Extensões da Máquina de Turing

Prof^a Jerusa Marchi

Departamento de Informática e Estatística

Universidade Federal de Santa Catarina

e-mail: jerusa@inf.ufsc.br

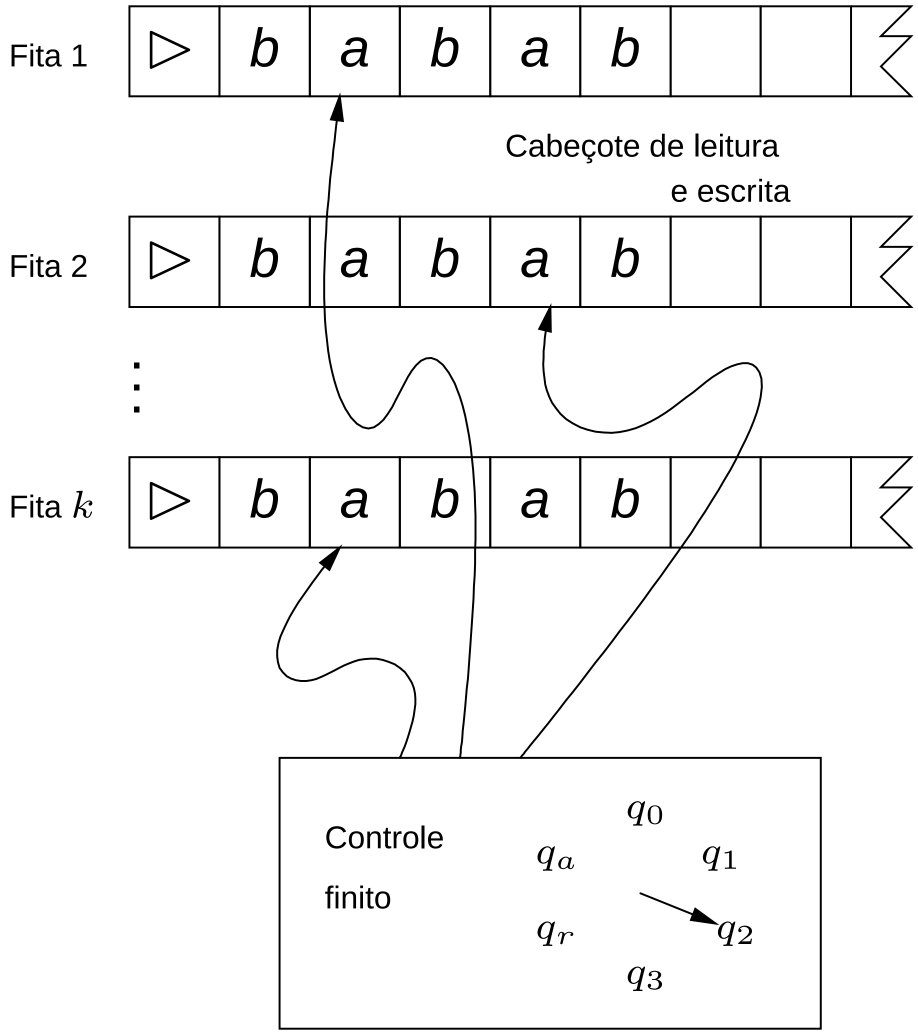
Máquinas de Turing

- são mecanismos poderosos que operam de modo lento e, por vezes, desajeitado
- podem ser melhorados, ou seja, podem ser incluídas modificações que reduzem a complexidade e facilitam o entendimento do funcionamento das máquinas
  - nenhuma destas modificações aumenta o poder computacional das máquinas
  - esse fato sugere que a Máquina de Turing é, de fato, um mecanismo computacional definitivo

Fitas Múltiplas

- Considera-se que o mecanismo possua múltiplas fitas
- Cada fita possui seu cabeçote de leitura/escrita
- A máquina pode, em um passo de operação, ler os símbolos de todos os cabeçotes, e dependendo destes símbolos e de seu estado atual:
  - regravar algumas células lidas
  - mover alguns cabeçotes à esquerda ou à direita
  - mudar de estado

Fitas Múltiplas



Fitas Múltiplas

- Uma Máquina de Turing com k fitas é uma séctupla:

$$M = (K, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$$

  - K = conjunto finito de estados
  - Σ = alfabeto de entrada $\cup \{\triangleright\}$
  - Γ = alfabeto da fita, onde $\sqcup \in \Gamma$ e $\Sigma \subseteq \Gamma$
  - q_0 = estado inicial ($q_0 \in K$)
  - q_{accept} = estado de aceitação ($q_{accept} \in K$)
  - q_{reject} = estado de rejeição ($q_{reject} \in K$)
  - $\delta : K \times \Gamma^k \longrightarrow (K \times \Gamma^k \times \{\leftarrow, \rightarrow\}^k)$, ou seja:

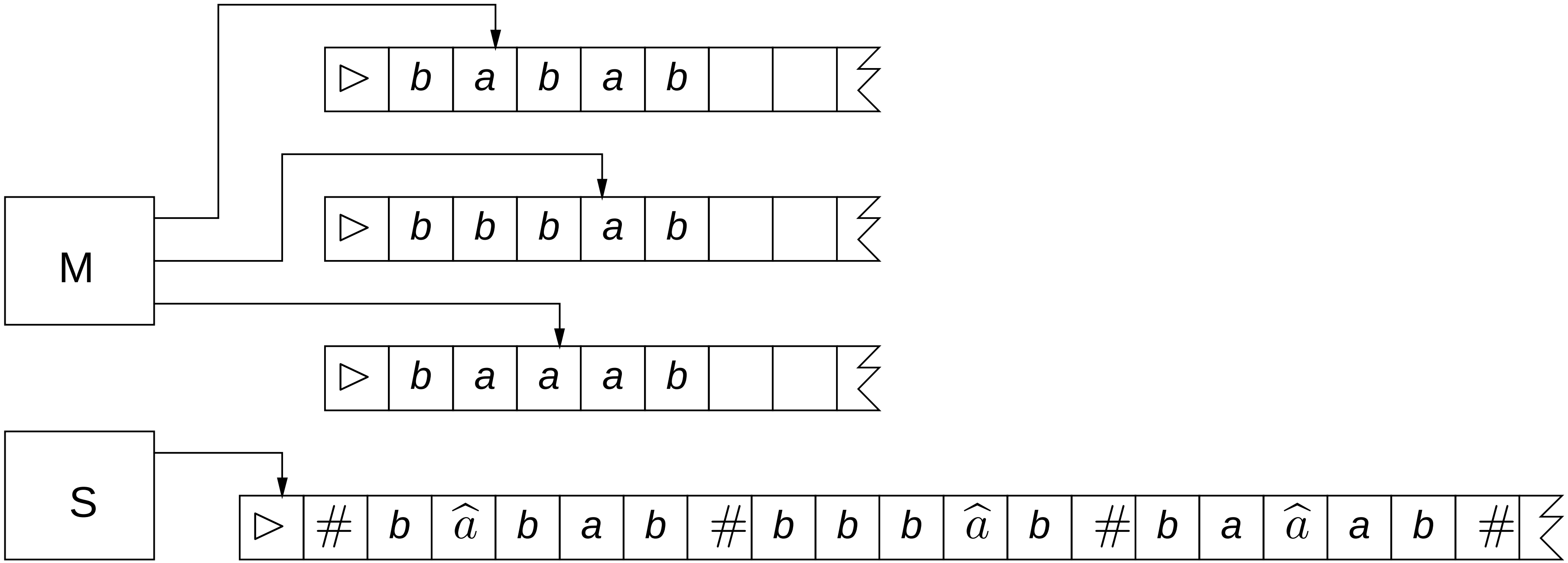
$$\delta(q_i, a_1, \cdots, a_k) \rightarrow (q_j, b_1, \cdots, b_k, m_1, \cdots, m_k\})$$

onde k é o número de fitas e m é o sentido do movimento do cabeçote.

Fitas Múltiplas

- **Teorema:** Para toda a máquina de Turing multifitas (M) há uma máquina de Turing com fita única (S) equivalente.
- **Prova:** Para demonstrar esta asserção basta apresentar como simular M com S
  Suponha que M possua k fitas. Então, S simula a operação das k fitas armazenando seus conteúdos em uma fita única. Isto pode ser feito utilizando um símbolo delimitador (como $\#$ por exemplo). Para assinalar a posição de cada um dos k cabeçotes, S pode reescrever o símbolo com uma marca (como um $\frown$ por exemplo). O alfabeto de fita da máquina é aumentado para considerar tais símbolos.

Fitas Múltiplas



Fitas Múltiplas

- Continuação
  - Inicialmente, grave na fita de S o conteúdo das k fitas de M, seguindo o padrão adotado
  - Para simular um único movimento, S vare sua fita do primeiro $\#$ que marca o final à esquerda até ($k$ésimo $+1$) $\#$ que marca o final à direita, deteminando assim os símbolos sob os cabeçotes de M
  - Então S realiza o segundo passo para atualizar as fitas conforme as transições de M
  - Se a qualquer ponto S move o cabeçote para a direita sobre um $\#$, isto significa que M move o cabeçote correspondente para o primeiro espaço em branco daquela fita. Então S escreve o símbolo branco e desloca todo o conteúdo da fita até o $\#$ mais à direita, uma célula à direita, e continua a simulação como antes

Máquinas de Turing Não Determinísticas

- Permite que a máquina, para certas combinações de estados e símbolos de entrada lidos, possa executar mais de um procedimento possível
- assim como para AFND, a computação de uma MT não determinística é uma árvore cujos ramos correspondem a diferentes possibilidades para a máquina
  - Se algum ramo leva a um estado de aceitação, a máquina aceita a entrada

Máquinas de Turing Não Determinísticas

- Uma Máquina de Turing não determinística é uma séptupla:

$$M = (K, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$$

Onde:

- K = conjunto finito de estados
- Σ = alfabeto de entrada $\cup \{\triangleright\}$
- Γ = alfabeto da fita, onde $\sqcup \in \Gamma$ e $\Sigma \subseteq \Gamma$
- q_0 = estado inicial ($q_0 \in K$)
- q_{accept} = conjunto de estados de parada que aceitam a entrada ($q_{accept} \in K$)
- q_{reject} = conjunto de estados de parada que rejeitam a entrada ($q_{reject} \in K$)
- $\delta : K \times \Gamma \longrightarrow \mathcal{P}(K \times \Gamma \times \{\leftarrow, \rightarrow\})$

Máquinas de Turing Não Determinísticas

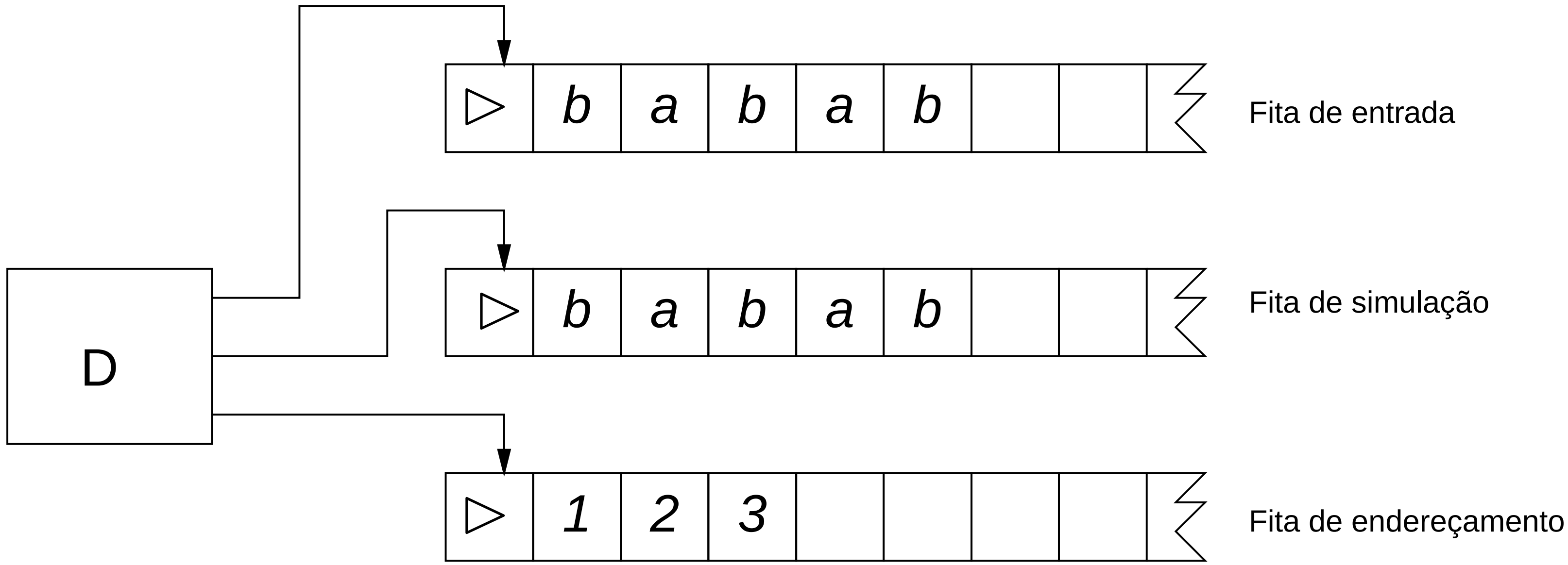
- **Teorema:** Toda máquina de Turing Não Determinística (N) possui uma máquina de Turing Determinística (D) equivalente
- **Prova:** Para demonstrar esta asserção basta apresentar como simular N com D. Na simulação, D tenta todos os possíveis ramos de computação de N. Se D encontra um estado aceitador em um destes ramos, D aceita a palavra e finaliza a computação. Caso contrário a simulação de D nunca termina.
  - As computações de N para algum w são vistas como uma árvore
  - Cada nó da árvore é uma configuração de N
  - A máquina D realiza uma busca em largura

Máquinas de Turing Não Determinísticas

Continuação:

- A máquina de Turing deteminística tem 3 fitas
  - Fita 1: contém a entrada e nunca é alterada
  - Fita 2: mantém uma cópia da fita de N em algum ramo da computação não determinística
  - Fita 3: mantém um registro da localização de D na árvore de computação não deteminística de N

Máquinas de Turing Não Determinísticas



Máquinas de Turing Não Determinísticas

Continuação:

- Representação de endereço:
  - Cada nodo da árvore pode ter no máximo b filhos (onde b é o maior conjunto de possíveis transições dadas pelas transições de N)
  - A cada nodo da árvore é associado um endereço construído sob o alfabeto $\Sigma_b = \{1, 2, ..., b\}$
  - Cada símbolo na palavra $addr$ que representa o endereço na fita 3 nos diz o qual o próximo passo a executar
  - Caso $addr$ seja um endereço inválido, ele é descartado

Máquinas de Turing Não Determinísticas

Continuação:

- Operação de D:
  1. Inicialmente a fita 1 contém a entrada w, as fitas 2 e 3 estão vazias
  2. A palavra é copiada da fita 1 para a fita 2
  3. A fita 2 é usada para simular N com a entrada w em um ramo de computação não determinística. Antes de cada passo de N, a fita 3 é consultada para determinar qual escolha deve ser feita dentre as permitidas pela função de transição de N. Caso não haja mais símbolos na fita 3 ou a escolha não determinística seja inválida, aborte e vá para o próximo passo. Vá para o próximo passo se uma configuração de rejeição for encontrada. Se uma configuração de aceitação for encontrada, aceite a entrada
  4. Substitua a palavra na fita 3 com a próxima palavra em ordem lexicográfica crescente. Volte para o segundo passo

Máquinas de Turing Não Determinísticas

- Se a máquina N sempre pára em todos os ramos de computação, D vai sempre parar
- Dizemos que uma MT não determinística é um **decisor** se todos os ramos param sobre todas as entradas

Enumeradores

- O termo *linguagem recursivamente enumerável* pode ser utilizado para denotar uma linguagem Turing-reconhecível
- Enumerador - variante de máquina de Turing, na qual uma impressora é anexada a máquina.
  - A MT pode usar a impressora como um dispositivo de saída para imprimir cadeias
  - Toda a vez que a MT quiser adicionar uma cadeia à lista, ela a envia para a impressora

Enumeradores

- Um enumerador E inicia com uma fita de entrada em branco
- Se ele não parar poderá imprimir uma lista infinita de cadeias
- A linguagem enumerada por E é a coleção de todas as cadeias que E em algum momento imprime
- As cadeias podem ser geradas em qualquer ordem, possivelmente com repetições

Enumeradores

- **Teorema:** Uma linguagem é Turing-Reconhecível sse algum Enumerador a enumera
- **Prova:** Para demonstrar esta asserção, primeiro precisamos mostrar que, se tivermos um enumerador E que enumera a linguagem A, então uma MT M reconhece A. A MT M funciona como segue:
  - A entrada w é gravada na fita de M
  - M executa E - para toda a saída que E imprime, M a compara com w
  - Se as cadeias forem iguais M pára e aceita w
- Ou seja, M aceitará exatamente aquelas cadeias que aparecem na lista de E

Enumeradores

Continuação: A prova no sentido oposto, consiste em demonstrar que se a MT M reconhece a linguagem A, é possível construir um enumerador E que enumera A

- Chamemos de $s_1, s_2, s_3, \cdots$ a lista todas as palavras possíveis em Σ^*
- O enumerador E funciona da seguinte maneira:
  - Ignorando a entrada, E repete para $i = 1, 2, 3, \cdots$
  - Execute M por i passos de computação sobre cada entrada $s_1, s_2, s_3, ..., s_i$
  - Se quaisquer computações levam a estados de aceitação, imprima s_j correspondente
- Ou seja, se M aceita uma cadeia específica s, em algum momento s aparecerá na lista de cadeias enumeradas por E
- De fato, a cadeia s aparecerá infinitas vezes, dado que para cada entrada s_i, E executa M desde o começo