Generalizações de M.T.

- Veremos que em várias direções, não ganhamos poder computacional ao considerarmos máquinas de Turing mais potentes

  Ex: - Com fita infinita nos dois sentidos
  - Múltiplas fitas
  - Múltiplos cabeçotes de leitura ou escrita
  - Movimentação menos restrita do cabeçote

- Isso quer dizer que qualquer linguagem decidida ou semi-decidida, ou função computável por uma máquina de Turing de um tipo tabém o é por uma M.T. de outro.

- O número de passos aumenta polinomialmente

M.T. com múltiplas fitas

- A função de transição $\delta : (Q - F) \times \Sigma^{K} \longrightarrow Q \times \left((\Sigma - \{\triangleright\}) \cup \{\leftarrow, \rightarrow\} \right)^{K}$

- Vamos representar o conteúdo das K-fitas em uma única fita
  - Para isso usamos o símbolo # para separar o conteúdo das fitas

- Precisamos indicar em que posição está cada cabeçote
  - Para isso dobramos os símbolos do alfabeto, criando uma cópia de cada símbolo com um • acima

$\triangleright$	#	$\dot{a}$	b	c	#	a	$\dot{b}$	a	a	#	c	c	$\dot{c}$	#		...

- Transições: serão feitos duas varreduras
  - A máquina deve percorrer a fita até o $(k+1)$-ésimo # e guardar o símbolo com • em seus estados

    ↳ Para cada estado q e k-tupla $(a_1, \dots, a_k) \in \Sigma^*$ criamos os estados $q_{a_1}, q_{a_1 a_2}, \dots, q_{a_1 \dots a_k}$

  - Na segunda varredura executamos as operações em cada parte

    ⇝ onde mover o cabeçote significa mudar o •

  - Nesse caso, entendemos os #'s como $\triangleright$, que limita o cabeçote de andar p/ esq.

  - Também temos que nos preocupar quando precisamos nos mover p/ direita, "invadindo" o espaço de uma fita. Neste caso, temos que fazer um shift-right.

M.T. não-determinística

- Análoga à M.T. determinística

- Função de transição: $\Delta : (Q \setminus F) \times \Sigma \longrightarrow \mathcal{P}\left((\Sigma \setminus \{\triangleright\}) \cup \{\leftarrow, \rightarrow\}\right)$

  ⇝ Se a máquina está no estado q e lê σ, ela escolherá de forma não-determinística $(q', \sigma') \in \Delta(q, \sigma)$

  ⇝ Conjunto de possíveis transições

DEF: Uma M.T. Não-Determinística M **aceita** (ou **semi-decide**) $L \subseteq \Sigma_0^*$ se, para cada $w \in \Sigma_0^*$, temos que $w \in L$ se e somente se ao menos uma computação em M termina (para) em um estado final.

DEF: Uma M.T. Não-Determinística M **decide** $L \subseteq \Sigma_0^*$ se, para cada $w \in \Sigma_0^*$, temos que para cada $w \in \Sigma_0^*$, M sempre para, e se $w \in L$, M para no estado de aceitação (s). Isso implica que se $w \notin L$, M para no estado de rejeição (n).

DEF: Uma M.T. não-determinística M **computa** uma função ϕ se, para todo $w \in \Sigma_0^*$ M sempre para e, para todas computações, termina com valor $\phi(w)$.

- Isso não trará ganho computacional
  - → Podemos simular M.T.N.D com M.T.D.
  - → Toda linguagem aceita/decidida/computada por uma M.T.N.D pode ser aceita/decidida/computada por uma M.T.D.

Simulando M.T.ND com M.T.D.

- Precisamos de três fitas.

- Vamos considerar que as transições estão ordenadas
  - → é possível pois o conjunto de transições é finito
  - → ordenamos por símbolos em $C_\beta = \{1, \dots, \beta\}$

- Vamos simular de forma ordenada todas as possíveis transições de M

- A primeira fita guarda um backup da entrada.
- A segunda fita recebe, para cada simulação, uma cópia da entrada
- A terceira fita guarda a sequência das escolhas feitas

Ex: 2 3 7 4 indica uma sequência de profundidade 4 que escolheu a 2ª transição, depois a 3ª, depois a 7ª, ...

- NÃO PODEMOS FAZER UMA "BUSCA EM PROFUNDIDADE" PORQUE É POSSÍVEL QUE M POSSUA ALGUMA COMPUTAÇÃO INFINITA.

- ASSIM, PRIMEIRO EXECUTAMOS TODAS AS "BUSCAS" DE PROF. 1, DEPOIS TODAS DE PROF. 2, ... ATÉ ENCONTRAR (OU NUNCA ENCONTRAR) UM ESTADO DE PARADA.

- DEPOIS DA PALAVRA β, ESCREVEMOS

$$
\begin{array}{l}
11, 12, 13, \ldots, 1\beta \\
21, 22, \ldots, 2\beta \\
\vdots \quad \ddots \\
\beta 1, \beta 2, \ldots, \beta\beta \\
111, \ldots
\end{array}
$$

- NOTE QUE SE HÁ UMA COMPUTAÇÃO NÃO-DETERMINÍSTICA COM m PASSOS, TEREMOS UMA COMPUTAÇÃO DETERMINÍSTICA COM ATÉ β^m PASSOS

Fechamento das linguagens Rec e RE

- Gostaríamos de saber se tais linguagens são fechadas com respeito às operações de complemento, união, interseção, diferença.

Rec)

1) Complemento: basta trocar s por n
2) União: Rodar em paralelo $Q_1 \times Q_2$
   ⇝ Criar uma máquina com duas fitas e aceita uma palavra se atinge aceitação em pelo menos uma máquina
3) Interseção: Rodar em paralelo $Q_1 \times Q_2$
   ⇝ Criar uma máquina com duas fitas e aceita uma palavra se atinge aceitação nas duas máquinas
4) $L_1 \setminus L_2 = L_1 \cap \overline{L_2}$ ✓

RE)

2 e 3) Igual a Rec

1 e 4) Vamos mostrar no futuro que RE não é fechado por complemento ou diferença

Teorema: Se L e $\overline{L}$ são RE, então L e $\overline{L}$ são Rec.

Prova: Podemos decidir L com uma máquina que roda as máquinas M_1 e M_2 que aceitam L e $\overline{L}$ em paralelo. Se w é aceita por M_1, aceitamos w, se é aceita por M_2, rejeitamos w.

TESE DE CHURCH-TURING

- M.T. É UM MODELO SIMPLES
  ↳ PODEMOS ESTUDAR O QUE PODE E
  O QUE NÃO PODE SER COMPUTADO

- MÚLTIPLAS FITAS, CABEÇOTES, NÃO-DETERMINISMO NÃO AUMENTAM SEU PODER COMPUTACIONAL
  ↳ OBVIAMENTE NÃO ESTAMOS FALANDO DE VELOCIDADE

- HÁ OUTROS MODELOS EQUIVALENTES

  → λ-CÁLCULO
  → FUNÇÕES μ-RECURSIVAS

- ADOTAMOS MÁQUINA DE TURING QUE DECIDE OU COMPUTA COMO MODELO TEÓRICO DE ALGORITMO

TESE DE CHURCH-TURING: UM PROBLEMA DE DECISÃO OU FUNÇÃO É ALGORITMICAMENTE COMPUTÁVEL SE E SOMENTE SE EXISTE UMA MÁQUINA DE TURING QUE O DECIDE OU COMPUTA, RESP.

→ NÃO PODE SER PROVADA
→ PODE SER REFUTADA: BASTA DESENVOLVER UM MODELO DE COMPUTAÇÃO MAIS PODEROSO QUE M.T.

→ IMPROVÁVEL: TODOS OS MODELOS DESENVOLVIDOS ATÉ HOJE SE MOSTRARAM EQUIVALENTES A M.T.

→ INCLUSIVE COMPUTAÇÃO QUÂNTICA

- PODEMOS ENTÃO PROVARMOS QUE ALGUNS PROBLEMAS NÃO PODEM SER RESOLVIDOS POR ALGORITMO

A Máquina de Turing Universal

- A M.T. Universal $\mathcal{U}$ é uma M.T. capaz de simular qualquer outra M.T.

- $\mathcal{U}$ tem que receber o programa da máquina a ser simulada.

- Temos que descrever o alfabeto, estados, e transições a partir do alfabeto de $\mathcal{U}$.

- $\mathcal{U}$ é um modelo de computador programável

OBS: Há várias formas de definir $\mathcal{U}$

- Alf entrada: $\Sigma_{\mathcal{U}} = \{0, 1, \sigma, q, X, Y, Z, \#, a, b\}$

- Alf fita: $\Sigma_{\mathcal{U}} \cup \{\triangleright, \sqcup\}$

- Queremos simular uma máquina M

- Transições de M serão enumeradas na fita de $\mathcal{U}$ separadamente

- Os símbolos e estados de M serão representados no sistema unário ($0^n = 0 \cdots 0$)

- As setas serão representadas $\leftarrow, \rightarrow$ por símbolos de M também

- Símbolos iniciam por σ e estados por q

- Seja M uma M.T. com n estados e m símbolos

→ Porque as setas são símbolos

→ Reservamos $m+3$ casas para cada símbolo, e $n+1$ para cada estado

Ex: r-ésimo símbolo $\sigma\, 0^r 1^{m+2-r}$

r-ésimo estado $q\, 0^r 1^{n-r}$

Convenção: $\triangleright, \leftarrow, \rightarrow$

são os primeiros símbolos

Símbolo	Código
$\triangleright$	$\sigma 0 1^{m+1}$
$\rightarrow$	$\sigma 0^1 1^m$
$\leftarrow$	$\sigma 0^2 1^{m-1}$

- Como codificar uma transição?

Ex: $\delta(q_i, \sigma_j) = (q_r, \sigma_s)$

$S =$ | X | $q 0^i 1^{n-i}$ | $\sigma 0^j 1^{m+2-j}$ | $q 0^r 1^{n-r}$ | $\sigma 0^s 1^{m+2-s}$ | X |

- A descrição completa é:

$$\triangleright X S_1 X S_2 \cdots X S_t Y Q Z \sigma u_1 \sigma u_2 \# u_3 \cdots$$

- Entre $\triangleright$ e Y : Transições de M

- Entre Y e Z : $Q = q 0^i 1^{m-i}$ estado no qual M se encontra no estágio atual da computação

- A partir de Z : w, a palavra atual, descrita nos símbolos de $\mathcal{U}$

Ex: Considere M com alf $\{0, \sqcup, \triangleright\}$, estados $\{q_0, q_1, h\}$, $F = \{h\}$, e transições

Estado	Entrada	Transições
q_0	0	$(q_1, \sqcup)$
	$\sqcup$	$(h, \sqcup)$
	$\triangleright$	$(q_0, \rightarrow)$
q_1	0	$(q_0, 0)$
	$\sqcup$	$(q_0, \rightarrow)$
	$\triangleright$	$(q_1, \rightarrow)$

Símbolos e estados

Estado	Código	Símbolo	Código
q_0	$q 0 1^2$	$\triangleright$	$\sigma 0 1^4$
q_1	$q 0^2 1$	$\rightarrow$	$\sigma 0^2 1^3$
h	$q 0^3$	$\leftarrow$	$\sigma 0^3 1^2$
		$\sqcup$	$\sigma 0^4 1$
		0	$\sigma 0^5$

- A primeira transição é codificada por $X q 0 \perp^2 \sigma 0^5 q 0^2 \perp \sigma 0^4 \perp X$

- O processamento de uma máquina de Turing tem duas partes
  1) verificar se a descrição é legítima
  2) simular a máquina cuja descrição foi dada

1) Primeiro varremos a fita, verificamos se X, Y, Z, q, σ estão na ordem certa, e se as representações têm o mesmo tamanho
   - → se alguma condição não é satisfeita, $\mathcal{U}$ caminha p/ direita sem parar
   - → Comparar tamanhos pode ser feito substituindo 0's e 1's por a's e b's, resp.

2) dividido em três partes

2.1) → cabeçote à direita de D
   - → varre entre D e Y buscando uma quádrupla que comece com a representação do par (Q, t), onde
     - Q é o estado representado entre Y e Z; e
     - τ é o símbolo representado à esq. de #
   - → substituímos 0's e 1's por a's e b's
   - → se não encontrar tal quádrupla, então estamos num estado final.
     - ↳ e aí $\mathcal{U}$ vai p/ estado final também

2.2) O estado $\overset{Q}{q 0^i 1^{n-i}}$ e o símbolo $\sigma 0^j 1^{m+2-j} = t$ mais à esq. correspondem à ação de M

   - → Q e τ devem ser copiados para os campos correspondentes entre Y e Z, e logo após #, resp.
   - → Fazemos isso trocando 0's e 1's por a's e b's para saber quais símbolos foram trocados
   - → se τ representar uma seta, $\mathcal{U}$ deve mover # de posição

SÍMBOLO	AÇÃO
$\sigma 0^{1} 1^{m+1}$: $\longrightarrow$	MOVE # PARA O LUGAR DO PRÓXIMO σ
$\sigma 0^{2} 1^{m}$: $\longleftarrow$	MOVE # PARA O LUGAR DO σ ANTERIOR
$\sigma 0^{J} 1^{m+2-J}$ $J \geqslant 1$	SUBSTITUI O SEG APÓS #

2.3) REBOBINA SUBTITUINDO a's E b's POR 0's E 1's.