Computabilidade

Aula 3 – Máquina de Turing

Prof. Rodrigo Mafort

Últimas aulas

- Revisão de Complexidade Computacional
- Revisão de Autômatos

- Na última aula, terminamos a aula falando sobre os autômatos com pilha

Autômato com Pilha

- Uma evolução da ideia dos autômatos finitos.
- Possui uma pilha como componente adicional.
- A pilha é usada como uma memória para o autômato.

- Pilha:
  - Estrutura de dados que permite armazenar e recuperar informações de forma LIFO (last-in, first-out)
  - LIFO = O último elemento inserido na pilha é o primeiro a ser removido
  - As inserções e remoções são executadas exclusivamente no topo da pilha

Autômato com Pilha

- A pilha armazena informações sobre o histórico das transições realizadas pela máquina.

- O autômato pode ler um símbolo de entrada, fazer uma transição de estado e, opcionalmente, modificar a pilha adicionando ou removendo elementos.

- A transição de estado é determinada por três fatores:
  - o estado atual
  - o símbolo lido da fita de entrada
  - o topo da pilha

Autômatos com Pilha

- Assim como em um autômato finito, um AP pode ser determinístico ou não-determinístico.

- Em autômatos determinísticos com pilha:
- Para cada combinação de estado, símbolo de entrada e topo da pilha existe apenas uma transição possível.

Autômatos com Pilha

- Um autômato com pilha é formado por um 6-upla:
- $M = (\Sigma, \Gamma, Q, \delta, q_0, F)$
- Onde:
  - Σ é o alfabeto de símbolos de entrada (símbolos que a máquina reconhece)
  - Γ é o alfabeto de símbolos que podem ser escritos na pilha
  - Q é um conjunto finito de estados possíveis para o autômato
  - δ é o programa do autômato: $\delta: Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma$
  - q_0 é o estado inicial do autômato (ao ser iniciado, ele começa no estado q_0)
  - F é um subconjunto de Q chamado de estados finais. Se M terminar em um estado $q \in Q$ então a máquina “aceitou” ou “reconheceu” a entrada.

- Vale observar que $\varepsilon \in \Sigma$ e $\varepsilon \in \Gamma$. Onde ε significa uma entrada vazia (seja para a fila de entrada ou para a pilha)

Autômatos com Pilha

- Observe que cada transição é formada por:
  - $(q_i, a, b) \rightarrow (q_j, c)$

- Onde:
  - $q_i \in Q$ é o estado atual da máquina
  - $a \in \Sigma$ é a informação lida da fita de entrada
  - $b \in \Gamma$ é a informação que consta no topo da pilha
  - $q_j \in Q$ é o estado para o qual a máquina deve transacionar
  - $c \in \Gamma$ é a informação que deve ser escrita no topo da pilha

- Se estou no estado q_i E li a na fila E b está no topo da pilha então vá para o estado q_j E escreva c na pilha

Atividade:

- Pense em um AFD com Pilha para reconhecer uma linguagem formada por 0^n1^n.

- Por exemplo:
  - 000111 e 01 são aceitos
  - Enquanto 001100 e 00110011 não são aceitos

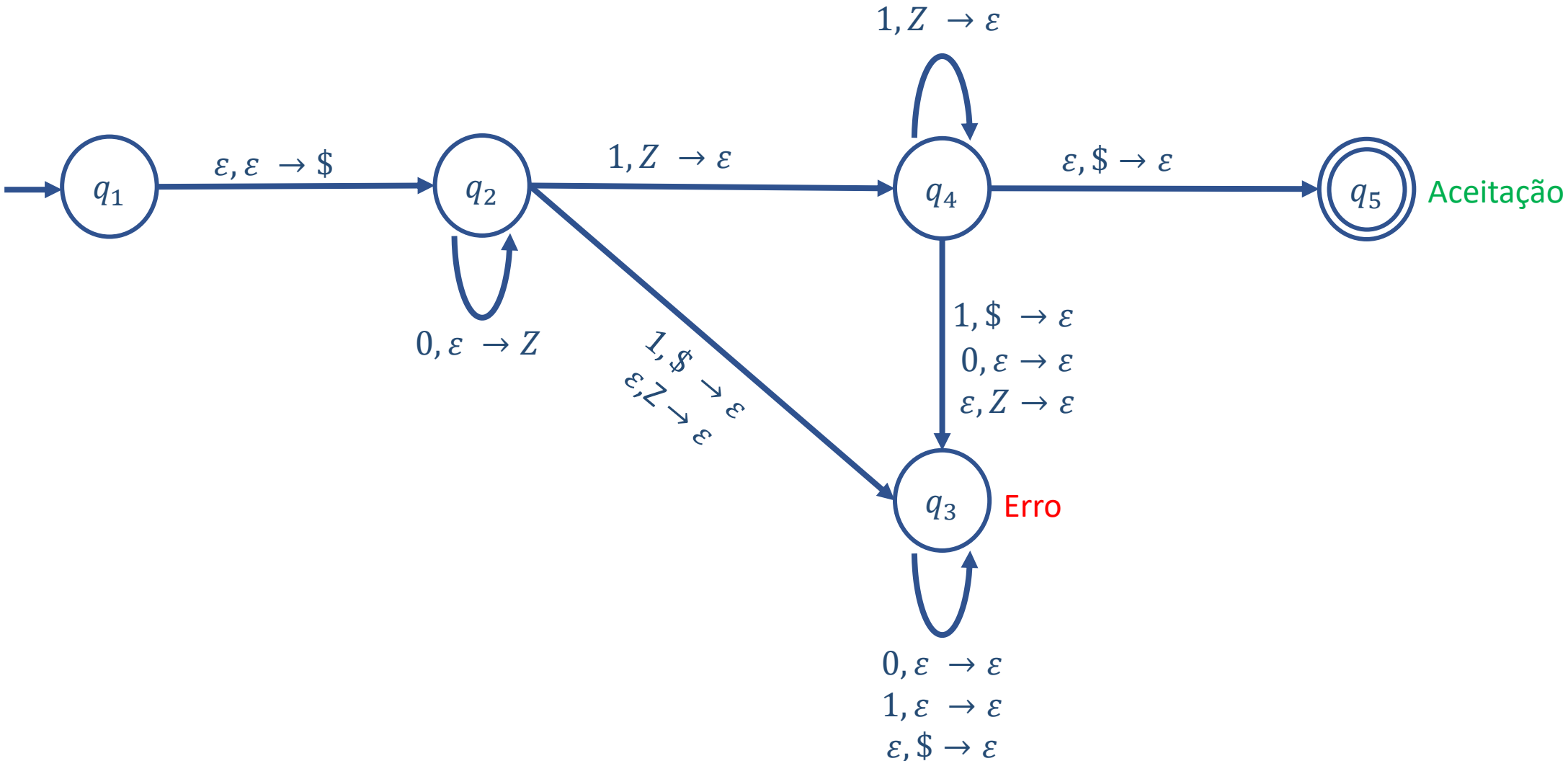
- Dica: Como verificar se a pilha está vazia?
  - Escrever um símbolo logo no início do autômato.

Resposta

- $M = (\{0,1,\varepsilon\}, \{\$, Z, \varepsilon\}, \{q_1, q_2, q_3, q_4, q_5\}, \delta, q_1, \{q_5\})$
- Função de Transição δ:

Estado Atual	Símbolo lido da fila	Valor lido da pilha	→	Novo Estado	Valor escrito na pilha
q_1	ε	ε		q_2	\$
q_2	0	ε		q_2	Z
q_2	1	\$		q_3	ε
q_2	ε	Z		q_3	ε
q_2	1	Z		q_4	ε
q_3	0,1	ε		q_3	ε
q_3	ε	\$		q_3	ε
q_4	1	Z		q_4	ε
q_4	1	\$		q_3	ε
q_4	0	ε		q_3	ε
q_4	ε	Z		q_3	ε
q_4	ε	\$		q_5	ε

Resposta


q_1
$\varepsilon, \varepsilon \rightarrow \$$
q_2
$1, Z \rightarrow \varepsilon$
q_4
$\varepsilon, \$ \rightarrow \varepsilon$
q_5
Aceitação
$1, Z \rightarrow \varepsilon$
$0, \varepsilon \rightarrow Z$
$1, \$ \rightarrow \varepsilon$
$\varepsilon, Z \rightarrow \varepsilon$
$1, \$ \rightarrow \varepsilon$
$0, \varepsilon \rightarrow \varepsilon$
$\varepsilon, Z \rightarrow \varepsilon$
q_3
Erro
$0, \varepsilon \rightarrow \varepsilon$
$1, \varepsilon \rightarrow \varepsilon$
$\varepsilon, \$ \rightarrow \varepsilon$

Pilha

- O uso da pilha implica que a leitura e escrita deve ser realizada em um único ponto: o topo da pilha.
- Não existe a opção de salvar em um outro ponto da pilha.
- Isso limita a capacidade do autômato.

Máquina de Turing

- Proposta por Alan Turing em 1936
- A máquina de Turing é um autômato que usa uma fita para armazenar as informações
- Essa fita pode ser lida, escrita e movimentada para a esquerda e para a direita.
- A fita possibilita uma memória infinita para a máquina e assegura a habilidade de ler as entradas mais de uma vez. Permite também sobrescrever valores das entradas.

Autômatos Finitos vs Máquina Turing

- Os autômatos de pilha possuem uma fita com a entrada. Essa fita pode ser lida e a cada leitura a fita se movimenta para a próxima entrada. A pilha é usada como uma memória auxiliar pelo autômato.

- Nas máquinas de Turing a entrada e a memória formam uma única fita. Os dados da entrada são escritos na fita, que pode ser alterada e movimentada livremente (para esquerda e para direita) pela máquina.

- Ao permitir escrever na fita, a Máquina de Turing permite alterar os dados codificados na fita.

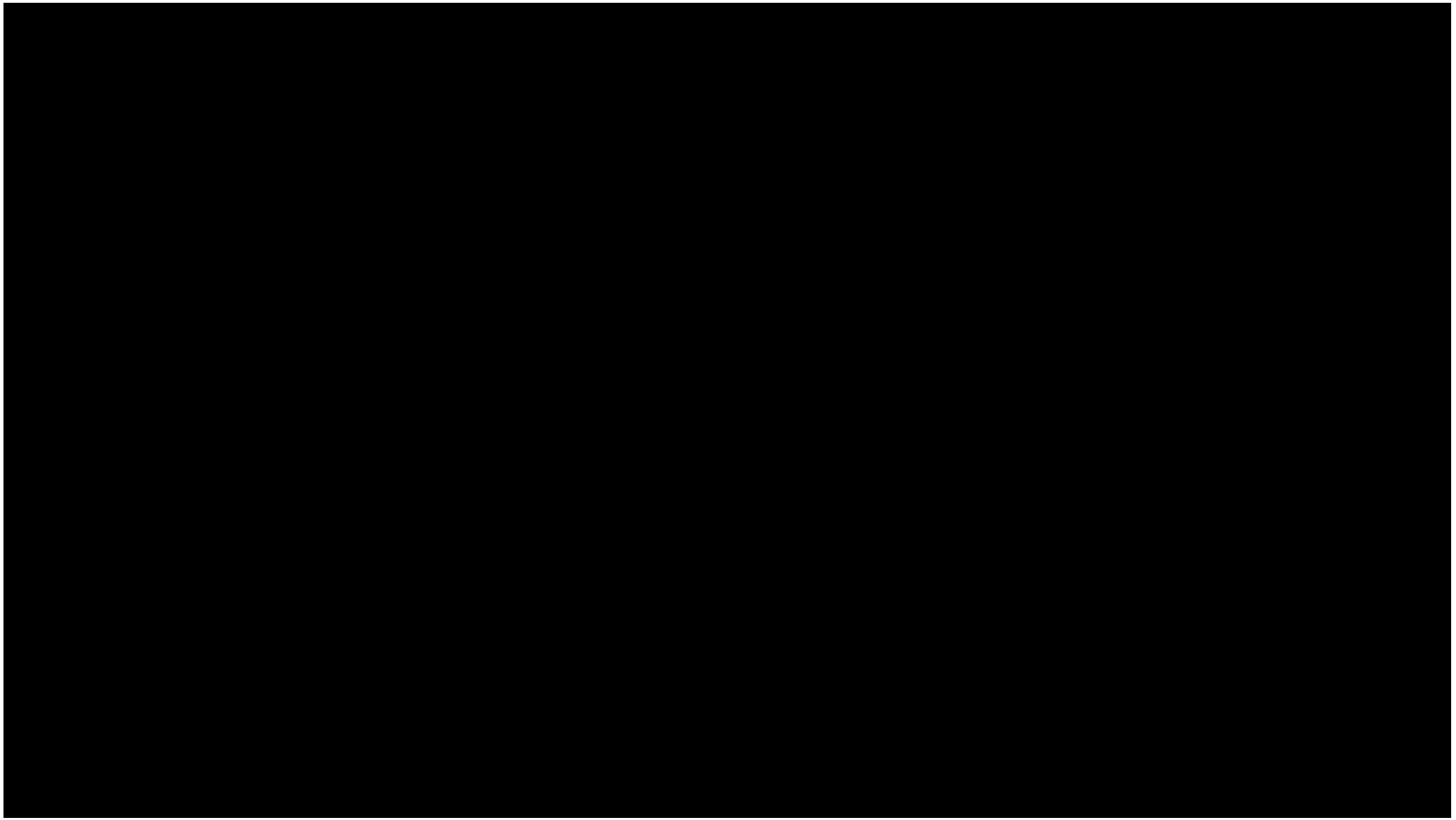
Autômato Finitos vs Máquina Turing

- Uma máquina de Turing pode ler e escrever na fita
- A cabeça de leitura pode ser mover para a esquerda e para a direita
- A fita é infinita

Máquina de Turing

- Composta por três partes:
  - Fita
  - Unidade de Controle
  - Programa

Máquina de Turing “Física”



https://www.youtube.com/watch?v=E3keLeMwfHY

Definição formal

- Uma máquina de Turing é definida por um 7-upla:
- $M = (Q, \Sigma, \Gamma, Q, \delta, q_0, q_{aceita}, q_{rejeita})$
- Onde:
  - Q é um conjunto finito de estados possíveis para a máquina
  - Σ é o alfabeto de símbolos de entrada (sem o símbolo de branco β)
  - Γ é o alfabeto da fita ($\beta \in \Gamma$ e $\Sigma \subseteq \Gamma$)
  - δ é o programa da máquina: $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{E, D\}$
  - q_0 é o estado inicial da máquina (ao ser iniciada, ela começa no estado q_0)
  - q_{aceita} é o estado de aceitação da máquina ($q_{aceita} \in Q$)
  - $q_{rejeita}$ é o estado de rejeição da máquina ($q_{rejeita} \in Q$ e $q_{rejeita} \neq q_{aceita}$)
- Diferente dos autômatos finitos, os estados de aceitação e rejeição fazem efeito imediatamente: a máquina para ao entrar em um desses estados.

Programa do Autômato

- δ é o programa da máquina (função de transição):
- $q_i \times \gamma_a \rightarrow \ q_j \times \gamma_b \times \ \mathrm{s}$

- Se
  - a máquina está no estado $q_i \in Q$ e
  - lê o símbolo $\gamma_a \in \Gamma$ da fita

- Então:
  - vá para o estado $q_j \in Q$ e
  - escreva o símbolo $\gamma_b \in \Gamma$ na fita e
  - mova a fita no sentido $s \in \{E, D\}$ (esquerda ou direita)

Atividade Proposta

- Suponha uma fita que é formada por dois números na base binária separadas por um sinal de #.

- Como escrever um “programa” capaz de verificar se os dois números são iguais usando uma máquina de Turing?

- Pergunta 1: Qual é o alfabeto de entrada Σ na fita?
- Pergunta 2: Quais símbolos precisamos ler ou escrever na fita (Γ) além de Σ e β (branco)?
- Pergunta 3: Quais e quantos estados (além de q_0, q_{aceita} e $q_{rejeita}$) vamos precisar?
- Pergunta 4: Como definir as transições (o programa)?

Ideias:

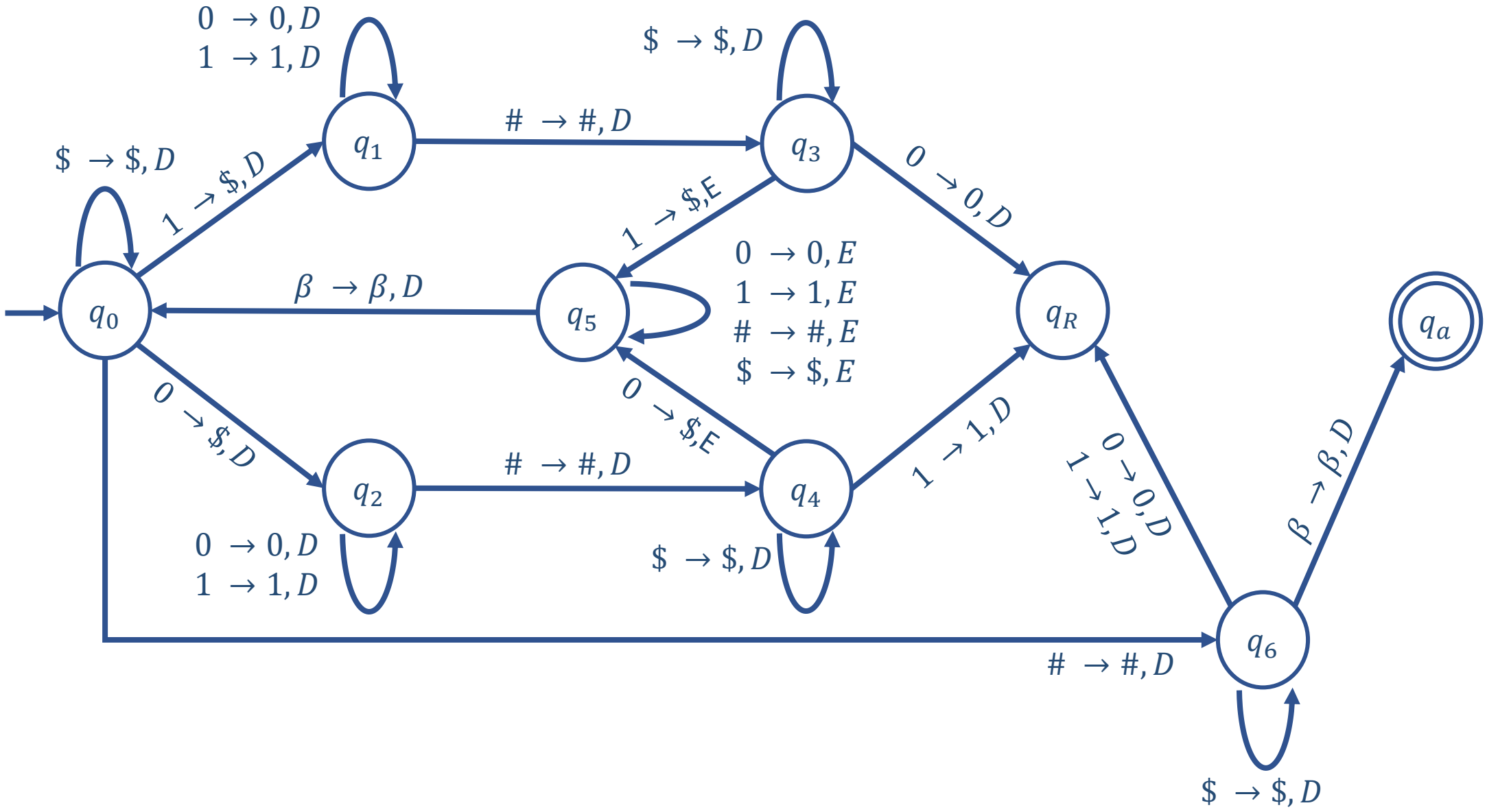
- Pergunta 1: Qual é o alfabeto de entrada Σ na fita?
  - $\Sigma = \{0,1,\#\}$
- Pergunta 2: Quais símbolos precisamos ler ou escrever na fita além de Σ e β (branco)?
  - $\Gamma = \{\beta, \$, 0,1,\#\}$ (trocar símbolos já lidos e processados por $)
- Pergunta 3: Quais e quantos estados (além de q_0, q_{aceita} e $q_{rejeita}$) vamos precisar?
  - Meu modelo tem 9 estados (inicial, aceitação, rejeição e outros 6)
- Pergunta 4: Como definir as transições (o programa)?
  - O que fazer ao encontrar um 0 ou 1 no número da esquerda?
  - Como buscar o mesmo dígito no número da direita?

Teste seu modelo

- https://turingmachinesimulator.com/

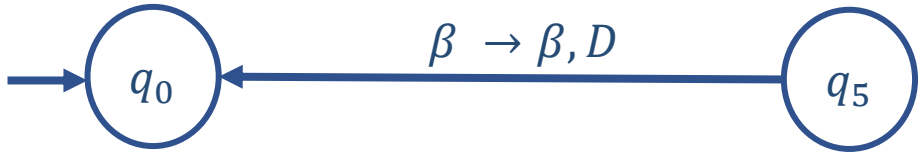
- init: q0 — o estado inicial
- accept: qaceita — o estado de aceitação

- q0,$ — estando no estado q0, se ler $
- q0,$,> — vá para o estado q0, escreva $ e se desloque para direita


q_0
q_1
q_2
q_3
q_4
q_5
q_6
q_R
q_a
0 → 0, D
1 → 1, D
$ → $, D
1 → $, D
→ #, D
0 → $, D
β → β, D
1 → $, E
0 → 0, E
1 → 1, E
→ #, E
$ → $, E
0 → $, E
1 → 1, D
0 → 0, D
1 → 1, D
β → β, D

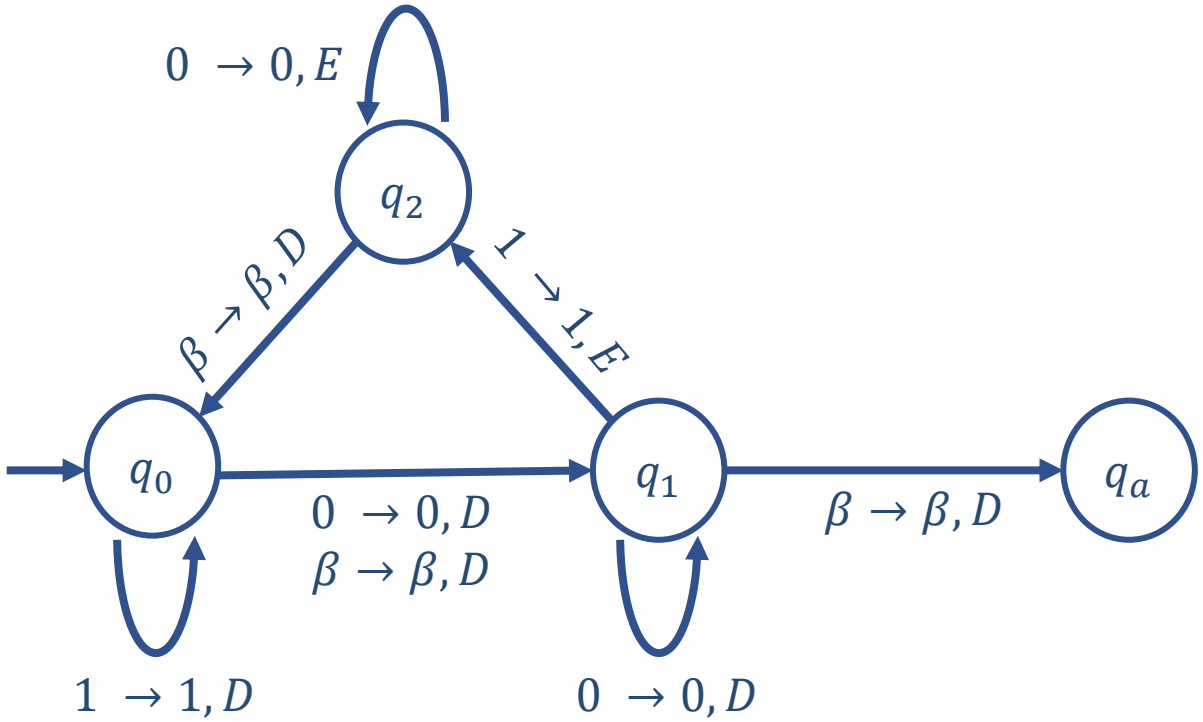
Exemplo 2:

- Escreva um programa para uma máquina de Turing verificar se um número na base dois é par.



Curiosidades:

- É possível gerar um loop infinito em autômato de pilha?
- E em uma máquina de Turing?



Variações da Máquina de Turing

- Máquinas de Turing Não Determinísticas
  - Consideram todas as possibilidades de transições
  - Toda máquina de Turing não determinística tem uma máquina determinística equivalente.

- Existem variantes da Máquina de Turing que admitem que a cabeça de leitura permaneça parada (além de se movimentar para esquerda e para a direita).
- Mas essa alteração não muda em nada a máquina: bastaria trocar essa possibilidade “ficar parado” por dois movimentos seguidos: vá para direita e, em seguida, vá para esquerda.

Variações da Máquina de Turing

- Existem ainda Máquinas de Turing com múltiplas fitas.
- Nesse caso, a função de transição deve indicar:
  - O que fazer considerando o valor de cada fita
  - Como atualizar as fitas (o que escrever e para onde movimentar)

- Embora máquinas com várias fitas pareçam mais poderosas, sua capacidade é a mesma de máquinas com apenas uma única fita.
- A fita é infinita. Basta considerar que cada uma das fitas está contida em uma única, sendo que o conteúdo de cada fita está separada por um símbolo especial (como no exemplo dos números iguais).

Máquinas de Turing e Computadores

- O conceito de máquina de Turing é a base na qual os computadores foram construídos.

- Todo computador é uma máquina de Turing?
  - +/- ....
  - Máquina de Turing tem memória infinita (fita infinita)
  - Computadores tem memória finita

- Tudo o que um computador pode fazer uma máquina de Turing também pode.

Algoritmos e Máquinas de Turing

- Em 1900, um matemático chamado David Hilbert disse que todo problema matemático bem formulado por ser resolvido por uma sequência bem estruturada de passos.

- Ele deu um exemplo: É possível escrever uma sequência de instruções para verificar se um polinômio apresenta ou não uma raiz inteira.

- Hoje sabemos que essa afirmação é falsa. Mas para demonstrar isso foi necessário definir formalmente o que é um algoritmo. Em 1936, Alonzo Church e Alan Turing definiram formalmente (ao mesmo tempo usando abordagens diferentes) o que são algoritmos.

Tese de Church-Turing

- **A noção intuitiva de um algoritmo é igual aos algoritmos de Máquinas de Turing.**
- Isso é:
  - Todo algoritmo (como conhecemos hoje) apresenta uma versão equivalente em uma Máquina de Turing.
  - Em outras palavras, todo algoritmo pode ser transformado em um conjunto de estados e funções de transições (um algoritmo) da Máquina de Turing.
- Esse conceito permitiu demonstrar os limites da computação.
- Isso é: problemas que não podem ser tratados computacionalmente (não é possível definir um algoritmo para resolver esse problema).
- Dois casos interessantes:
  - Problema da Parada (tema da próxima aula)
  - Em 1970 foi demonstrado que o problema das raízes inteiras dos polinômios não pode ser tratado computacionalmente (não é possível construir um algoritmo que resolva esse problema).