A MÁQUINA DE TURING UNIVERSAL ⇝ $\mathcal{U}$

- CAPAZ DE SIMULAR QUALQUER MÁQUINA DE TURING
- TEM QUE RECEBER O "PROGRAMA" DESSA MÁQUINA
- TEMOS QUE DESCREVER OS ALFABETOS, ESTADOS, E TRANSIÇÕES
- $\mathcal{U}$ É UM MODELO DE COMPUTADOR PROGRAMÁVEL

OBS: HÁ VÁRIAS FORMAS DE DEFINIR $\mathcal{U}$

- ALF. DE ENTRADA $\sum_{\mathcal{U}} = \{0, 1, \sigma, q, X, Y, Z, \#, a, b\}$
- ALF. DA FITA $\sum_{\mathcal{U}} \cup \{\triangleright, \sqcup\}$

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

- QUEREMOS SIMULAR A MÁQUINA M.
  - TRANSIÇÕES DE M SERÃO ENUMERADAS NA FITA DE $\mathcal{U}$ SEPARADAMENTE
  - OS SÍMBOLOS DE M SERÃO REPRESENTADOS NO SISTEMA UNÁRIO.
  - AS SETAS DE M SERÃO SÍMBOLOS DE M
  - SÍMBOLOS INICIAM POR σ E ESTADOS POR q

- SUPONHA QUE M TEM m SÍMBOLOS E n ESTADOS
  - RESERVAMOS $m+3$ CASAS P/ CADA SÍMBOLO: $\sigma\, 0^{r}\, 1^{m+2-r}$ (r-ÉSIMO SÍMBOLO)
  - " $n+1$ CASAS P/ CADA ESTADO : $q\, 0^{r}\, 1^{n-r}$

SÍMBOLO	CÓDIGO
▷	$\sigma 01 \cdots 1$
→	$\sigma 001 \cdots 1$
←	$\sigma 0001 \cdots 1$

- CODIFICAR TRANSIÇÃO

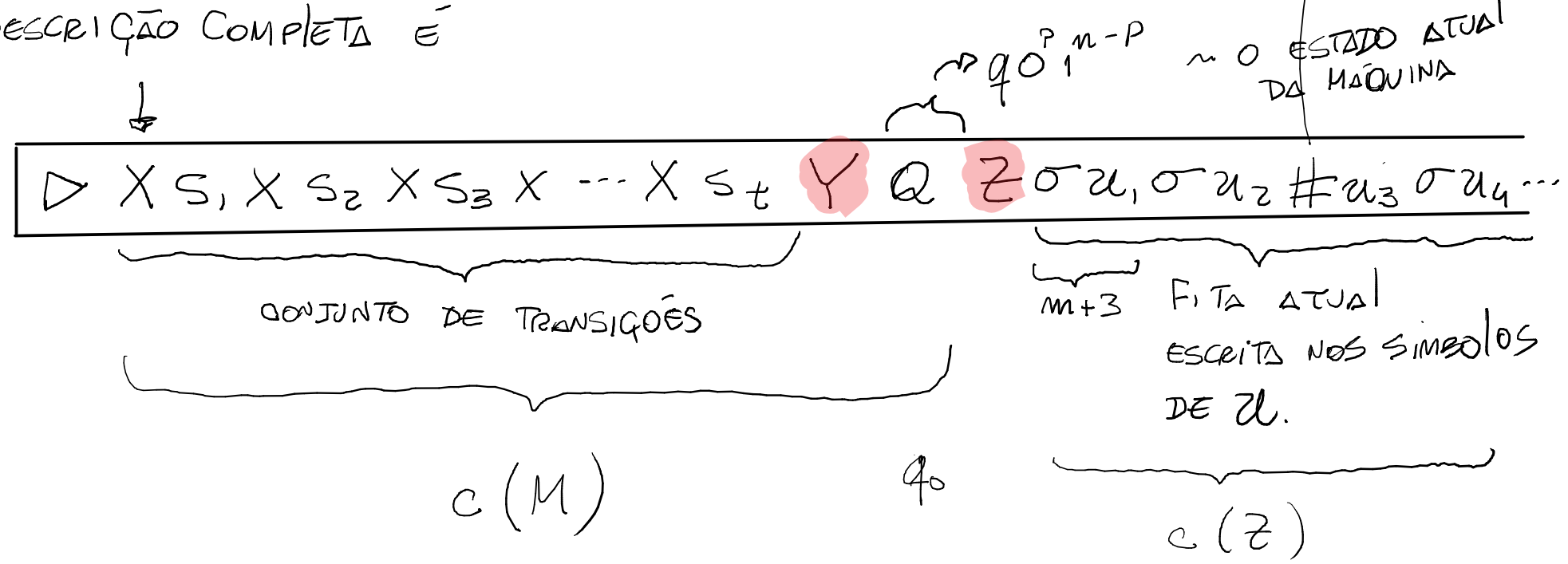
$\Sigma_u = \{0, 1, \sigma, q, X, Y, Z, \#, a, b\}$

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

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

$n+1 \quad m+3 \quad n+1 \quad m+3$

- A DESCRIÇÃO COMPLETA É



EX: SEJA M COM ALFABETO $\sum = \{0, \triangleright, \sqcup\}$, ESTADOS $\{q_0, q_1, h\}$ $F = \{h\}$

ESTADO	ENTRADA	TRANSIÇÕES
q_0	0	$(q_1, \sqcup)$
	$\sqcup$	$(h, \sqcup)$
	$\triangleright$	$(q_0, \rightarrow)$
q_1	0	$(q_0, 0)$
	$\sqcup$	$(q_0, \rightarrow)$
	$\triangleright$	$(q_1, \rightarrow)$

ESTADO	CÓDIGO	SÍMBOLO	CÓDIGO
q_0	$q011$	$\triangleright$	$\sigma01111$
		$\rightarrow$	$\sigma00111$
q_1	$q001$	$\leftarrow$	$\sigma00011$
h	$q000$	$\sqcup$	$\sigma00001$
		0	$\sigma00000$

$\delta(q_0, 0) = (q_1, \sqcup)$

$X\, q011\, \sigma00000\, q001\, \sigma00001\, X$

PROCESSAMENTO : DUAS PARTES

$\sum_{TM} = \{0, 1, \sigma, q, X, Y, Z, \#, a, b\}$

1) VERIFICAR SE A DESCRIÇÃO É LEGÍTIMA

- SE X's, Y, Z, q's E σ's ESTÃO NA ORDEM CORRETA
- SE AS REPRESENTAÇÕES TÊM O MESMO TAMANHO

2) SIMULAR A MÁQUINA CUJA DESCRIÇÃO FOI DADA

TRÊS ETAPAS:

2.1) CABEÇOTE À DIREITA ▷

VARREMOS ENTRE ▷ E Y BUSCANDO UMA QUÁDRUPLA QUE COMECE COM A REP. DO PAR (q, t), ONDE

q É O ESTADO REPRESENTADO ENTRE Y E Z; E
t É O ESTADO REPRESENTADO À ESQ. DE #

SUBSTITUÍMOS 0's E 1's POR a's E b's, RESP.

SE NÃO ENCONTRARMOS TAL QUÁDRUPLA, ESTAMOS EM UM ESTADO FINAL

2.2) O ESTADO Q E SÍMBOLO $\mathcal{U}$ MAIS À ESQ. CORRESPONDEM À AÇÃO DE M

NO Q E $\mathcal{U}$ DEVEM SER COPIADOS PARA OS CAMPOS CORRESPONDENTES

ENTRE Y E Z, DEPOIS DE #

SE $\mathcal{U}$ REPRESENTA ← OU → MOVEMOS A POSIÇÃO DE #

$\sigma 0^1 1^{m+1}$: → MOVE # PARA O LUGAR DO PRÓXIMO σ

$\sigma 0^2 1^m$: ← MOVE # PARA O LUGAR DO σ ANTERIOR

$\sigma 0^J 1^{m+2-J}$ SUBSTITUI O SEG APÓS # POR $\sigma 0^J 1^{m+2-J}$

$J \geq 2$

2.3) REBOBINA A FITA SUBSTITUINDO a'S E b'S POR 0'S E 1'S.

O Problema de Parada

Def: Dada uma M.T. M, denotamos por $c(M)$ o "programa" de M para a M.T. universal $\mathcal{U}$.

- Considere a linguagem → Máq. de Turing com A/F. de entrada $\Sigma_{\mathcal{U}}$

$$H = \{ c(M) \# w : \text{a máquina } M \text{ para com entrada } w \in \Sigma_{\mathcal{U}}^* \} \subseteq \Sigma_{\mathcal{U}}^*$$

- Seja $T_{\Sigma_{\mathcal{U}}}$ uma máquina que lê w e retorna $c(w)$

- Considere a máquina $T_{\Sigma_{\mathcal{U}}}$ que lê $\triangleright c(M) \# w$ e retorna $\triangleright c(M) \# c(w)$

- H é recursivamente enumerável: a máquina $M_H = T_{\Sigma_{\mathcal{U}}} \to \mathcal{U}$

é uma máquina que aceita H.

- Considere a linguagem

$$H' = \{ c(M) : \text{a M.T. } M \text{ para com entrada } c(M) \in \Sigma^*_u \}$$

$$\rightsquigarrow M \text{ t.q. } c(M) \in L_{aceita}(M)$$

Prop: H' é R.E.

Prova: Considere a máquina C que transforma $\triangleright c(M)$ em $\triangleright c(M) \# c(M)$

e seja $M_{H'}$ a máquina $C \longrightarrow M_H$

se $c(M) \in H'$, então $M_{H'}$ aceita $c(M)$

- Considere $\overline{H'} \subseteq \sum_{2a}^{*}$

  1) $w = c(M)$ e M não para com entrada $c(M)$

  2) $w = c(M)$ e M não processa palavras em $\sum_{a}^{*}$

  3) w não é um programa de M.T.

Pergunta $\overline{H'}$ é R.E?

TEOREMA: $\overline{H'}$ NÃO É R.E.

PROVA: SUPONHA QUE $\overline{H'}$ É R.E. E SEJA N UMA M.T. DE ACEITA $\overline{H'}$

PERGUNTA: $C(N) \in H'$ OU $C(N) \in \overline{H'}$?

CASO 1: $C(N) \in H'$. PELA DEF. DE H', ENTÃO N PARA COM $C(N)$

ISSO IMPLICA QUE $C(N)$ É ACEITA POR N, PORTANTO

$C(N) \in L_{aceita}(N) = \overline{H'}$, UMA CONTRADIÇÃO

CASO 2: $C(N) \in \overline{H'}$: $C(N)$ É DO TIPO 1,

LOGO N É UMA M.T. QUE NÃO PARA COM $C(N)$

$C(N) \notin L_{aceita}(N) = \overline{H'}$

PORTANTO, NÃO HÁ M.T. QUE ACEITA $\overline{H'}$

NÃO EXISTE M.T. QUE ACEITE $\overline{H'}$

COROLÁRIO: H' NÃO É RE

COROLÁRIO: R.E. NÃO É FECHADO POR COMPLEMENTO

PROBLEMA DE PARADA: FIXE UM ALFABETO Σ.

DADA M.T. M COM ALF. Σ E $w \in \Sigma^*$

EXISTE M.T. U^* QUE TENDO $c(M) \# w$ COMO ENTRADA,

DECIDE SE M PARA COM ENTRADA w?

RESP: NÃO

PROVA: SEJA L (→ PODE SER H') UMA LINGUAGEM R.E. E M UMA M.T. QUE ACEITA L

SEJA A_M QUE TRANSFORMA $\triangleright w$ EM $\triangleright c(M) \# w$.

ENTÃO $A_M \to U^*$ DECIDE L.

~> NEM TODA QUESTÃO MATEMÁTICA PODE SER RESOLVIDA ALGORITMICAMENTE.

- HÁ OUTROS PROBLEMAS INDECÍDÍVEIS
  1) DECIDIR SE A INTERSEÇÃO DE DUAS LLC'S É REGULAR
  2) DECIDIR SE UMA LLC É REGULAR
  3) DECIDIR SE UMA LLC É INERENTEMENTE AMBÍGUA.

REDUÇÕES

- MÉTODO PARA PROVAR QUE UMA LINGUAGEM É INDECIDÍVEL

DEF: UMA LINGUAGEM L_1 É REDUTÍVEL A L_2 SE EXISTE FUNÇÃO

COMPUTÁVEL $\psi : \sum_0^* \longrightarrow \sum_0^*$ T.q. $w \in L_1$ SE E SÓ SE $\psi(w) \in L_2$

NOTAÇÃO: $L_1 \leq_{M_\psi} L_2$

- SE $L_1 \leq_{M_\psi} L_2$ E L_1 NÃO É REC, ENTÃO L_2 NÃO É REC

CASO CONTRÁRIO $M_\psi \longrightarrow M_{L_2}$

TEORIA DA COMPLEXIDADE

ℓ BITS

- FATORIZAÇÃO DE INTEIROS $n = a \cdot b$ $\quad 2^{\ell} \leq n \leq 2^{\ell+1}$

- HÁ MUITOS PROBLEMAS DECIDÍVEIS PARA OS QUAIS É NECESSÁRIO MUITO TEMPO PARA RESOLVER.

  EX: COLORAÇÃO DE GRAFOS, FATORIZAÇÃO DE INTEIROS

- AQUI TEREMOS PROBLEMAS TRATÁVEIS E INTRATÁVEIS.

Complexidade de Tempo

Def: Uma M.T. é **polinomialmente limitada** se existe função polinomial $P(n)$ ($\leq An^r$) t.q. para toda entrada $w \in \Sigma_0^*$, qualquer computação a partir do estado inicial alcança um estado final em no máximo $P(|w|)$ passos.

Def: A **classe P** é a família das linguagens que podem ser decididas por uma M.T. **determinística** Pol. limitada.

Em geral, dizemos que um problema é **tratável** se está em P.

Def: A **classe NP** é a família das linguagens que podem ser decididas por uma M.T. **não-determinística** Pol. limitada.

Ex: As codificações dos números compostos

Teorema: $P \subseteq NP$

Prova: Toda M.T. det. é não-det

$P = NP$?

Complexidade de Tempo 2

Def: Uma M.T. é **exponencialmente limitada** se existe const. C e função polinomial $P(n)$ t.q. para toda entrada $w \in \Sigma_0^*$, qualquer computação a partir do estado inicial alcança um estado final em no máximo $C^{P(|w|)}$ passos.

Def: A **classe EXPTIME** é a família das linguagens que podem ser decididas por uma M.T. **determinística** exp. limitada.

Def: A **classe NEXPTIME** é a família das linguagens que podem ser decididas por uma M.T. **não-determinística** exp. limitada.

Teorema: $EXPTIME \subseteq NEXPTIME$

Prova: Toda M.T. det. é não-det

Teorema: $NP \subseteq EXPTIME$

Prova: Basta notar que a máquina construída p/ simular M.T. não-det. é exp.

$$P \subseteq NP \subseteq EXPTIME \subseteq NEXPTIME$$

Complexidade de espaço

Def: Uma M.T. é **espacialmente pol. lim** se há um polinômio $p(n)$ t.q para toda entrada $w \in \Sigma_0^*$, qualquer computação a partir do estado inicial sempre alcança um estado de parada tendo visitado no máximo $p(|w|)$ casas distintas.

Def: A classe PSPACE é a família de linguagens que podem ser decididas por uma M.T. **determinística** espacialmente pol. limitada

Def: A classe NPSPACE é a família de linguagens que podem ser decididas por uma M.T. **não-determinística** espacialmente pol. limitada

Obs: Não sabemos se $P = NP$ ou $EXPTIME = NEXPTIME$

Teorema de Savitch: $PSPACE = NPSPACE$

TEOREMA: $P \subseteq PSPACE$

PROVA: EM CADA PASSO PODEMOS VISITAR NO MÁXIMO UMA NOVA CASA.

- COM O MESMO ARGUMENTO, TEMOS $NP \subseteq NPSPACE$

TEOREMA: $NP \subseteq PSPACE$

TEOREMA: $PSPACE \subseteq EXPTIME$

PROVA: SUPONHA QUE USAMOS $P(|w|)$ CASAS. E $|\Sigma_0| = m$

HÁ NO MÁXIMO $r = |Q| \cdot p(|w|) \cdot m^{P(|w|)}$ CONFIGURAÇÕES DISTINTAS

ENTÃO A MÁQUINA NÃO REALIZA MAIS DO QUE r PASSOS.

$|Q| \leq m^{|Q|}$

$P(|w|) \leq m^{P(|w|)}$

$r \leq m^{|Q| + 2P(|w|)}$

Co-classes

DEF: SE C É UMA CLASSE DE COMPLEXIDADE, co-C É A FAMÍLIA

$$Co\text{-}C = \{L : \bar{L} \in C\}$$

TEMOS

1) $P = Co\text{-}P$
2) $PSPACE = Co\text{-}PSPACE = NPSPACE = Co\text{-}NPSPACE$
3) $EXPTIME = Co\text{-}EXPTIME$

PERGUNTA: $Co\text{-}NP = NP$? $Co\text{-}NEXPTIME = NEXPTIME$?

LEMA: SE $C_1 \subseteq C_2$, ENTÃO $Co\text{-}C_1 \subseteq Co\text{-}C_2$

PROVA: SE $L \in Co\text{-}C_1$, ENTÃO $\bar{L} \in C_1$, LOGO $\bar{L} \in C_2$, PORTANTO $L \in Co\text{-}C_2$

$\Rightarrow Co\text{-}P \subseteq Co\text{-}NP$

$$P \subseteq NP \subseteq \begin{matrix} \subseteq C_0\text{-}NP \\ \subseteq \end{matrix} PSPACE \subseteq EXPTIME \begin{matrix} \subseteq NEXPTIME \\ \subseteq C_0\text{-}NEXPTIME \end{matrix}$$

Reduções Polinomiais

Def: Uma função é **Pol. Computável** se existe M.T. determinística Pol. lim. que a computa

Def: Dadas $L_1, L_2 \subseteq \Sigma_0^*$, dizemos que existe **redução polinomial** de L_1 p/ L_2, denotada $L_1 \leq_p L_2$, se existe uma função Pol. Computável $f: \Sigma_0^* \to \Sigma_0^*$ t.q. $w \in L_1$ se e só se $f(w) \in L_2$

Prop: Se C é uma classe, $L_2 \in C$ e $L_1 \leq_p L_2$, então $L_1 \in C$

Def: Uma linguagem L é **C-difícil** se para toda $L' \in C$, temos $L' \leq_p L$.

Def: Uma linguagem L é **C-completa** se

1) L é C-difícil
2) $L \in C$

LEMA: SE L É C-COMPLETA E $L \leq_p L'$, ENTÃO L' É C-COMPLETA

TEOREMA DE COOK-LEVIN: SAT É NP-COMPLETA.