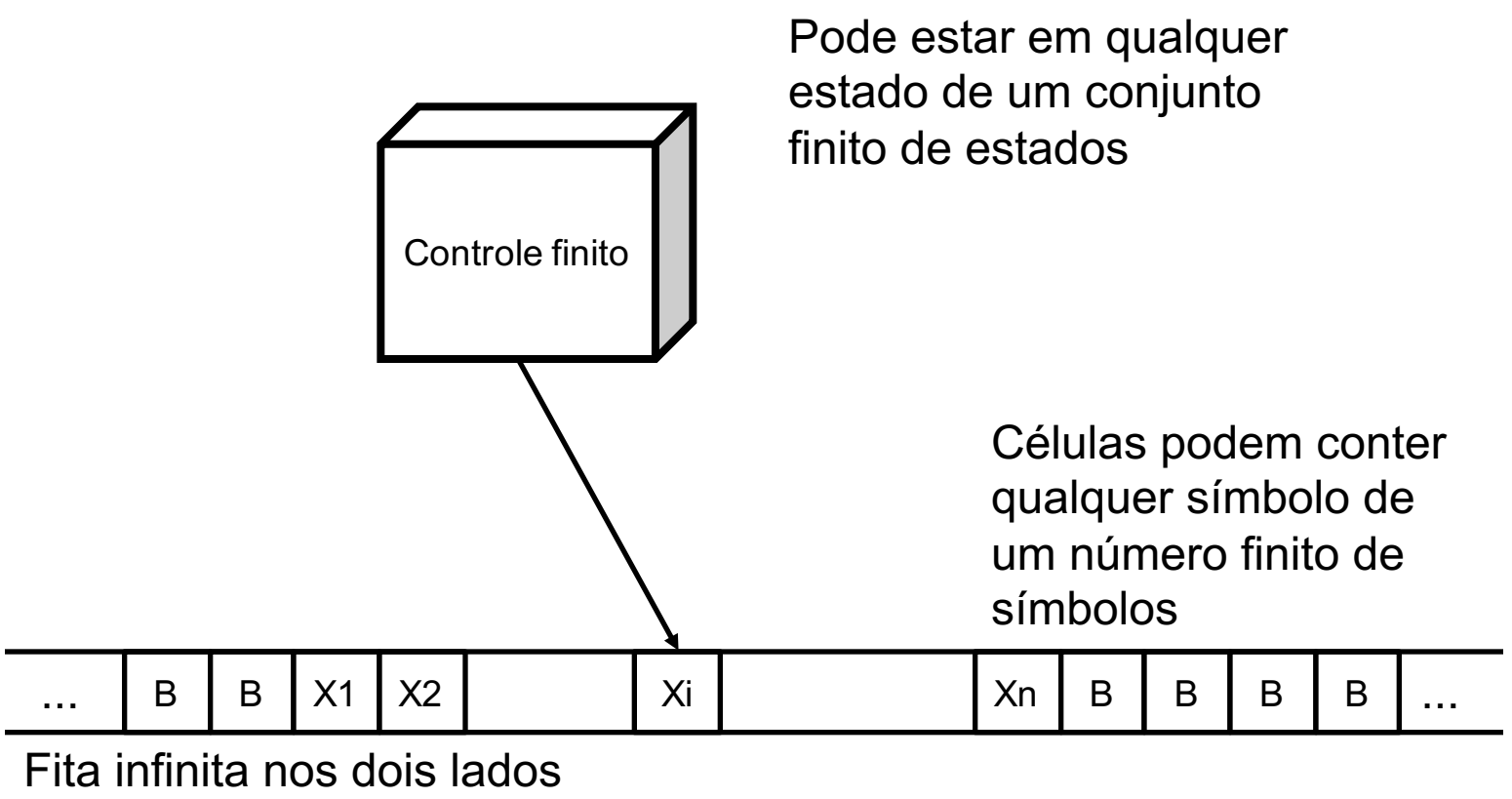
- D = {p | p é um polinômio com uma raiz inteira}

- Turing diz que:
  - Se conseguirmos projetar um autômato decisor para esta linguagem
  - Então D é decidível
  - Então existe um algoritmo para decidir (resolver) D

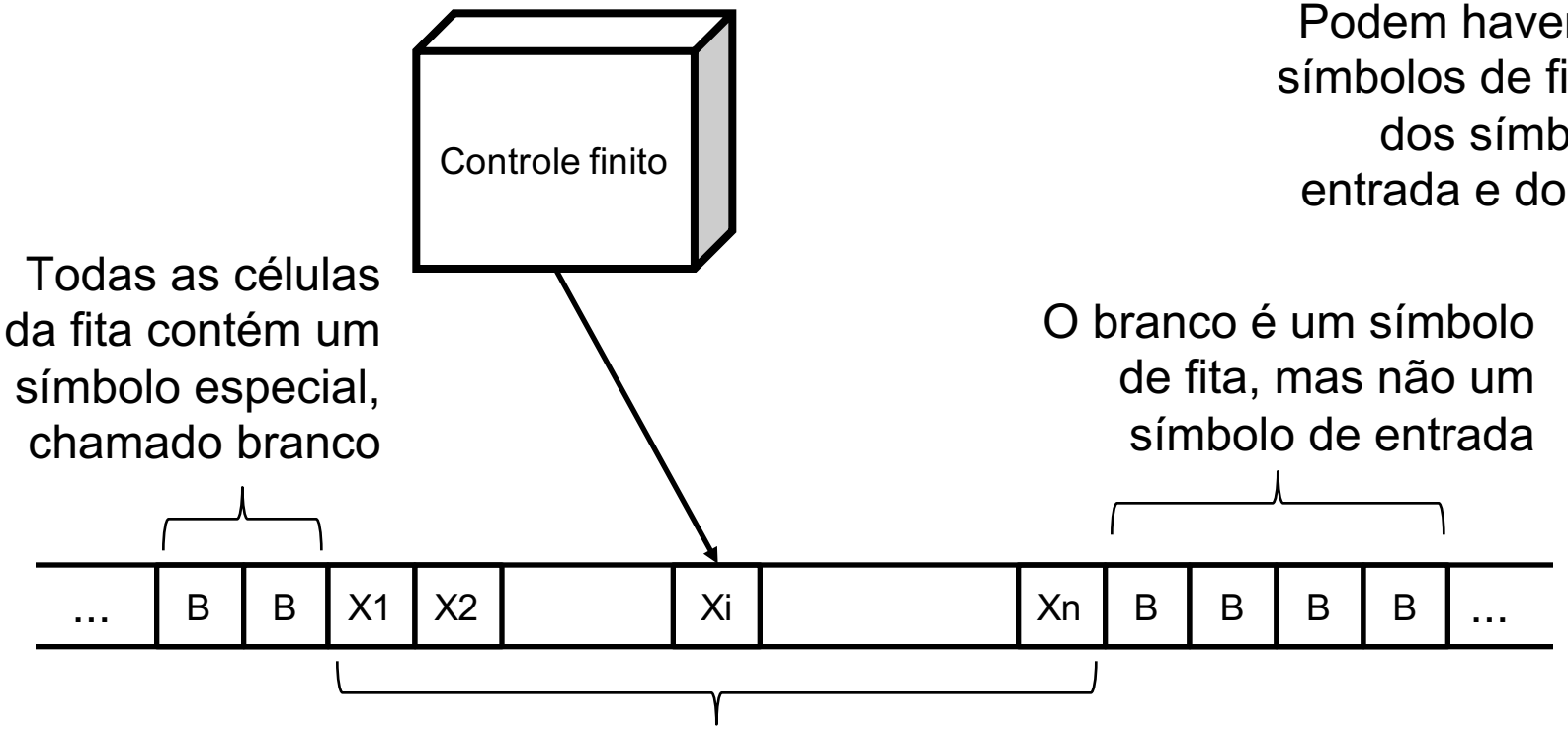
- Nesta aula, vamos estudar o formalismo de Turing – as “máquinas” de Turing
  - Se tiver curiosidade matemática, procure saber mais sobre λ-cálculo – as definições são equivalentes

A máquina de Turing

Uma notação para a máquina de Turing

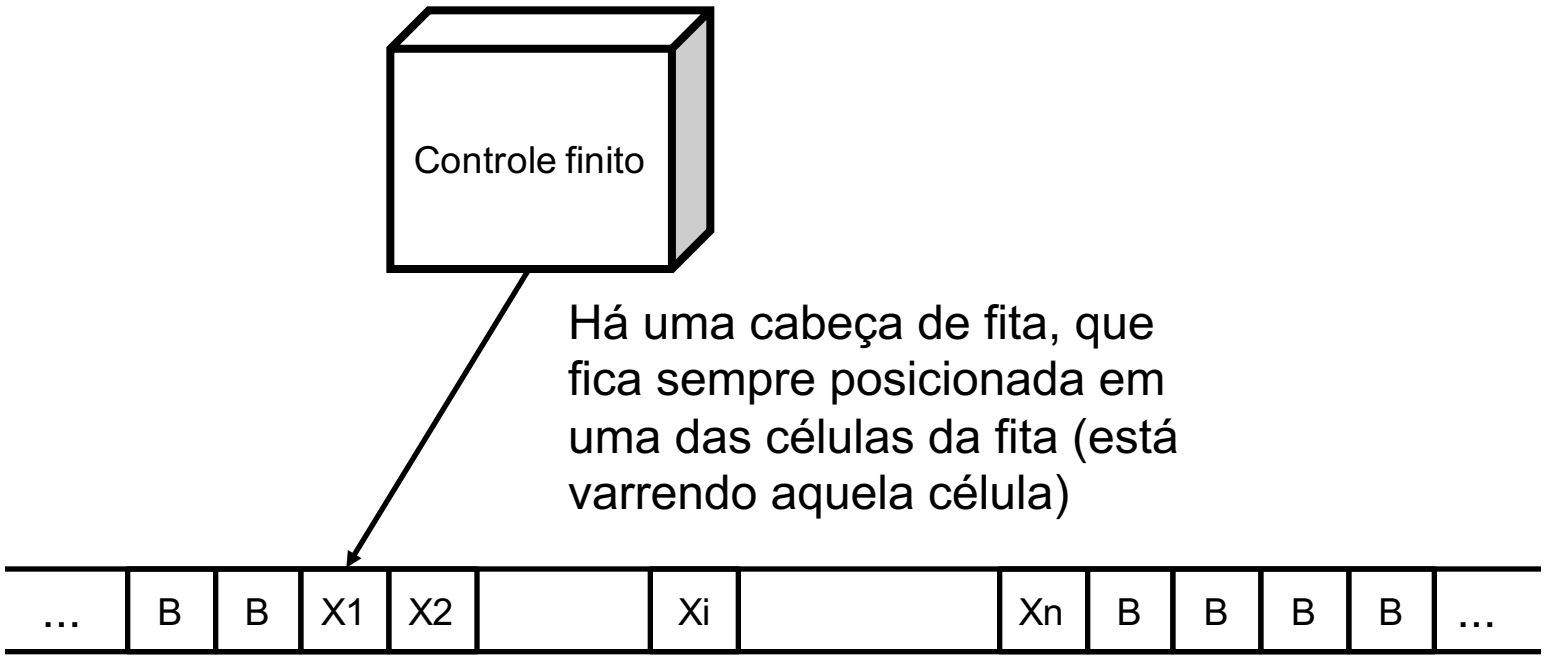


Uma notação para a máquina de Turing



Inicialmente, a entrada (cadeia finita de símbolos escolhidos do alfabeto de entrada) é colocada na fita, em qualquer lugar, substituindo os brancos

Uma notação para a máquina de Turing



Inicialmente, a cabeça encontra-se na célula mais à esquerda que contém a entrada

Uma notação para a máquina de Turing

- Um movimento da máquina de Turing é uma função:
  - Do estado do controle finito
  - Do símbolo de fita varrido
- Em um movimento, a máquina de Turing
  - Mudará de estado
  - Gravará um símbolo de fita na célula varrida, substituindo qualquer símbolo que estava nessa célula
  - Movimentará a cabeça da fita para a esquerda ou para a direita
    - Estritamente, não pode ficar parada
    - Mas é possível simular um movimento estacionário, movendo para a direita e para a esquerda em seguida, sem alterar o estado

Definição formal

- M = (Q,Σ,Γ,δ,q0,B,F), onde:
  - Q = conjunto finito de estados
  - Σ = conjunto finito de símbolos de entrada
  - Γ = conjunto completo de símbolos de fita
    - Σ é sempre um subconjunto de Γ
  - q0 = estado inicial
  - B = o símbolo branco
    - Este símbolo está em Γ mas não pode estar em Σ.
    - O branco aparece inicialmente em todas as células da fita, exceto nas células que contém a entrada
  - F = conjunto de estados finais ou de aceitação

Definição formal

- Função de transição
  - δ: Q x $\Gamma \rightarrow$ Q x Γ x {E,D}
  - Os argumentos de δ(q,X) são:
    - um estado q e um símbolo de fita X
  - O valor de δ(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

Descrição instantânea

- Uma cadeia que mostra um determinado momento da execução de uma máquina de Turing
- Uma sequência de descrições instantâneas permite descrever formalmente o que uma MT faz
  - E nos permite visualizar passo-a-passo sua execução

Descrição instantânea

- $X_1X_2...X_{i-1}qX_iX_{i+1}...X_n$
- Onde:
  - q é o estado atual da máquina de Turing
  - A cabeça da fita está varrendo o i-ésimo símbolo a partir da esquerda
  - $X_1X_2...X_{i-1}qX_iX_{i+1}...X_n$ é a parte da fita entre o não-branco mais à esquerda e o não-branco mais à direita
    - Ou seja, não mostramos os (infinitos) brancos à esquerda e à direita
    - Mas se a cabeça estiver nos extremos (X_i ou X_n), mostramos um branco para deixar claro que a partir desse momento começam os infinitos brancos

Exemplo

- $\{0^n1^n \mid n \geq 1\}$
- $M = (\{q0,q1,q2,q3,q4\},\{0,1\},\{0,1,X,Y,B\},\delta,q0,B,\{q4\})$

δ	**Símbolo**				
Estado	**0**	**1**	**X**	**Y**	**B**
q0	(q1,X,R)	-	-	(q3,Y,R)	-
q1	(q1,0,R)	(q2,Y,L)	-	(q1,Y,R)	-
q2	(q2,0,L)	-	(q0,X,R)	(q2,Y,L)	-
q3	-	-	-	(q3,Y,R)	(q4,B,R)
q4	-	-	-	-	-

Exercício

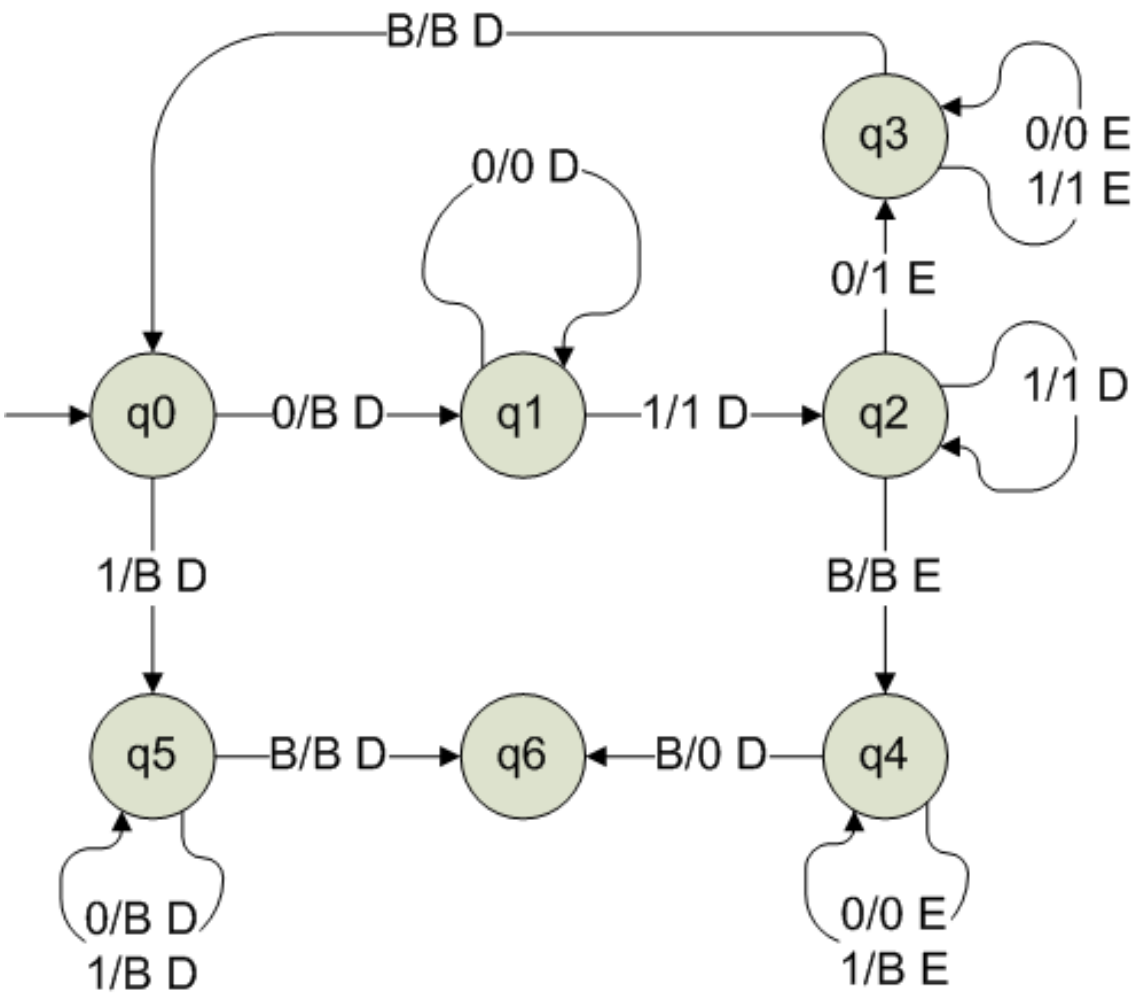
- Teste as cadeias 000111 e 0010

Notação de diagrama

- É também possível representar a Máquina de Turing na forma de diagrama
  - Nós = estados
  - Arcos = transições
  - Transições = X/Y S, onde
    - $\delta(q,X) = (p,Y,S)$
    - A seta leva de q até p
    - X e Y são símbolos de fita
    - S é um sentido, L ou R, E ou D,← ou →

Exemplo

- M = ({q0,q1,q2,q3,q4,q5,q6}, {0,1},{0,1,B},δ,q0,B)
  - Obs: essa máquina não tem estados de aceitação
  - Pois não é utilizada para aceitar entradas



Exercício

- Teste as cadeias 0010, 0000100 e 01000
- Resp: calcula a operação monus ou subtração própria
  - Max(m-n,0)
- Entrada: 0^m10^n
- Na fita irá restar: 0^x
  - Onde x = m monus n

A linguagem de uma máquina de Turing

- A cadeia de entrada é colocada na fita
- A cabeça da fita começa no símbolo de entrada mais à esquerda
- Se a MT entrar eventualmente em um estado de aceiação
  - A entrada será aceita
  - Caso contrário, não será aceita

A linguagem de uma máquina de Turing

- Formalmente

- Seja M uma máquina de Turing
  - $M = (Q,\Sigma,\Gamma,\delta,q0,B,F)$
- Então L(M) é o conjunto de cadeias w em Σ^* tais que
  - $q0w \vdash^* \alpha p \beta$
  - Para algum estado p em F e quaisquer cadeias de fita α e β

A linguagem de uma máquina de Turing

- O conjunto de linguagens aceitas pelas máquinas de Turing é chamado de
  - Linguagens recursivamente enumeráveis
  - Linguagens Turing-reconhecíveis
- Na hierarquia de Chomsky

Hierarquia	Gramáticas	Linguagens	Autômato mínimo
Tipo-0	**Recursivamente Enumeráveis**	**Recursivamente Enumeráveis**	**Máquinas de Turing**
Tipo-1	?	?	?
Tipo-2	Livres de contexto	Livres de contexto	Autômatos de pilha
Tipo-3	Regulares (Expressões regulares)	Regulares	Autômatos finitos

Máquinas de Turing e sua parada

- Existem três resultados possíveis para a execução de uma máquina de Turing em uma dada entrada
  - Não há mais movimentos possíveis, e o último estado é de aceitação
  - Não há mais movimentos possíveis, e o último estado não é de aceitação
  - A máquina entra em loop infinito
    - Comportamento simples ou complexo que nunca leva a máquina a parar

Máquinas de Turing e sua parada

- Máquinas de Turing que sempre param são interessantes
  - Ou seja, em TODA entrada possível ela é capaz de decidir se aceita ou não, sem entrar em loop
- São chamadas de decisores, pois sempre tomam uma decisão
  - Em contraste, por exemplo, que máquinas que param quando aceitam, mas podem entrar em loop em algumas cadeias que não aceitam
- Linguagem aceitas por decisores:
  - Linguagens recursivas
  - Linguagens Turing-decidíveis
  - Linguagens decidíveis

Máquinas de Turing e sua parada

- É muito simples projetar estados de aceitação ou não-aceitação que forçam a MT a parar
- Basta criar um estado q que não possui nenhuma transição a partir dele
  - Se q pertence a F, assim que a máquina entrar em T, a cadeia é aceita
  - Se q não pertence a F, assim que a máquina entrar em T, a cadeia é rejeitada
- Dessa forma, facilita-se o projeto de decisores (máquinas que eventualmente param, aceitando ou não)

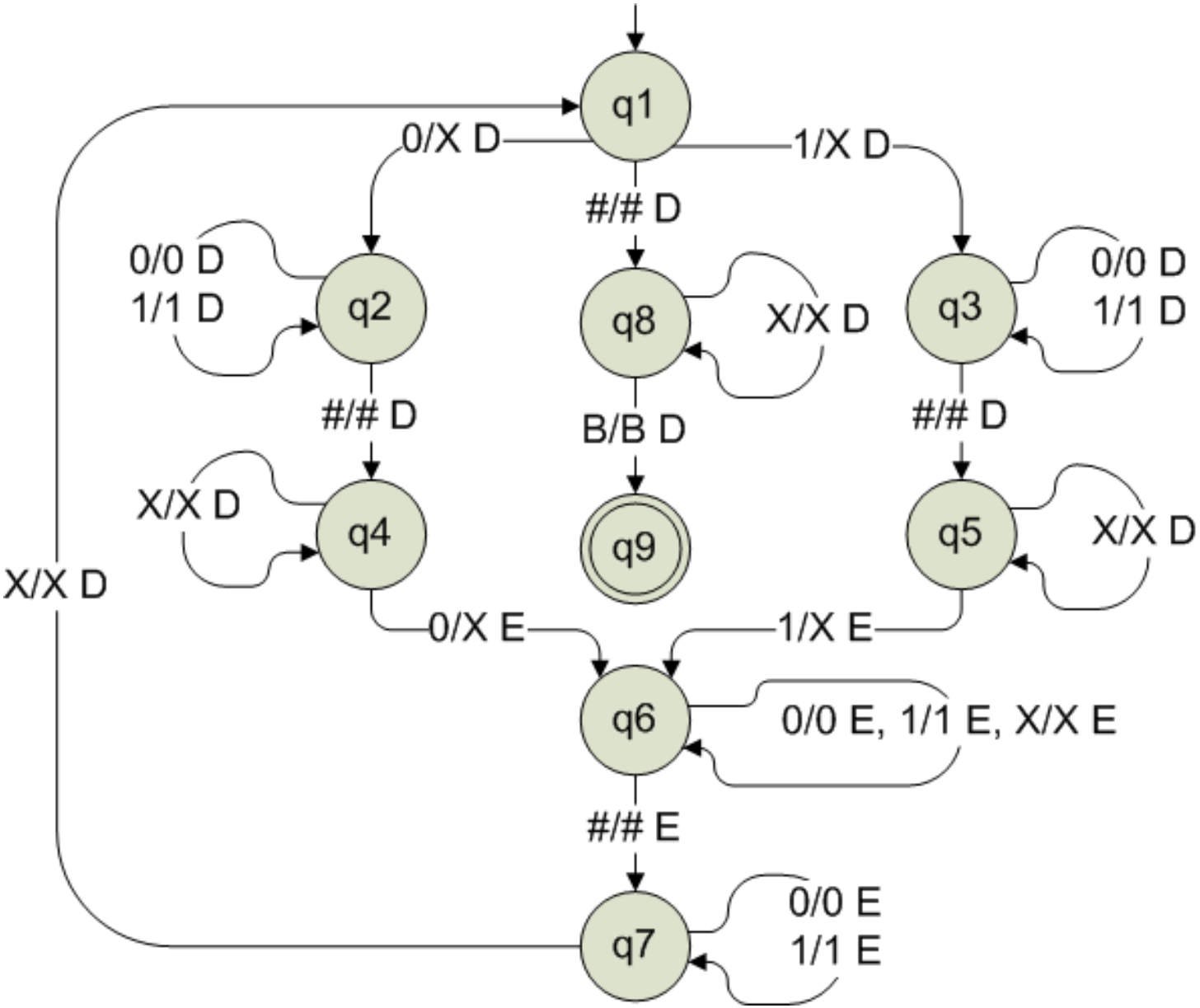
Projetando Máquinas de Turing

- Vamos projetar uma máquina de Turing para a linguagem B = {w#w | w está em {0,1}*}
- Imagine-se numa estrada com 1 km de comprimento
- Na estrada existe uma cadeia de 0s e 1s escrita no chão. Em algum ponto no meio dessa cadeia tem um “#” escrito
  - Conclusão óbvia: não dá para "memorizar" (em estados) toda a cadeia
  - Mas você pode riscar os 0s e 1s à vontade
  - Como você resolveria o problema de decidir se essa cadeia pertence a B?

Projetando Máquinas de Turing

- Solução:
  - Comece no primeiro símbolo e memorize-o
  - Faça um X sobre esse símbolo
  - Ande pela estrada até encontrar o meio (#)
  - A partir desse momento, encontre o primeiro símbolo não-riscado e compare com aquele memorizado no início
    - Se for diferente, você já sabe que a cadeia não pertence a B!! Pode ir pra casa descansar
    - Se for igual, marque-o com um X e volte em direção ao início da cadeia, passando pelo #, parando assim que encontrar o primeiro símbolo riscado. Recomece o processo.
  - Quando todos os símbolos à esquerda do # tiverem sido marcados, verifique a existência de algum símbolo remanescente à direita do #. Se houver, a cadeia não pertence a B.
  - Se não houver, a cadeia pertence a B!!

Projetando Máquinas de Turing



Projetando Máquinas de Turing

- É a mesma noção de construir um algoritmo
- Primeiro pensa-se nos passos
  - Considerando as restrições e os possíveis movimentos da máquina
- Depois fazemos a “tradução” para um diagrama ou tabela
  - Criamos estados para lembrar de algumas coisas
  - Outras coisas armazenamos na própria fita

Exercício

- Projete uma máquina de Turing que decide
  - $A = \{0^{2^n} \mid n \geq 0\}$
  - Todas as cadeias de 0s cujo comprimento é uma potência de 2

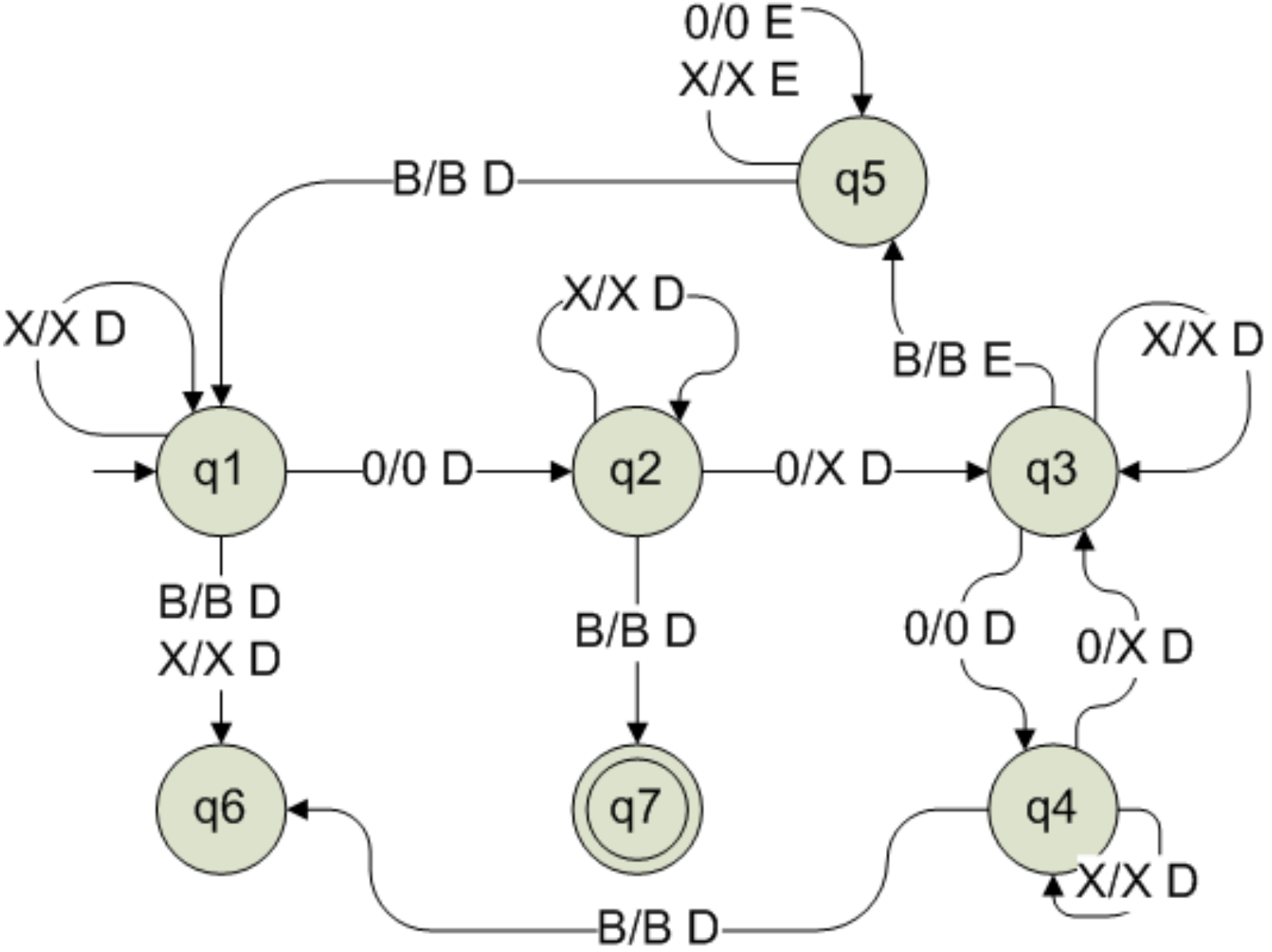
Solução (algoritmo)

1. Faça uma varredura da esquerda para a direita na fita, marcando um 0 não, e outro, sim
2. Se no estágio 1, a fita continha um único 0, aceite
3. Se no estágio 1, a fita continha mais que um único 0 e o número de 0s era ímpar, rejeite
4. Retorne a cabeça para a extremidade esquerda da fita
5. Vá para o estágio 1

Solução (algoritmo)

- Cada iteração do estágio 1 corta o número de 0s pela metade.
  - Nessa verredura da fita, a máquina mantém registro de se o número de 0s é par ou ímpar
  - Se for ímpar e maior que 1, o número de 0s não pode ser uma potência de 2 → rejeita
  - Se o número de 0s visto for 1, o número original deve ter sido uma potência de 2 → aceita

Solução (formal)

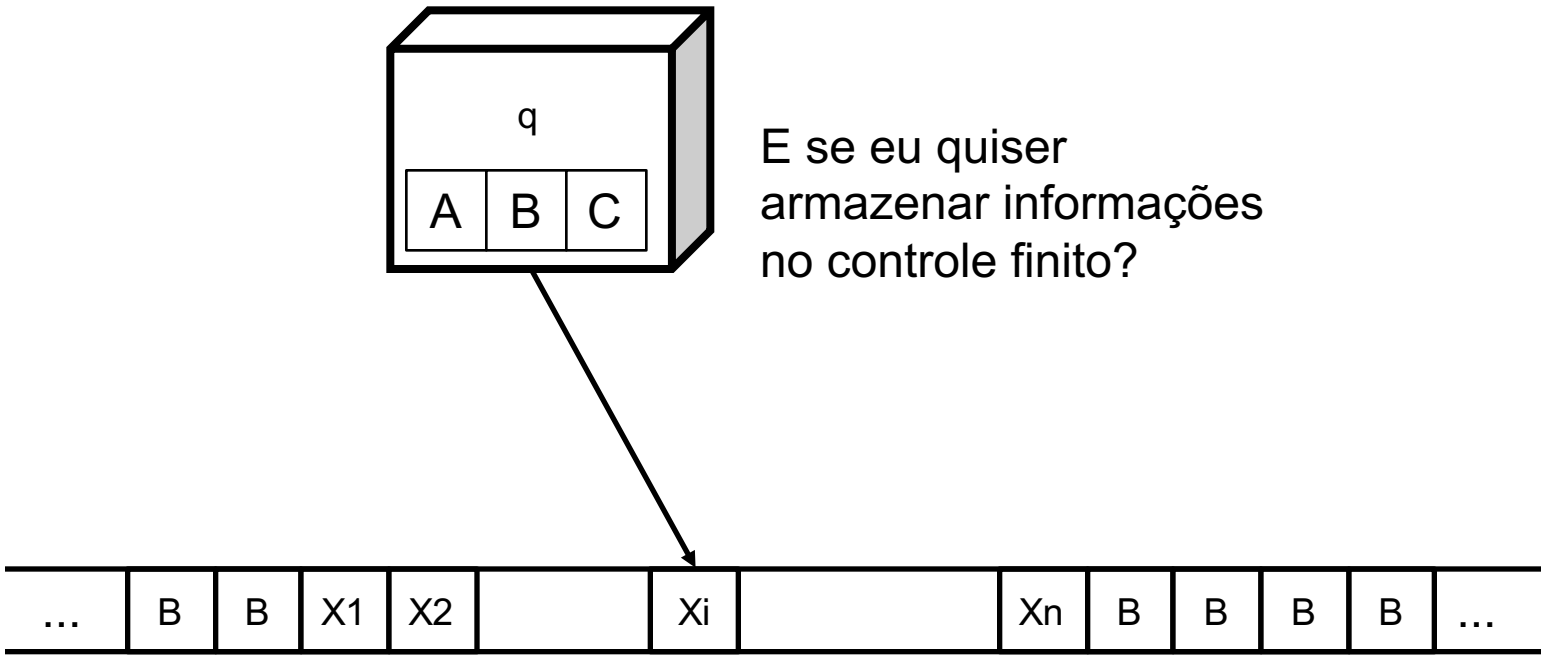


Variantes

Variantes

- Existem muitas variantes para o modelo que vimos anteriormente
  - Todas possuem o mesmo poder de reconhecimento
- Facilitam o entendimento de alguns aspectos
- Facilitam a “programação”

Armazenamento no estado



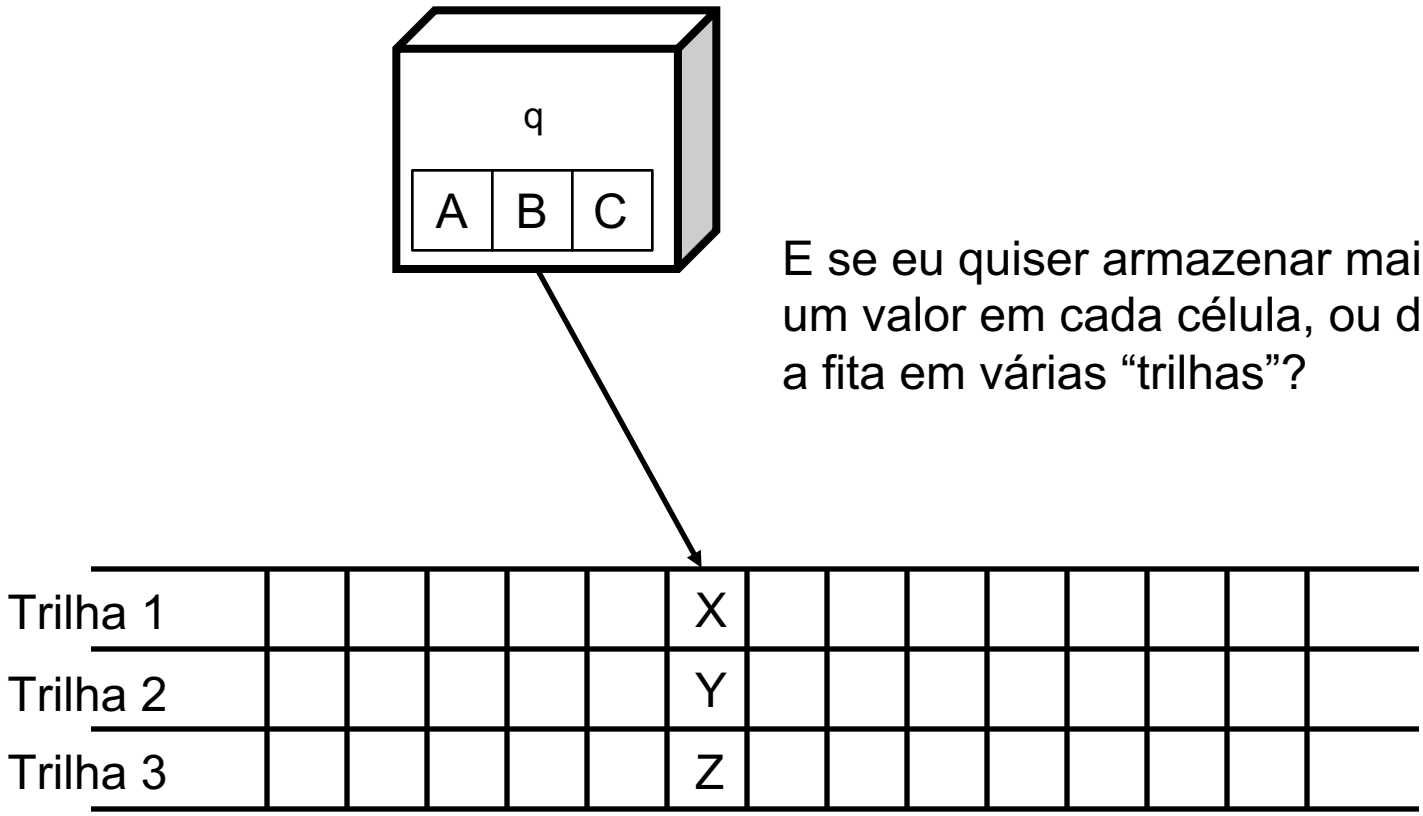
Armazenamento no estado

- Por exemplo, uma linguagem baseada em {a,b}* onde os primeiros três símbolos não se repetem ao longo da cadeia:
  - “aabbbabbbaaa” faz parte (aab não se repete)
  - “ababbbaabbbabbb” faz parte (aba não se repete)
  - “aabbbaabbbabba” não faz parte (aab aparece depois)
- Seria interessante “armazenar” as três primeiras letras e mantê-las “por perto” para poder consultar

Armazenamento no estado

- A solução é simples
- Não exige modificações no modelo da MT
  - Basta dar nomes adequados e criar quantos estados forem necessários para cobrir todas as combinações dos valores a serem armazenados
  - E também incluir os estados “normais” (q1, q2, etc)
- Ex:
  - q=q1, P1=a, P2=a, P3=a → q1aaa
  - q=q1, P1=a, P2=a, P3=b → q1aab
  - q=q4, P1=a, P2=b, P3=a → q4aba

Várias trilhas em uma única fita

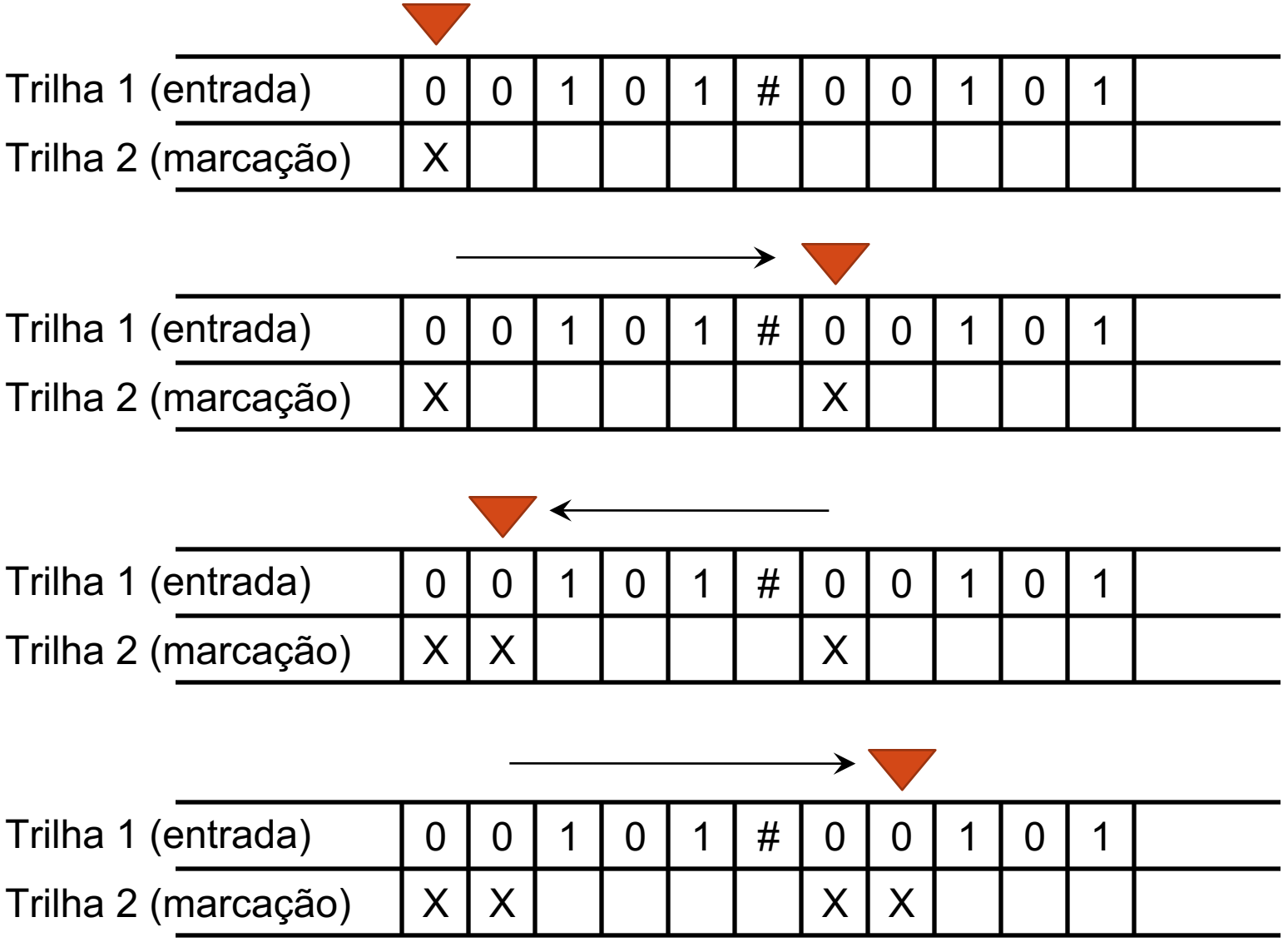


E se eu quiser armazenar mais de um valor em cada célula, ou dividir a fita em várias “trilhas”?

Várias trilhas em uma única fita

- Por exemplo, considere a seguinte linguagem:
  - {w#w | w em {0,1}*}
  - Anteriormente, projetamos uma MT que ziguezagueia pela entrada, “riscando” símbolos e substituindo-os por um x
  - Cada x representa um símbolo que já foi verificado
  - Quando toda a cadeia estiver somente com “x”s, ela é aceita
- Mas...
  - E se eu precisar de algum comportamento adicional?
  - E se eu precisar lembrar quais eram os símbolos que estavam lá?

Várias trilhas em uma única fita

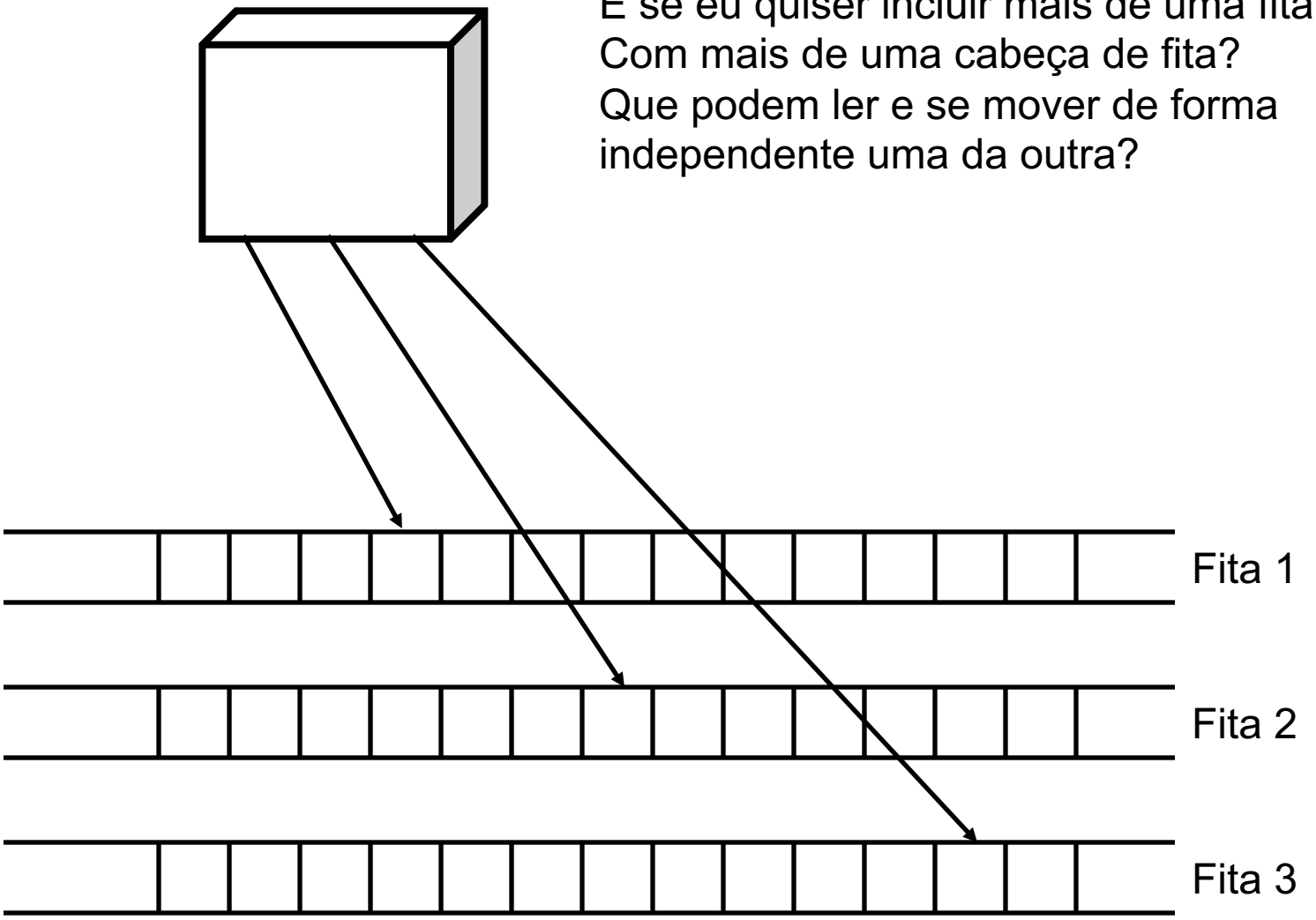


Várias trilhas em uma única fita

- O “truque” é fazer o mesmo que o armazenamento no estado
  - Ou seja, cada símbolo de fita, na verdade, é uma tupla que representa uma combinação diferente de símbolos, um para cada trilha
  - Por exemplo, na MT original (com uma única trilha):
    - $\Sigma = \{0,1\}$
    - $\Gamma = \{B,0,1,X\}$
  - Utilizando duas trilhas
    - $\Sigma = \{0,1\}$
    - $\Gamma = \{B,0,1,0',1'\}$
    - Onde 0’ representa o 0 “marcado” com um X, e 1’ representa o 1 “marcado” com um X
      - Ou: $\Gamma = \{(B,B),(0,B),(1,B),(0,X),(1,X)\}$
- Nessa situação, ainda é possível ver qual é o símbolo original, mesmo sendo “marcado”
  - É inclusive possível limpar a trilha auxiliar após seu uso

MT com várias fitas

E se eu quiser incluir mais de uma fita?
Com mais de uma cabeça de fita?
Que podem ler e se mover de forma independente uma da outra?



MT com várias fitas

- A inclusão de outras fitas não aumenta o poder de definição de linguagens ao modelo básico, com uma única fita
  - Mas facilita a sua “programação”
  - Além de outras demonstrações interessantes

MT com várias fitas

- Inicialmente:
  - A entrada, uma sequência finita de símbolos de entrada, é colocada na primeira fita
  - Todas as outras células de todas as fitas contêm brancos
  - O controle finito está no estado inicial
  - A cabeça da primeira fita está na extremidade esquerda da entrada
  - Todas as outras cabeças de fitas estão em alguma célula arbitrária (na verdade, tanto faz)

MT com várias fitas

- Num movimento:
  - O controle entra em um novo estado (que pode ser o mesmo que o anterior)
  - Em cada fita, um novo símbolo de fita é escrito na célula varrida (pode ser o mesmo que o anterior)
  - Cada uma das cabeças faz um movimento
    - Para a esquerda, direita, ou estacionário (já vimos que movimento estacionário pode ser simulado por um movimento à esquerda seguido por um movimento à direita)
- Ou seja:
  - $\delta(q, a_1, a_2, \ldots, a_k) = (p, b_1, b_2, \ldots, b_k, E, D, E, \ldots, E)$

MT com várias fitas

- MTs com uma fita reconhecem as linguagens recursivamente enumeráveis
  - MTs com múltiplas fitas também reconhecem as linguagens RE (óbvio)
  - Mas elas reconhecem mais linguagens?

Hierarquia	Gramáticas	Linguagens	Autômato mínimo
Tipo-0	Recursivamente Enumeráveis	Recursivamente Enumeráveis	Máquinas de Turing
Tipo-1	?	?	?
Tipo-2	Livres de contexto	Livres de contexto	Autômatos de pilha
Tipo-3	Regulares (Expressões regulares)	Regulares	Autômatos finitos

Máquinas de Turing com múltiplas fitas

Estaria a MT com múltiplas fitas aqui em cima?

MT com várias fitas

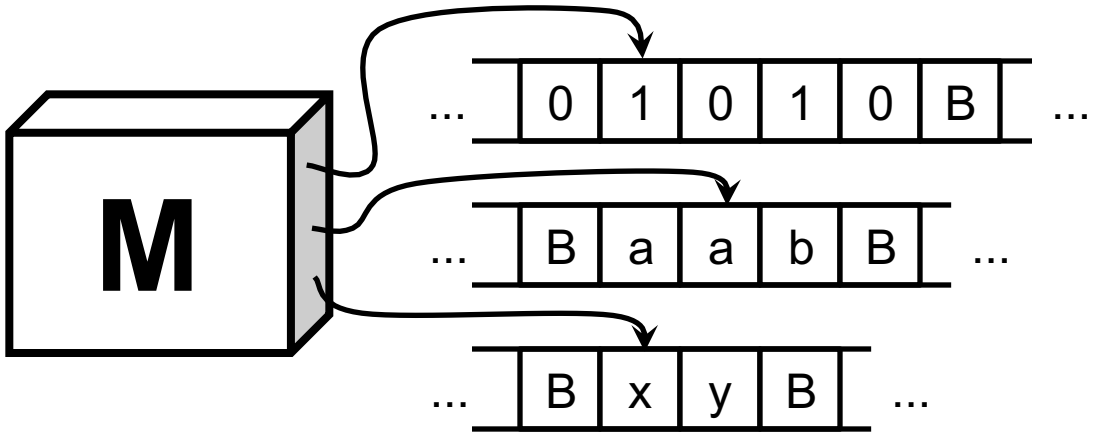
- A resposta é dada pelo seguinte teorema:

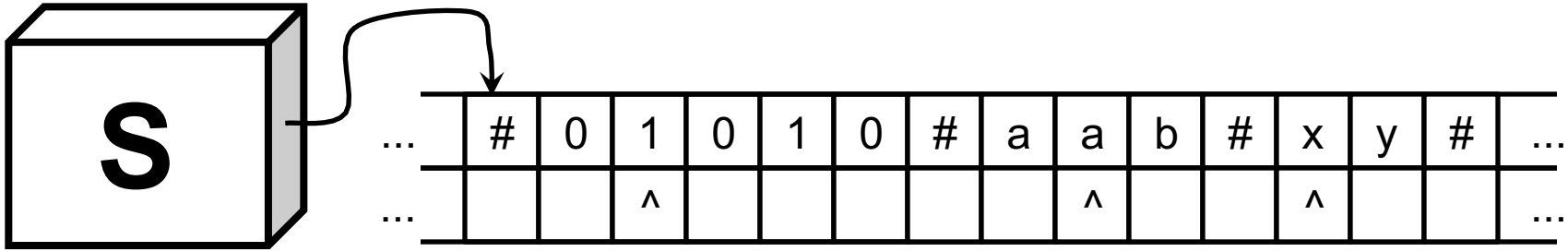
- “Toda linguagem aceita por uma MT de várias fitas é recursivamente enumerável”

- Prova: por construção = conversão de múltiplas fitas para uma única fita
  - Prova 1
  - Prova 2

MT com várias fitas

- Prova 1: conversão de M para S





MT com várias fitas

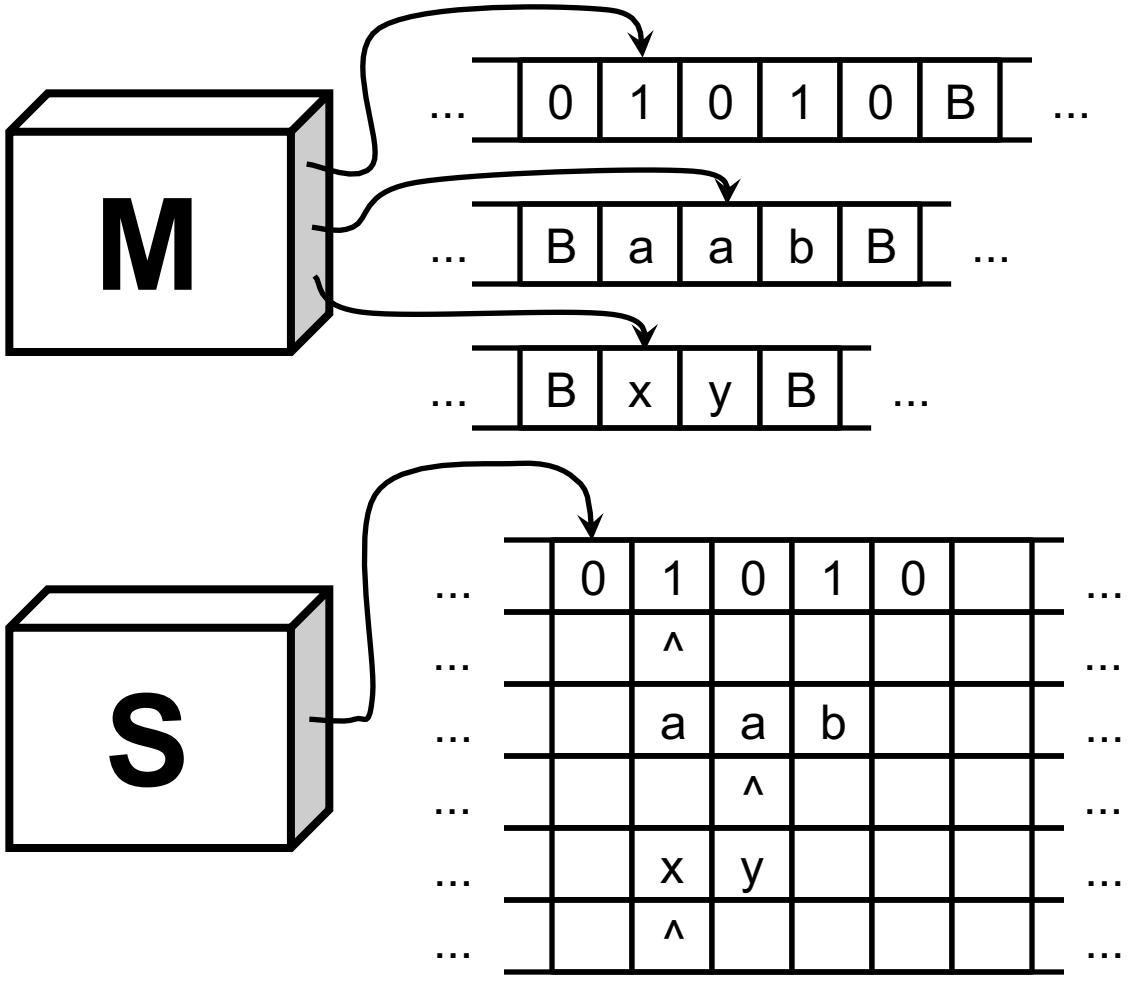
- Prova 1: conversão de M para S
  - M = múltiplas fitas
  - S = uma fita
- Inicialmente, a fita de S é formatada:
  - #w_1'w_2...w_n#B'#B'#...#
  - O conteúdo de cada fita é colocado em sequência, separados por um símbolo especial (#)
  - A primeira fita contém a cadeia de entrada, as demais contém brancos somente
  - Uma segunda trilha irá conter apenas "ponteiros" que apontam onde as cabeças estão em um determinado momento (representados por apóstrofos (') aqui)

MT com várias fitas

- Prova 1: conversão de M para S
- Em um único movimento:
  - S faz uma varredura na fita, desde o primeiro # até o último #
    - Nessa varredura, S "armazena" as posições de cada cabeça
  - S faz então uma segunda varredura, atualizando as fitas conforme a função de transição de M
- Se em algum ponto S move uma das cabeças virtuais sobre um #, significa que M teria encontrado um branco
  - S então escreve um branco sobre o # e desloca o restante da fita, símbolo a símbolo, aumentando seu comprimento, para a esquerda ou direita, conforme necessário

MT com várias fitas

- Prova 2: conversão de M para S



MT com várias fitas

- Prova 2: conversão de M para S
- Inicialmente:
  - Para cada fita em M, cria-se duas trilhas em S
    - Uma para os símbolos
    - Outra apenas para marcar a posição da cabeça de M
- Em um movimento:
  - S percorre as trilhas de marcação (armazenando quantos ela já descobriu, para não se perder)
    - E armazena os símbolos lidos na posição correspondente da trilha que representa a fita original
  - S volta ao início e percorre novamente as trilhas, atualizando os símbolos e as posições das cabeças virtuais, conforme definido em M

MT com várias fitas

- Um questionamento: por que não “armazenar” as posições das cabeças no controle finito?
  - Resposta: porque existem infintas posições (as fitas são infinitas)
  - E o armazenamento no controle finito usa um número finito de estados
  - Ou seja, se eu estiver na posição 1456633 da fita, eu teria que ter um estado q1456633
    - Mas sempre poderá existir uma posição 1456634!!
    - Ou seja, seria necessário ter infinitos estados

Uma análise importante

- MT com várias fitas e com uma fita são equivalentes:
  - em termos de capacidade reconhecedora !!
- Mas e em tempo de execução?
  - Obviamente gasta-se mais tempo executando uma MT com uma única fita!!
- De que isso importa?
  - Mais adiante usaremos MTs como uma forma de analisar tratabilidade!
  - Relacionado com tempos de execução viáveis ou inviáveis
  - Nesse contexto, a análise do tempo de execução faz toda a diferença!

Tempo de execução de uma MT

- Não é tempo real
  - Porque a MT não é uma máquina real!!
- É um tempo relativo:
  - Tempo de execução da MT M sobre entrada w é o número de movimentos que M faz antes de parar
  - Se M não parar em w, o tempo será infinito
- Isso leva ao estudo da complexidade
  - Veremos mais adiante na disciplina

Tempo de construção de muitas fitas para uma

- Aparentemente, a diferença de execução entre uma e várias fitas é grande
  - Mas não é muito!!
  - A MT de uma fita demora um tempo não maior que o quadrado do tempo tomado pela outra
  - Teorema: dada uma máquina de múltiplas fitas M que, dada uma entrada w, executa em n movimentos. Uma máquina de uma fita S equivalente a M, para a mesma entrada w, executa em $O(n^2)$ movimentos

Tempo de construção de muitas fitas para uma

- Prova: M (k fitas) convertido em S (1 fita)
- M executou n movimentos
  - Os marcadores de cabeça estarão com certeza distantes em 2n células ou menos
  - Ou seja, para cada um dos n movimentos de M:
    - S consegue, começando do marcador de cabeça mais à esquerda, em 2n movimentos no máximo, encontrar todos os marcadores nas suas respectivas trilhas
    - Em seguida, em 2n movimentos no máximo, S consegue com certeza voltar para o começo
    - Nesse caminho de volta, ele pode ter que ajustar as posições das cabeças conforme as transições de M
    - Ou seja, algumas cabeças irão para a direita, outras irão para a esquerda. Considerando k cabeças, em 2k movimentos ou menos é possível ajustar todas
    - Total até agora: $4n + 2k = O(n)$, pois k é uma constante
  - Se cada movimento de M leva O(n), n movimentos em S levará n vezes isso, ou seja: $O(n^2)$

Tempo de construção de muitas fitas para uma

- Resumindo:
  - Se em múltiplas fitas leva 10 movimentos, em uma fita levará no máximo 100
  - Se em múltiplas fitas leva 100 movimentos, em uma fita levará no máximo 10000
- Essa diferença é pequena em termos computacionais!
- Pois é uma diferença polinomial
  - $y=x^2$
- O problema está em taxas de crescimento mais altas
  - São o limite entre o que podemos resolver por computadores e o que não tem solução prática
  - Mais detalhes depois

Fim