Aula 02 – Máquinas de Turing e Linguagens

Prof. Eduardo Zambon

Departamento de Informática (DI)
Centro Tecnológico (CT)
Universidade Federal do Espírito Santo (Ufes)

Algoritmos e Fundamentos da Teoria de Computação (ToCE)
Engenharia de Computação

Introdução

- Máquinas de Turing – *Turing machines* (TMs) são um mecanismo abstrato de computação.
- TMs servem como um modelo matemático de um computador.
- **Estes *slides***: definição, conceitos e propriedades de TMs.
- **Objetivos**: apresentar os fundamentos de TMs para o estudo de computações efetivas.

Referências

Chapter 8 – Turing Machines
T. Sudkamp
Chapter 3 – The Church-Turing Thesis
M. Sipser
Chapter 4 – Turing Machines and the Church-Turing Thesis
A. Maheshwari

TMs como Reconhecedoras de Linguagens

- TMs são o nosso paradigma para computação efetiva.
- TMs podem ser projetadas para computar funções e aceitar linguagens (similar a um autômato finito – AF).
- O resultado de uma computação pode ser definido:
  - pela configuração da fita quando a máquina termina, no caso de computação de uma função; ou
  - pelo estado aonde a computação termina, no caso de aceite de linguagens.
- Nesta aula vamos tratar TMs como reconhecedoras de linguagens.
- TM aceita ou rejeita uma string de entrada.
- Inicialmente, aceite definido pelo estado final da TM.
- Similar aos AFs, mas com uma diferença.
- Uma TM não precisa ler (consumir) toda a string de entrada para aceitá-la/rejeitá-la.

TMs como Reconhecedoras de Linguagens

Definição abaixo estende a definição da TM padrão para incluir estados finais $F \subseteq Q$.

Definição 8.2.1 (Sudkamp)

- Seja $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ uma TM.
- A string $u \in \Sigma^*$ é aceita por um estado final se a computação de M com entrada u termina em um estado final $q_f \in F$.
- Se a computação de M termina em um estado não-final $q_i \in Q \setminus F$, a string u é rejeitada.
- Uma computação que termina anormalmente (TM andar para esquerda na posição 0) também rejeita a entrada.
- $L(M)$ é a linguagem de M, o conjunto de todas as strings aceitas por M.

TMs como Reconhecedoras de Linguagens

- Uma linguagem aceita por uma TM é dita recursivamente enumerável.
- Resultados possíveis para uma computação de uma TM:
  1. TM para e aceita a entrada.
  2. TM para e rejeita a entrada.
  3. TM não para: loop infinito.
- Caso 3 não existia para AFs.
- M reconhece L: M aceita L mas pode entrar em *loop* para algumas das strings de entrada $w \in \Sigma^*$ ($w \notin L$).
- Isto é, as computações de M identificam as strings em L mas podem não dar uma resposta para as strings que não estão em L.

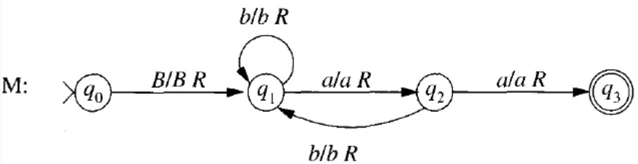
TMs como Reconhecedoras de Linguagens

- Linguagem recursiva: uma linguagem aceita por uma TM que para (*halts*) para todas as strings de entrada.
- Pertinência em uma linguagem recursiva é decidível.
- Isto é, a resposta para “$w \in$ L?” é sempre “sim” ou “não”.
- Computações de uma TM que para para toda entrada provêm um procedimento para determinar (decidir) se uma string está na linguagem ou não.
- Uma TM desse tipo decide a linguagem.
- Ser recursiva é propriedade da linguagem e não da TM!
- Há várias TMs que aceitam uma mesma linguagem L.
- Algumas dessas TMs podem decidir L outras podem apenas reconhecer L.
- Se existe ao menos uma TM que decide L $\Rightarrow$ linguagem é recursiva.

TMs como Reconhecedoras de Linguagens

Exemplo 8.2.1 (Sudkamp)

A TM M aceita $L = (\mathbf{a} \cup \mathbf{b})^*\mathbf{aa}(\mathbf{a} \cup \mathbf{b})^*$.



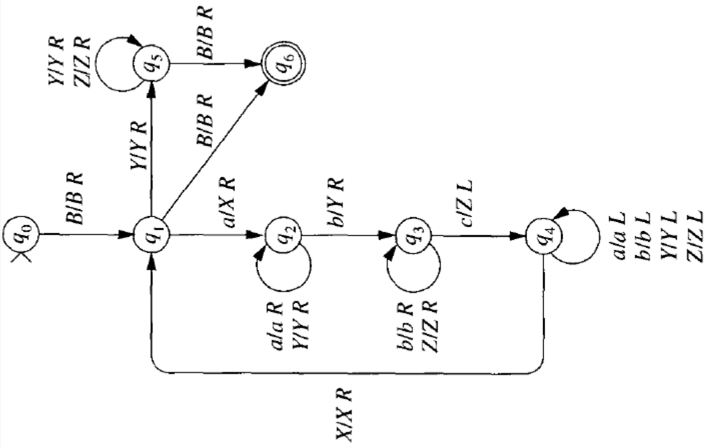
A computação

$$q_0 BaabbB \vdash Bq_1 aabbB \vdash Baq_2 abbB \vdash Baaq_3 bbB$$

lê apenas metade da entrada antes de aceitar a string.
A linguagem L é recursiva (na verdade é regular).

Exemplo 8.2.2 (Sudkamp)

A linguagem recursiva $L = \{a^i b^i c^i \mid i \geq 0\}$ é aceita pela TM:



Critérios Alternativos de Aceite

- Definição 8.2.1: Aceite por estado final.
- Outros critérios de aceite existem.

Definição 8.3.1 (Sudkamp)

Seja $M = (Q, \Sigma, \Gamma, \delta, q_0)$ uma TM. Uma string $u \in \Sigma^*$ é aceita por parada se a computação de M com entrada u termina (normalmente).

- Computações que terminam anormalmente rejeitam a string (como de costume).
- Qualquer computação para uma entrada que não está na linguagem não termina.
- TMs com aceite por parada são usadas para reconhecimento de linguagens.

Critérios Alternativos de Aceite

O teorema abaixo mostra que os dois critérios de aceite são equivalentes.

Teorema 8.3.2 (Sudkamp)

As duas afirmações são equivalentes:

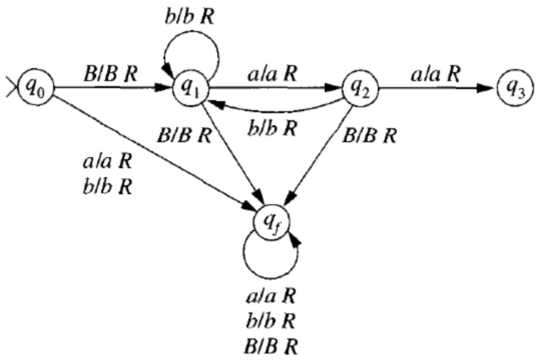
1. A linguagem L é aceita por uma TM que aceita por estado final.
2. A linguagem L é aceita por uma TM que aceita por parada.

- Seja M a máquina do caso 1 e M′ a máquina do caso 2.
- M pode ser construída a partir de M′ transformando todos os estados em estados de aceite.
- M′ pode ser construída a partir de M criando um estado de erro que deixa M′ em *loop* sempre que M tem uma computação sem sucesso.

Critérios Alternativos de Aceite

Exemplo 8.3.1 (Sudkamp)

A TM do Exemplo 8.2.1 é alterada para aceitar por parada.



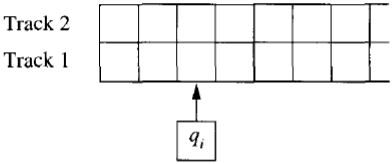
Outro critério de aceite: aceite por entrada – string aceita se a computação entra em um estado final (também é equivalente).

Variações de TM

- Existem muitas variações da TM padrão.
- Parecem aumentar a capacidade de computação da máquina mas não é o caso.
- Todas as variações que serão apresentadas a seguir são equivalentes à TM padrão.
- Uso das variações por conveniência.
- Tipos de variações a serem estudadas:
    - Máquinas multi-faixa (ou multi-pista).
    - Máquinas com fita *two-way*.
    - Máquinas multi-fita.

Máquinas Multi-Faixa

- Fita multi-faixa (*multitrack*): uma única fita dividida em duas ou mais faixas.
- Não confundir com máquina multi-fita.
- Fita com uma única faixa é a TM padrão.
- Cada posição de uma fita com *n*-faixas contém *n* símbolos do alfabeto.
- Exemplo de uma fita duas-faixas com a cabeça parada na posição 2.



Máquinas Multi-Faixa

- A cabeça lê/escreve todas as faixas de uma posição de uma vez.
- Uma posição de uma fita n-faixas é uma tupla: $[x_1, \ldots, x_n]$.
- Uma transição é dada por $\delta(q_i, [x_1, \ldots, x_n]) = [q_j, [x'_1, \ldots, x'_n], d]$, onde $d \in \{L, R\}$.
- Demais componentes são como da TM padrão.
- Entrada fica na faixa 1. Demais faixas inicialmente em branco.
- Aceite em TMs multi-faixa normalmente é por estado final.
- Teorema a seguir mostra que TMs padrão e TMs duas-faixas são equivalentes.
- Pode ser facilmente generalizado para n faixas.

Máquinas Multi-Faixa

Teorema 8.4.1 (Sudkamp)

Uma linguagem L é aceita por uma TM de duas-faixas M_2 se, e somente se, L é aceita por uma TM padrão M_1.

- $\Leftarrow$: Trivial. Se $s \in L$ é aceita por M_1 basta M_2 ignorar a presença da segunda fita.
- $\Rightarrow$: Temos de codificar cada posição da fita de M_2 (que contém dois símbolos) em um único símbolo da fita de M_1.
- Se Σ_2 e Γ_2 são os alfabetos de M_2, fazemos

$$\Sigma_1 = \Sigma_2 \times \{B\} \qquad \Gamma_1 = \Gamma_2 \times \Gamma_2.$$

- E a função de transição fica definida trivialmente:

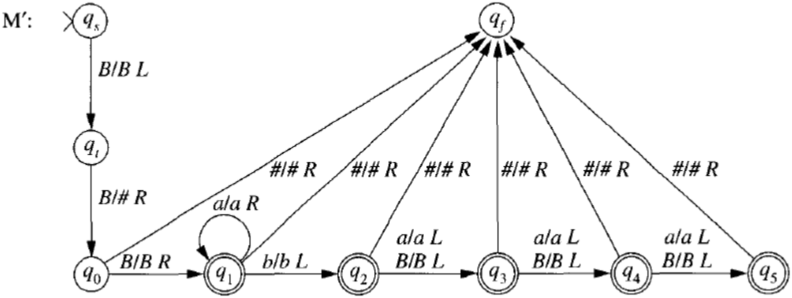
$$\delta_1(q_i, [x, y]) = \delta_2(q_i, [x, y]).$$

Máquinas com Fita *Two-Way*

- Fita *two-way*: estende indefinidamente nos dois sentidos.
- Demais componentes iguais a uma TM padrão.
- Fita não tem limite à esquerda $\Rightarrow$ entrada pode ficar em qualquer lugar da fita.
- Computação inicia com cabeça lendo primeiro branco à esquerda da entrada.
- TM *two-way* pode simular uma TM padrão.
- Basta colocar um símbolo especial $\#$ para simular o limite à esquerda da fita.
- Computação segue como de costume.
- Terminação anormal da TM *one-way*: TM *two-way* lê $\#$ e entra em um estado de sumidouro.

TM padrão que (tenta) andar quatro posições para a esquerda após ler o primeiro *b*.

TM *two-way* que simula a TM padrão acima.

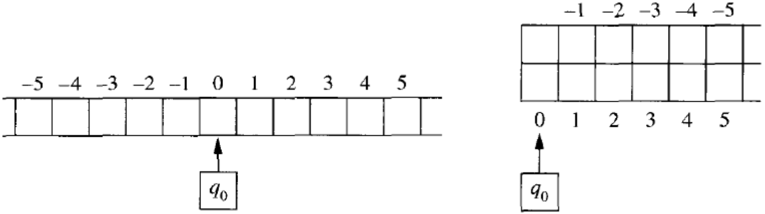


Máquinas com Fita *Two-Way*

Teorema 8.5.1 (Sudkamp)

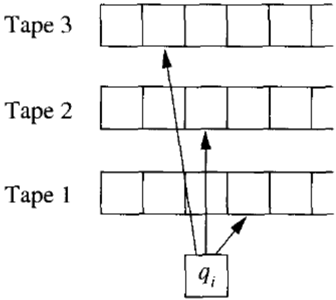
Uma linguagem L é aceita por uma TM *two-way* M_2 se, e somente se, L é aceita por uma TM padrão M_1.

- $\Leftarrow$: Trivial. Vide exemplo do slide anterior.
- $\Rightarrow$ Ideia geral: é possível "dobrar" uma fita *two-way* e simular M_2 usando uma máquina duas-faixas M'.
- Maior desafio é definir M' em termos de M_2. (Ver no livro.)
- Equivalência entre M' e M_1 é dada pelo Teorema 8.4.1.



Máquinas Multi-Fita

- Uma máquina *k*-fitas possui *k* fitas, cada uma com uma cabeça de leitura/escrita independente.
- Não confundir com multi-faixa: uma única cabeça de leitura.
- A máquina lê todas as fitas ao mesmo tempo e possui um único estado.



Máquinas Multi-Fita

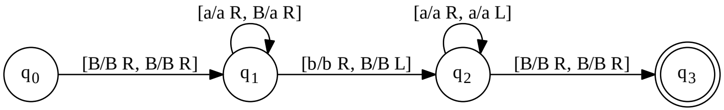
- Uma transição é determinada pelo estado da TM e pelos símbolos lidos por cada cabeça.
- Uma transição em uma TM multi-fita faz:
  1. Mudar o estado.
  2. Escrever um símbolo em cada uma das fitas.
  3. Reposicionar de forma independente cada cabeça.
- Nova ação de (não) movimento: *S - stationary*, deixa a cabeça parada.
- Transição de uma máquina duas-fitas lendo x_1 na fita 1 e x_2 na fita 2: $\delta(q_i, x_1, x_2) = [q_j; y_1, d_1; y_2, d_2]$ onde $x_i, y_i \in \Gamma$ e $d_i \in \{L, R, S\}$.
- Entrada fica na fita 1. Demais fitas inicialmente em branco.
- Todas as cabeças começam na posição 0 de todas as fitas.

Máquinas Multi-Fita

Duas vantagens de TMs multi-fita são a habilidade de copiar dados entre as fitas e de comparar strings em fitas diferentes.

Exemplo 8.6.1 (Sudkamp)

A TM duas-fitas abaixo aceita a linguagem $\{a^i b a^i \mid i \geq 0\}$.



- Uma computação copia os *a*'s iniciais para a fita 2.
- Depois de ler um *b*, compara conteúdo das duas fitas.
- A linguagem é recursiva. Por quê?

Máquinas Multi-Fita

- Uma TM padrão é uma máquina multi-fita com uma única fita.
- $\Rightarrow$ Toda linguagem recursivamente enumerável é aceita por uma máquina multi-fita.
- É possível mostrar que uma máquina duas-fitas pode ser simulada por uma máquina cinco-faixas. (Prova no livro.)
- Argumento geral: qualquer linguagem aceita por uma TM k-fitas é aceita por uma TM $2k + 1$-faixas.
- Equivalência de aceite entre máquinas padrão e multi-fita leva ao teorema abaixo.

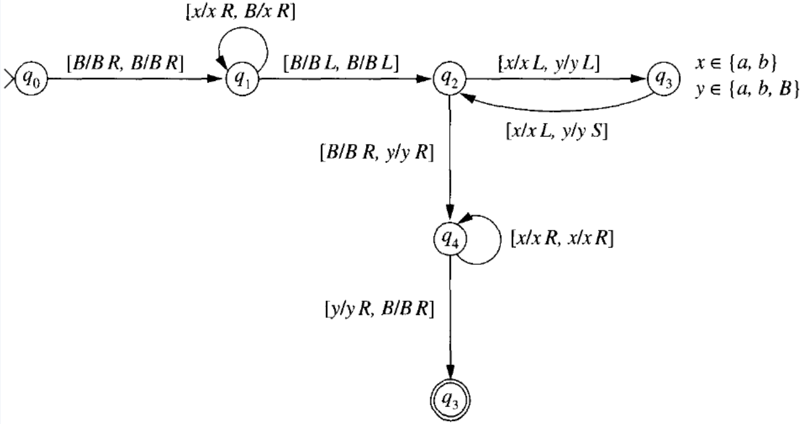
Teorema 8.6.1 (Sudkamp)

Uma linguagem L é aceita por uma TM multi-fita M_2 se, e somente se, L é aceita por uma TM padrão M_1.

Máquinas Multi-Fita

O exemplo a seguir ilustra o uso de fitas adicionais para armazenar e manipular dados em uma computação.

Exemplo 8.6.3 (Sudkamp)



Máquinas Multi-Fita

- TM do slide anterior aceita a linguagem $\{uu \mid u \in \{a, b\}^*\}$.
- Computação começa com uma cópia da entrada na fita 2.
- A seguir a fita 1 move 2 posições enquanto a fita 2 move somente 1 posição.
- *Loop* em q_4 compara a primeira metade da entrada com a segunda.
- Se elas são iguais, aceite em q_5 (figura errada no livro: dois estados q_3).

Máquinas de Turing Não-Determinísticas

- Uma Máquina de Turing Não-Determinística – *Nondeterministic Turing Machine* (NTM) é uma TM aonde o número de transições para uma configuração é ≥ 0.
- Função de transição de uma NTM: $\delta\colon Q \times \Gamma \rightharpoonup \mathcal{P}(Q \times \Gamma \times \{L, R\})$.
- Demais componentes iguais à TM determinística (DTM).
- DTM é um caso especial de NTM aonde $card(\mathcal{P}(\cdot)) \leq 1$.
- Quando δ indica mais de uma possível ação, a máquina escolhe arbitrariamente uma das transições.
- Pense em uma NTM como um processo que se multiplica sempre que há uma escolha.

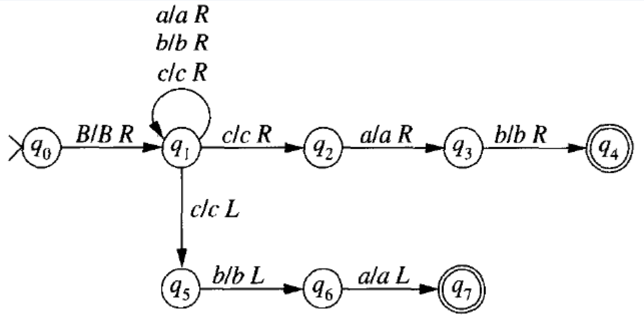
Máquinas de Turing Não-Determinísticas

- Critérios de aceite de NTM também podem variar.
- Seja *u* uma string de entrada.
- Aceite por estado final:
  - A string *u* é aceita por uma NTM se ao menos uma computação termina em um estado final.
  - Podem existir outras computações que param em estados não-finais ou não param $\Rightarrow$ irrelevantes.
  - Equivalente ao critério de aceite de um autômato finito não-determinístico (NFA).
- Aceite por parada:
  - A string *u* é aceita por uma NTM se ao menos uma computação termina.
  - Podem existir outras computações aonde a TM não para $\Rightarrow$ irrelevantes.
- Linguagem L(M): conjunto de todas as strings aceitas pela NTM M. (Mesma definição de sempre.)

Máquinas de Turing Não-Determinísticas

Exemplo 8.7.1 (Sudkamp)

A NTM abaixo aceita strings contendo um *c* precedido ou sucedido por *ab*.



A NTM "escolhe" um *c* e depois "escolhe" uma das condições para testar.

Máquinas de Turing Não-Determinísticas

Exemplo 8.7.1 – continuação

Possíveis computações da NTM para a entrada *acab*.

$q_0BacabB$	$q_0BacabB$	$q_0BacabB$
$\vdash Bq_1acabB$	$\vdash Bq_1acabB$	$\vdash Bq_1acabB$
$\vdash Baq_1cabB$	$\vdash Baq_1cabB$	$\vdash Baq_1cabB$
$\vdash Bacq_1abB$	$\vdash Bacq_2abB$	$\vdash Bq_5acabB$
$\vdash Bacaq_1bB$	$\vdash Bacaq_3bB$	
$\vdash Bacabq_1B$	$\vdash Bacabq_4B$	

A string é aceita pois a segunda computação termina no estado final q_4.

Máquinas de Turing Não-Determinísticas

- Introdução de não-determinismo em autômatos finitos não alterava o poder de computação: para qualquer NFA existe um DFA equivalente.
- E nas Máquinas de Turing?
- Mesmo resultado: para qualquer NTM existe uma DTM equivalente.
- Lembrar sempre que o poder de computação é diferente:
  - NFA e DFA aceitam linguagens regulares.
  - NTM e DTM aceitam linguagens recursivamente enumeráveis.
- DTM $\Rightarrow$ NTM: trivial.
- Mostrar equivalência requer construir um DTM M' a partir de uma NTM M aonde $L(M') = L(M)$.

Máquinas de Turing Não-Determinísticas

- NTM $\Rightarrow$ DTM: mostrar que múltiplas computações de M para uma dada entrada podem ser geradas sequencialmente e testadas por M′.
- Gerar sequencialmente computações $\Rightarrow$ execução sistemática (algorítmica) por M′ de todas as execuções de M.
- 1o. passo: ordenar (numerar) todas as alternativas de transições de M.
- Seja n o valor máximo de $card(\delta(\mathrm{q_i}, \mathrm{x}))$ para qualquer combinação de $q_i \in Q$ e $x \in \Gamma$ de M.
- Exemplo. Para a máquina do Exemplo 8.7.1: $n = 3$, pois se a TM está em q_1 lendo c, há três transições possíveis e esse é o número máximo para todos estados.

Máquinas de Turing Não-Determinísticas

- A numeração assume que $\delta(q_i, x)$ define n transições não necessariamente distintas para cada par q_i e x.
- Se δ define menos de n transições para um par de argumentos, uma das transições é escolhida (arbitrariamente) para receber vários números, até completar a ordenação.
- A ordenação das transições é arbitrária, isto é, qualquer uma serve.
- Isso porque a DTM M′ tem de passar por todas as possibilidades de transições.
- A tabela do próximo slide mostra uma possível ordenação da NTM do Exemplo 8.7.1.

Máquinas de Turing Não-Determinísticas

State	Symbol	Transition	State	Symbol	Transition
q_0	B	1 q_1, B, R 2 q_1, B, R 3 q_1, B, R	q_2	a	1 q_3, a, R 2 q_3, a, R 3 q_3, a, R
q_1	a	1 q_1, a, R 2 q_1, a, R 3 q_1, a, R	q_3	b	1 q_4, b, R 2 q_4, b, R 3 q_4, b, R
q_1	b	1 q_1, b, R 2 q_1, b, R 3 q_1, b, R	q_5	b	1 q_6, b, L 2 q_6, b, L 3 q_6, b, L
q_1	c	1 q_1, c, R 2 q_2, c, R 3 q_5, c, L	q_6	a	1 q_7, a, L 2 q_7, a, L 3 q_7, a, L

Máquinas de Turing Não-Determinísticas

- Uma sequência $(m_1, \ldots, m_k)$, com $1 \leq m_i \leq n$, define unicamente uma computação da NTM.
- Computação associada com a sequência tem k ou menos transições.
- Suponha que M está no estado q_i lendo x.
  - Se $\delta(q_i, x) = \emptyset$, a computação para.
  - Caso contrário, se a computação está no passo j, M executa a transição $\delta(q_i, x)$ numerada por m_j.

As computações da máquina do Exemplo 8.7.1 para entrada *acab* são definidas pelas sequências:

$$(1,1,1,1,1),\ (1,1,2,1,1),\ (2,2,3,3,1).$$

$$\begin{array}{lll}
\quad q_0BacabB & \quad q_0BacabB & \quad q_0BacabB \\
\vdash Bq_1acabB & \vdash Bq_1acabB & \vdash Bq_1acabB \\
\vdash Baq_1cabB & \vdash Baq_1cabB & \vdash Baq_1cabB \\
\vdash Bacq_1abB & \vdash Bacq_2abB & \vdash Bq_5acabB \\
\vdash Bacaq_1bB & \vdash Bacaq_3bB & \\
\vdash Bacabq_1B & \vdash Bacabq_4B &
\end{array}$$

Máquinas de Turing Não-Determinísticas

As computações da máquina do Exemplo 8.7.1 para entrada *acab* são definidas pelas sequências:

$$(1,1,1,1,1),\ (1,1,2,1,1),\ (2,2,3,3,1).$$

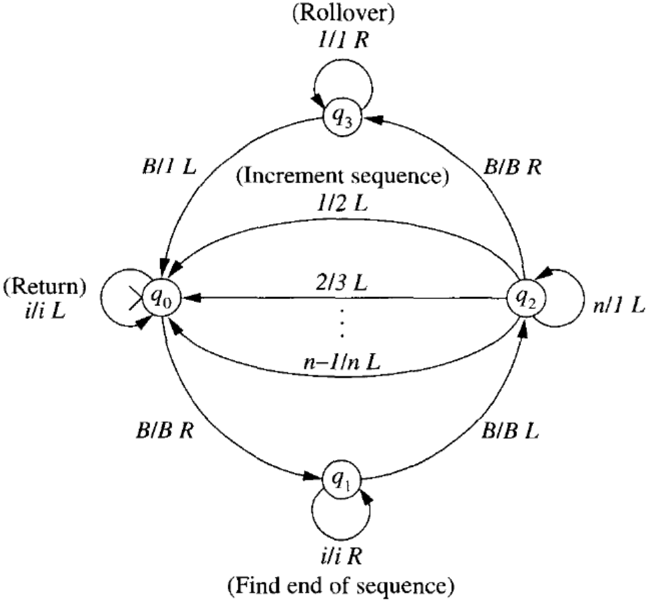
$$\begin{array}{lll}
\quad q_0BacabB\ 1 & \quad q_0BacabB\ 1 & \quad q_0BacabB\ 2 \\
\vdash Bq_1acabB\ 1 & \vdash Bq_1acabB\ 1 & \vdash Bq_1acabB\ 2 \\
\vdash Baq_1cabB\ 1 & \vdash Baq_1cabB\ 2 & \vdash Baq_1cabB\ 3 \\
\vdash Bacq_1abB\ 1 & \vdash Bacq_2abB\ 1 & \vdash Bq_5acabB \\
\vdash Bacaq_1bB\ 1 & \vdash Bacaq_3bB\ 1 & \\
\vdash Bacabq_1B & \vdash Bacabq_4B &
\end{array}$$

Máquinas de Turing Não-Determinísticas

- Para simular as execuções da NTM M, a DTM M′ precisa gerar todas as sequências finitas de inteiros entre 1 e n, onde os símbolos $1, 2, \ldots, n$ são símbolos da fita.
- A DTM GEN no próximo slide gera essas sequências.
- Primeiro as sequências de comprimento 1 são geradas na ordem.
- A seguir as sequências de comprimento 2, etc.
- A computação começa no estado q_0 lendo a posição 0.
- Quando GEN retorna à posição 0 a próxima sequência foi gerada.
- Exemplo: para $n = 2$, a máquina GEN produz

$$(1),\ (2),\ (1,1),\ (1,2),\ (2,1),\ (2,2),\ (1,1,1),\ \ldots$$

Máquinas de Turing Não-Determinísticas



Máquinas de Turing Não-Determinísticas

- 2o. passo: construir uma DTM M' equivalente à NTM M.
- Assuma que $M = (Q, \Sigma, \Gamma, \delta, q_0)$ aceita strings por parada.
- Facilita a prova mas não é restritivo. Qualquer máquina que aceita por estado final pode ser transformada em uma que aceita por parada.
- Vamos construir uma DTM $M' = (Q', \Sigma, \Gamma', \delta', q_0')$ que também aceita por parada.
- Note que o alfabeto de entrada é igual para garantir a equivalência das máquinas.
- A DTM M' é uma máquina três-fitas.

Máquinas de Turing Não-Determinísticas

- DTM M′ é construída para simular as computações de M.
- Correspondência entre sequências $(m_1, \ldots, m_k)$ e computações de M garante que todas as possibilidades são examinadas.
- Papel das três fitas de M′:
  - **Fita 1**: armazena a string de entrada.
  - **Fita 2**: simula a fita de M.
  - **Fita 3**: contém as sequências $(m_1, \ldots, m_k)$ para guiar a simulação.

Máquinas de Turing Não-Determinísticas

Uma computação de M′ é composta pelas seguintes ações:

1. Uma sequência $(m_1, \ldots, m_k)$ é escrita na fita 3. (Inicialmente, a sequência (1).)
2. String de entrada é copiada da fita 1 para a fita 2.
3. A computação de M definida pela sequência na fita 3 é simulada na fita 2.
4. Se a simulação para antes de executar k transições, a computação de M′ para e aceita a entrada.
5. Senão, M′ gera a próxima sequência na fita 3 usando GEN e volta para o passo 2.

Máquinas de Turing Não-Determinísticas

- Passos do slide anterior podem ser formalizados para definir os componentes de M′. (Detalhes no livro do Sudkamp.)
- O que acontece para uma entrada $w \in L(M)$:
  - M possui ao menos uma execução que para e aceita w.
  - $\Rightarrow$ M′ simula uma dessas execuções acima e aceita w.
- O que acontece para uma entrada $w \notin L(M)$:
  - M não para. (Lembre que o aceite é por parada.)
  - $\Rightarrow$ M′ continua indefinidamente no ciclo de gerar uma nova sequência e simular a computação de M.
- $\Rightarrow L(M') = L(M)$.

Máquinas de Turing Não-Determinísticas

- NTMs podem ser definidas com fitas multi-faixa, fitas *two-way* ou múltiplas fitas.
- Em todos os casos é possível mostrar que todas aceitam precisamente as linguagens recursivamente enumeráveis.
- Ao invés de aceitar (reconhecer) linguagens, as TMs também podem ser usadas para enumerar os elementos de uma linguagem.
- Fora do escopo do curso.

Aula 02 – Máquinas de Turing e Linguagens

Prof. Eduardo Zambon

Departamento de Informática (DI)
Centro Tecnológico (CT)
Universidade Federal do Espírito Santo (Ufes)

Algoritmos e Fundamentos da Teoria de Computação (ToCE)
Engenharia de Computação