Linguagens Formais e Autômatos

Aula 20 - Máquina de Turing

Prof. Dr. Daniel Lucrédio
Departamento de Computação / UFSCar
Última revisão: ago/2015

Referências bibliográficas

- **Introdução à teoria dos autômatos, linguagens e computação / John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman** ; tradução da 2.ed. original de Vandenberg D. de Souza. - Rio de Janeiro : Elsevier, 2002 (Tradução de: Introduction to automata theory, languages, and computation - ISBN 85-352-1072-5)
  - Capítulo 8 - Seções 8.1 e 8.2
- **Introdução à teoria da computação / Michael Sipser** ; tradução técnica Ruy José Guerra Barretto de Queiroz ; revisão técnica Newton José Vieira. -- São Paulo : Thomson Learning, 2007 (Título original : Introduction to the theory of computation. "Tradução da segunda edição norte-americana" - ISBN 978-85-221-0499-4)
  - Capítulo 3 - Seção 3.1

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

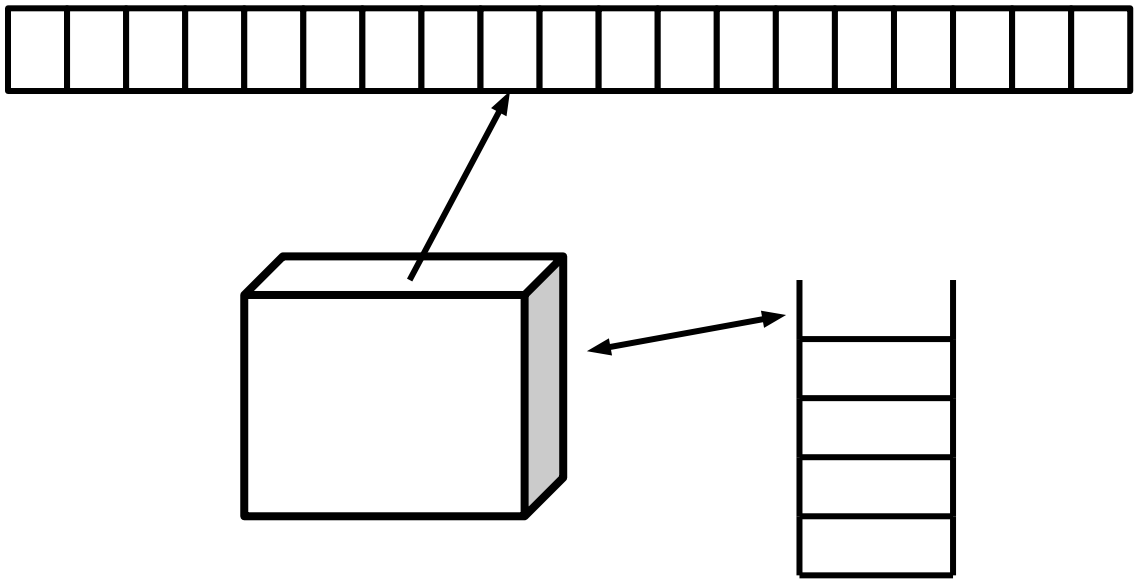
- Tipo-0:
  - Linguagem
  - Gramática
  - Autômato
- Se esse autômato não conseguir resolver um determinado problema
  - (Lembram da definição de problema? Decidir se uma cadeia w pertence a uma linguagem L)
  - Nenhum computador irá conseguir!!

Capacidade de computação

- Nenhum computador irá conseguir?
  - Isso! Nenhum!
- Nem o Jaguar-XT5? Do Oak Ridge National Laboratory? Com 224.256 núcleos de processamento Opteron? Capacidade para 1.75 petaflops?
  - Não
- Nem o Kraken? Do National Institute for Computational Sciences da Universidade do Tenessee? 129 terabytes de memória? 16.512 entradas para computadores?
  - Não
- Nem meu bom e rápido PC – AT 286 com 20 Megahertz?
  - Errr..... Mmmmm ... Nnnnnnão!

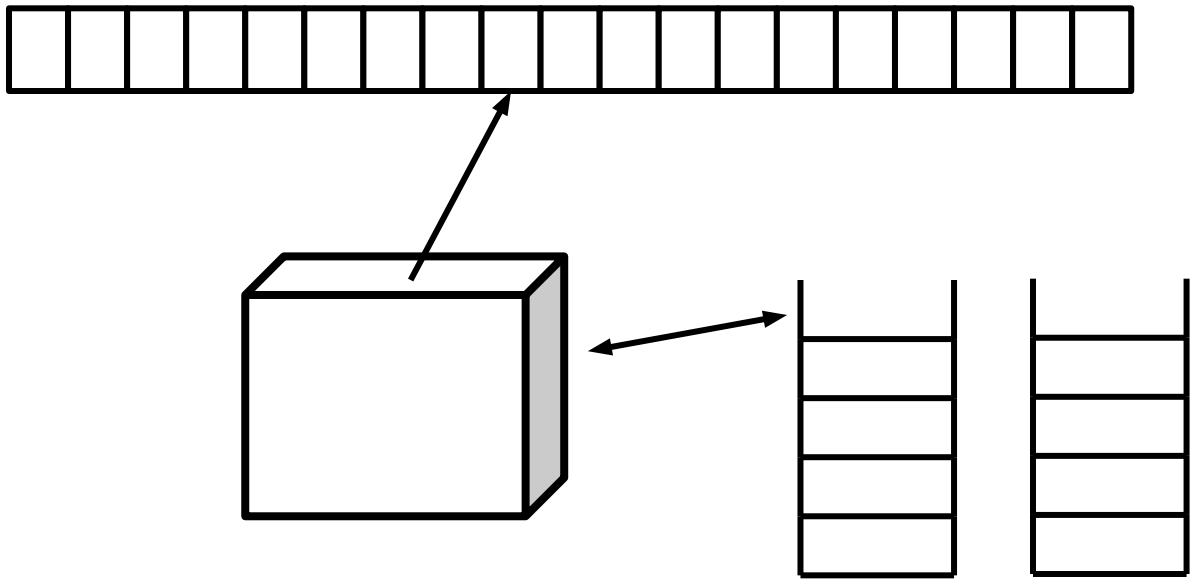
Conclusão

- 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 coleção 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

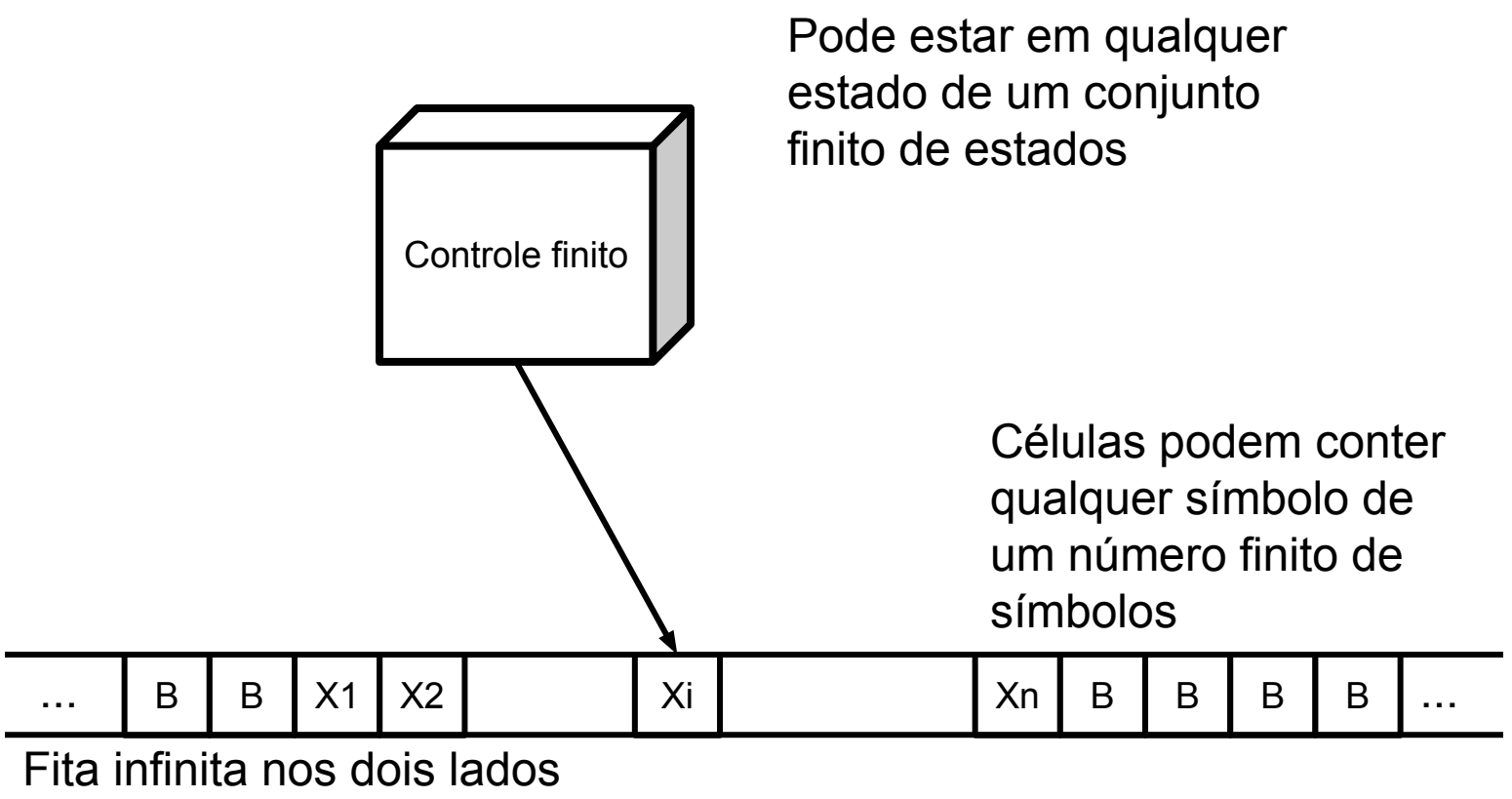
- 1936
  - Alonzo Church usou um sistema notacional denominado λ-cálculo para definir algoritmos
  - Alan Turing usou “máquinas” abstratas – autômatos – para definir algoritmos
- Conexão entre noção informal de algoritmo e definição precisa
  - ***Tese de Church-Turing***
- 1970
  - Yuri Matijasevic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

Na nossa terminologia

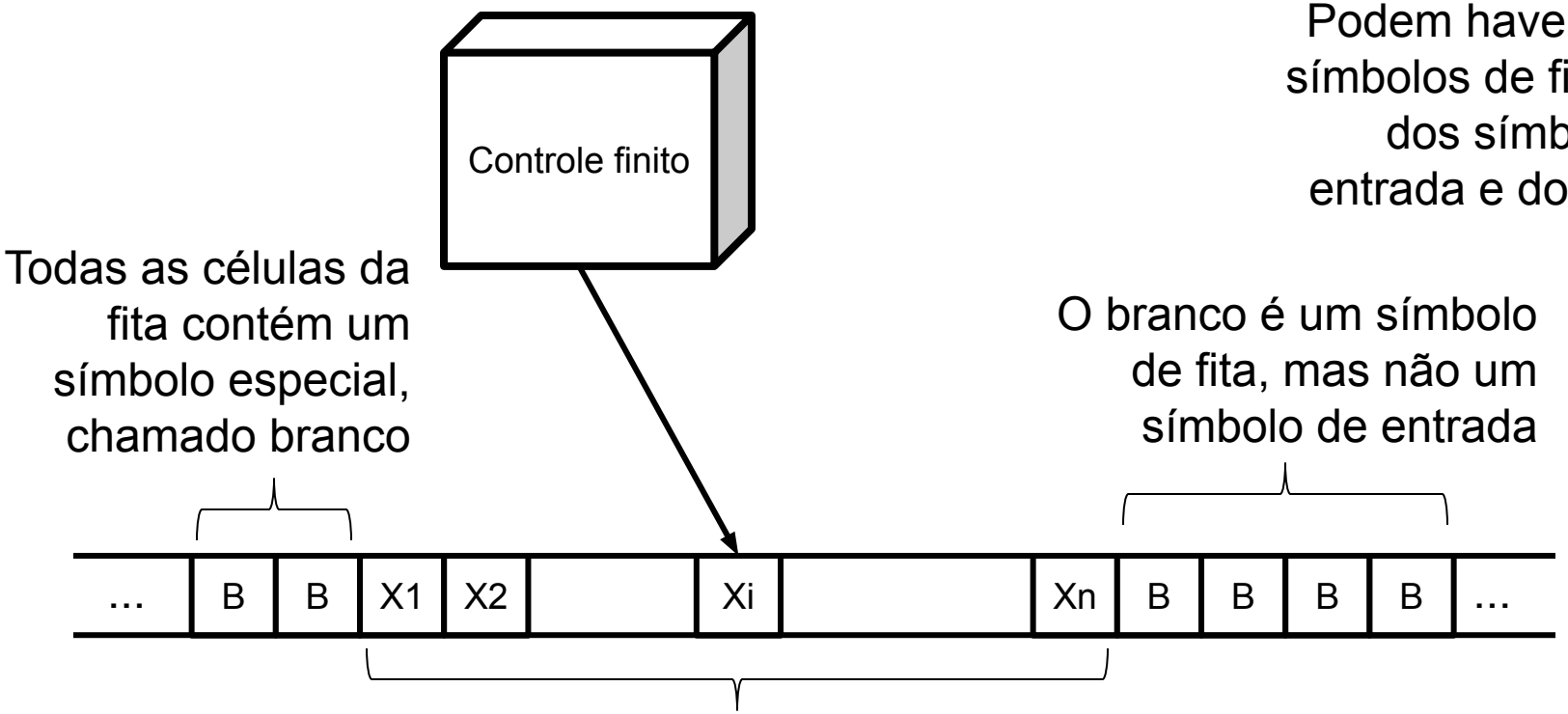
- D = {p | p é um polinômio com uma raiz inteira}
- Turing diz que:
  - Se conseguirmos projetar um autômato decisor para esta linguagem
  - Então D é decidível
  - Então existe um algoritmo para decidir (resolver) D
- Nesta aula, vamos estudar o formalismo de Turing – as “máquinas” de Turing
  - Se tiver curiosidade matemática, procure saber mais sobre λ-cálculo – as definições são equivalentes

A máquina de Turing

Uma notação para a máquina de Turing

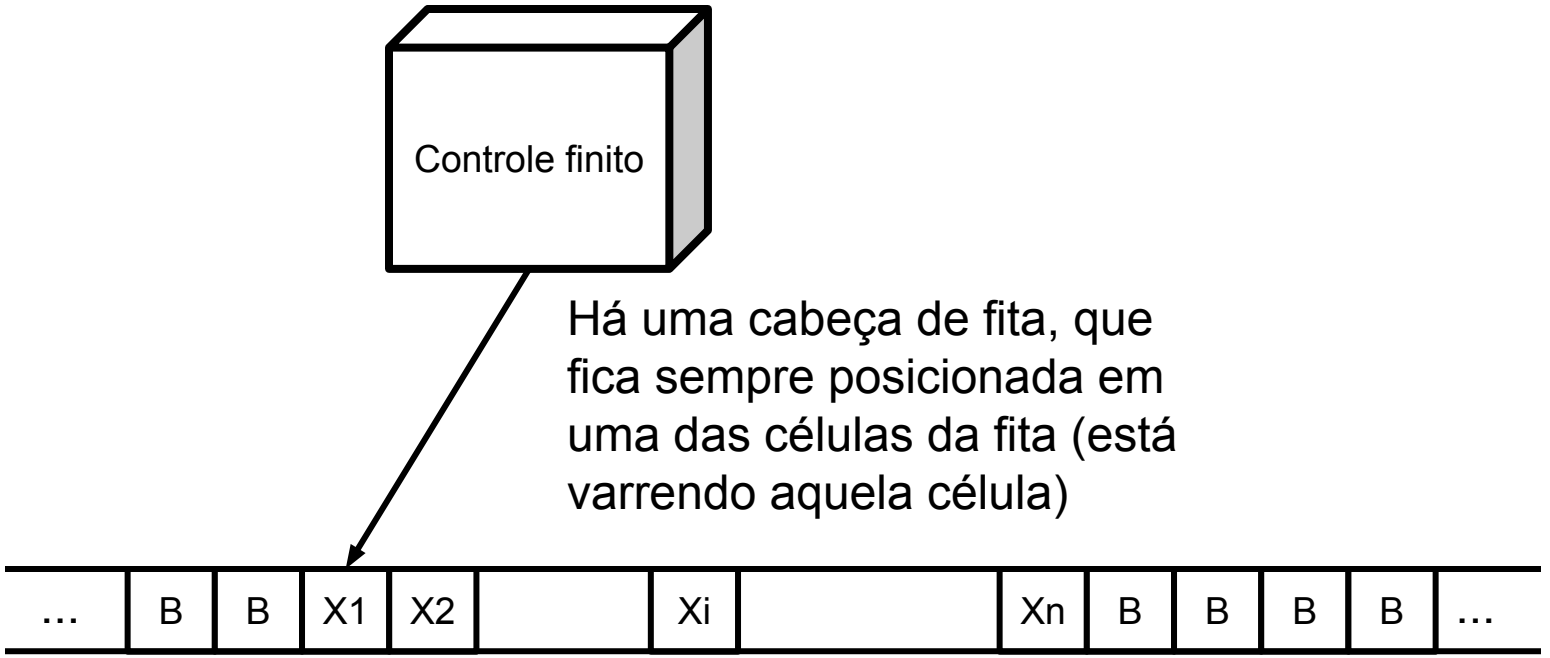


Uma notação para a máquina de Turing



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
  - $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{E,D\}$
  - 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

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},δ,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 →

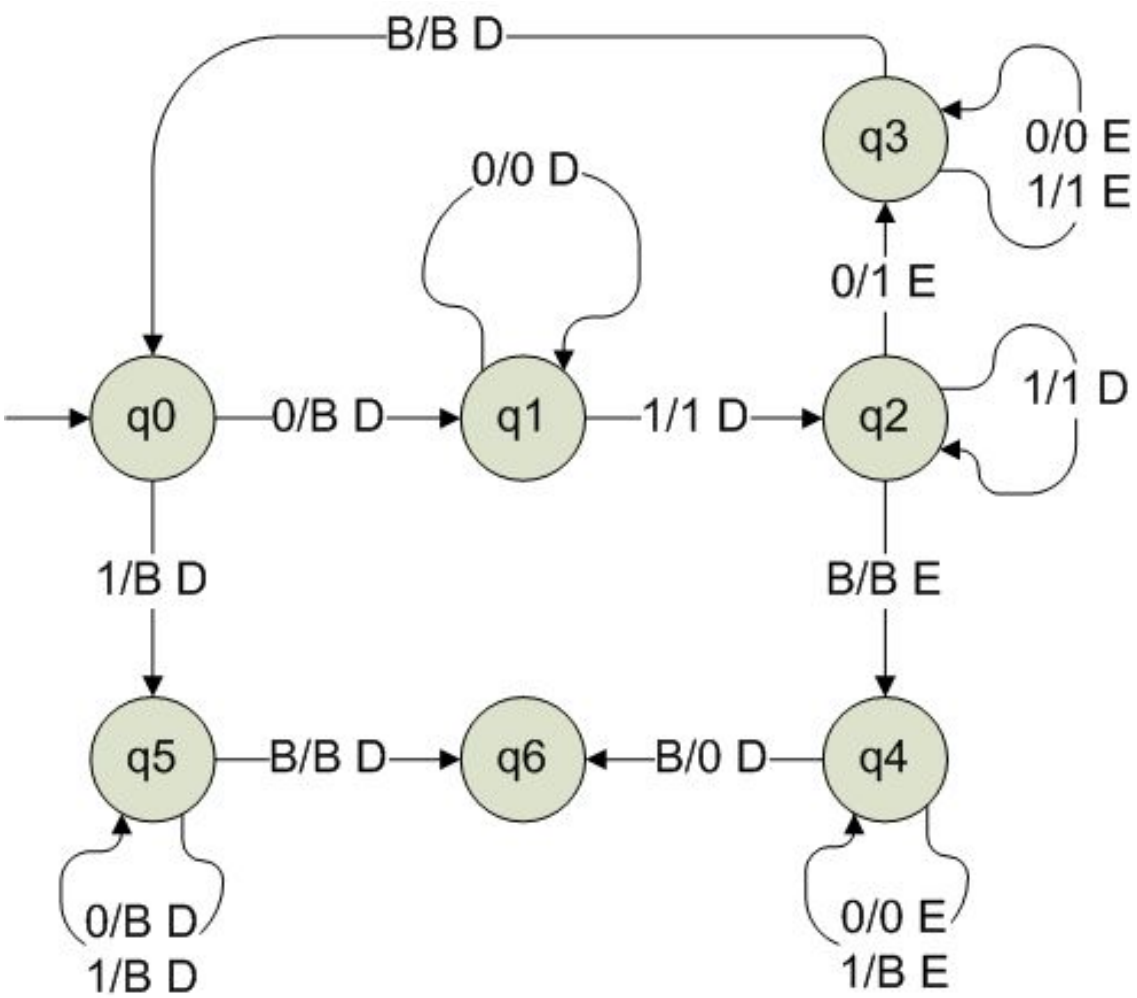
Exemplo

- 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 aceitação
  - 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

Máquina de Turing e sua parada

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, com 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)

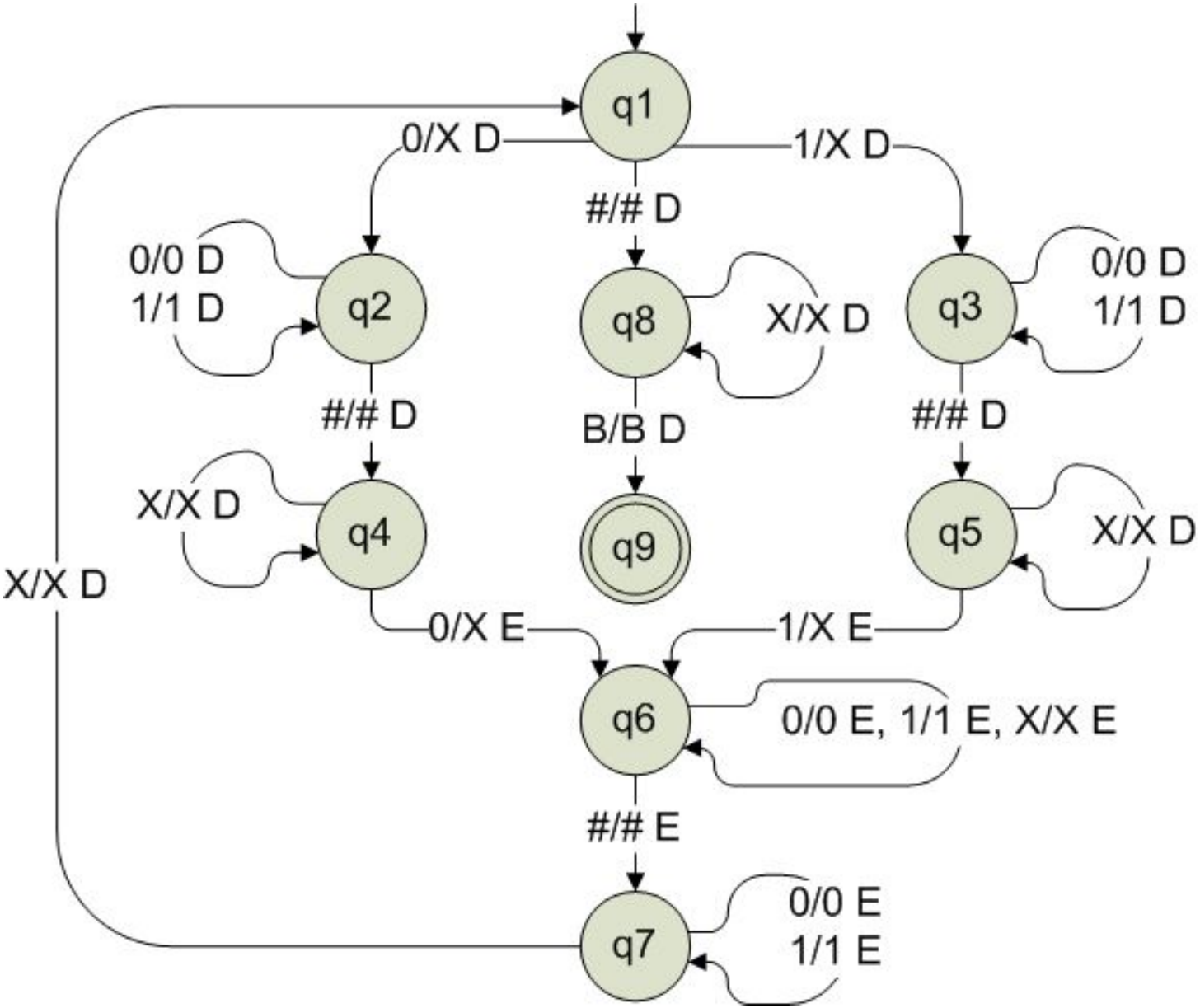
Projetando Máquinas de Turing

- 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
  - $A = \{0^{2^n} \mid n \geq 0\}$
    - Obs: 2^n = dois elevado a n
  - Todas as cadeias de 0s cujo comprimento é uma potência de 2

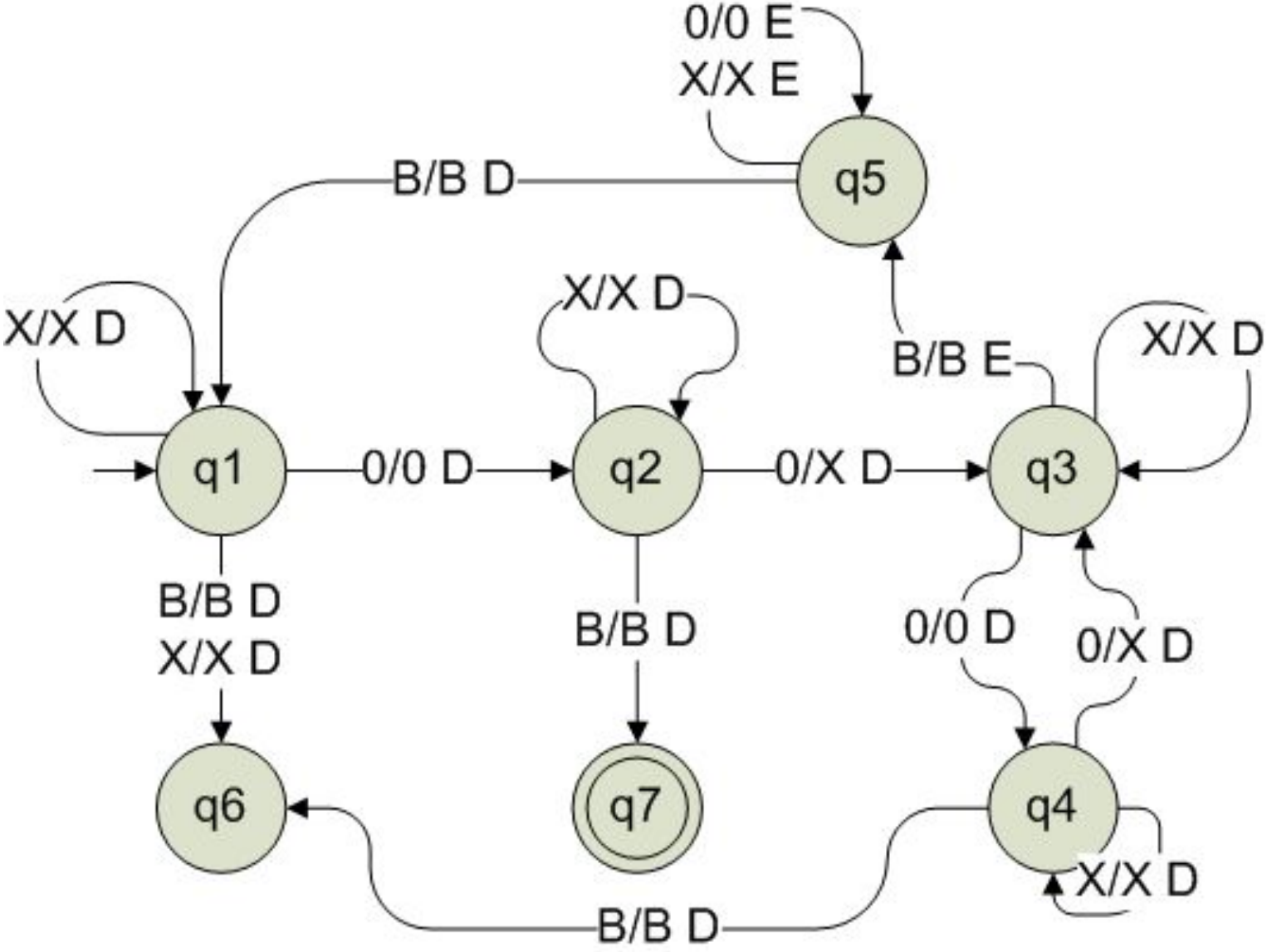
Solução (algoritmo)

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

Solucão (formal)



Fim

Aula 20 - Máquina de Turing