Decisor, Aceitador, Recursivo, Recursivamente Enumerável

Def: Um decisor é uma máquina de Turing $M = (\Sigma_0, \Sigma, Q, q_0, F, \delta)$ t.q.

1) M possui dois estados finais (de parada): um de aceitação e um de rejeição: S (SIM) e N (NÃO).

$$F = \{S, N\}$$

2) M sempre para (alcança estado final) com qualquer palavra de Σ_0^* colocada como entrada no início da fita.

- Um decisor M com alf. de entrada Σ_0 decide uma linguagem $L \subseteq \Sigma_0^*$ (ou, equiv. $L \subseteq \Sigma_0^*$ é decidida por M) se

$$w \in L \iff M \text{ para no estado } S \text{ com } w$$

- Como M é decisor, isso implica que

$$w \notin L \iff M \text{ para no estado } N \text{ com } w$$

Rec

Def: Uma linguagem L é recursiva ou decidível se existe decisor que decida L.

- Toda linguagem regular ou livre de contexto é recursiva.

- Como construir um decisor para decidir uma LR ou LLC.

DECISORES PARA LR

- SEJA $A = (\Sigma_f, Q_f, q_{0f}, F_f, \delta_f)$ UM AUTÔMATO FINITO.

- VAMOS CONSTRUIR UMA MÁQUINA DE TURING $M = (\Sigma_0, \Sigma_1, Q, q_0, F, \delta)$

  - $\Sigma_0 = \Sigma_f$
  - $\Sigma_1 = \Sigma_0 \cup \{\triangleright, \sqcup, X\}$, ONDE $X \notin \Sigma_f$
  - $Q = Q_f \cup \{S, N\}$, $S, N \notin Q_f$
  - $q_0 = q_{0f}$
  - $F = \{S, N\}$
  - δ

$$\delta(q, \sigma) = (\delta_f(q, \sigma), X), \text{ SE } \sigma \notin \{\triangleright, \sqcup, X\}$$

$$\delta(q, X) = \delta(q, \triangleright) = (q, \rightarrow)$$

$$\delta(q, \sqcup) = (S, \sqcup), \text{ SE } q \in F_f$$

$$\delta(q, \sqcup) = (N, \sqcup), \text{ SE } q \notin F_f$$

- COLOCAMOS $w \in \Sigma_0^*$ NO COMEÇO DA FITA (APÓS $\triangleright$), E O CABEÇOTE NO PRIMEIRO SÍMBOLO DE w.

- X É USADO PARA INDICAR QUE AQUELE SÍMBOLO JÁ FOI CONSUMIDO

- TAMBÉM É POSSÍVEL SIMULAR UM AUTÔMATO DE PILHA POR MÁQUINAS DE TURING (VEREMOS EM OUTRA AULA)

DEF: DIZEMOS QUE UMA MÁQUINA M ACEITA UMA LINGUAGEM $L \subseteq \Sigma_0^*$ SE, $\forall w \in \Sigma_0^*$ TEMOS QUE

$$w \in L \iff M \text{ PARA (ALCANÇA ESTADO FINAL) COM ENTRADA } w$$

ISSO IMPLICA QUE

$$w \notin L \iff M \text{ NÃO PARA (NUNCA ALCANÇA ESTADO FINAL) COM ENTRADA } w$$

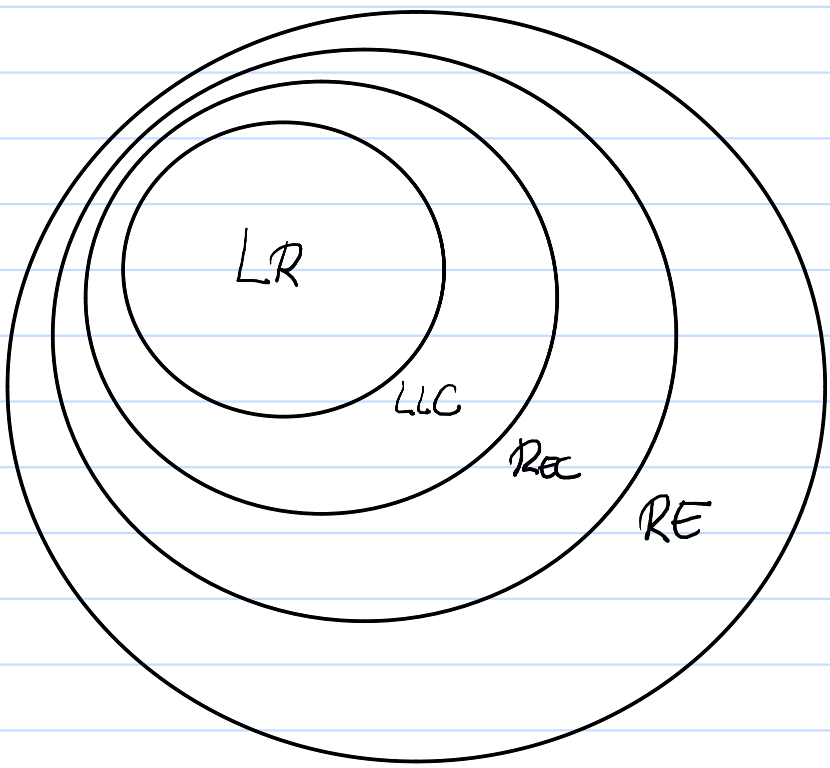
RE

DEF: DIZEMOS QUE UMA LINGUAGEM L É UMA LINGUAGEM RECURSIVAMENTE ENUMERÁVEL OU UMA LINGUAGEM SEMI-DECIDÍVEL SE EXISTE UMA MÁQUINA DE TURING QUE ACEITA L.

- TODA LINGUAGEM DECIDÍVEL É SEMI-DECIDÍVEL: BASTA MODIFICAR AS TRANSIÇÕES QUE LEVAM A N, E FAZER COM QUE CICLEM NO ESTADO ANTERIOR.

- EXISTEM LINGUAGENS RE QUE NÃO SÃO REC

HIERARQUIA DE CHOMSKY



- Podemos usar máquinas de Turing para resolver problemas que não são de decisão

- Seja $f : \Sigma_0^* \rightarrow \Sigma_0$ uma função

- Dizemos que uma MT M computa f se $\forall\, w \in \Sigma_0^*$, quando M recebe w no início (logo após $\triangleright$) e inicia com o cabeçote na primeira casa, ela termina parando com $f(w)$ no início da fita

- Dizemos que f é uma função computável se existe MT que computa f.

Diagramas de Composição

- Escrever todas as transições pode ser trabalhoso

- Usaremos diagramas:
  1) Máquinas básicas
  2) Regras de composição

- 1) Máquinas básicas

  D = Move cabeçote para direita e para

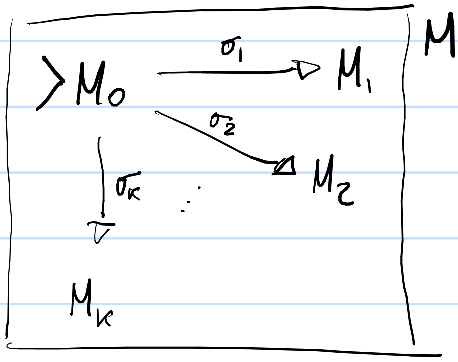
  E = Move cabeçote para esquerda e para

  W_a = Escreve a

  P = Para sem realizar ação

  S = Para no estado de aceitação

  N = Para no estado de rejeição



- $>$ indica que a máquina começa em M_0
  ↳ enquanto M_0 não parar, M executa de forma idêntica a M_0

- Quando M_0 para, M verifica o símbolo na fita e faz a transição para o estado correspondente.
  ↳ se o símbolo lido não está presente, M para.

- Permite que escrevamos uma ML "por partes"

(1) $\begin{tikzcd} M_1 \arrow[r, bend left, "a"] \arrow[r, bend right, "b"'] & M_2 \end{tikzcd}$ pode ser simplificado p/ $M_1 \xrightarrow{a,b} M_2$

(2) $M_1 \longrightarrow M_2$ indica que executamos M_2 imediatamente após M_1
pode ser simplificado para M_1M_2

(3) $M_1 \xrightarrow{a \neq \sigma} M_2$ vai para M_2 quando lê símbolo diferente de σ
pode ser simplificado para $M_1 \xrightarrow{\bar{\sigma}} M_2$

- E_σ = move p/ esq. até encontrar σ : $\begin{tikzcd} E \arrow[loop right, "\bar{\sigma}"] \end{tikzcd}$

- D_σ = move p/ dir até encontrar σ : $\begin{tikzcd} D \arrow[loop right, "\bar{\sigma}"] \end{tikzcd}$

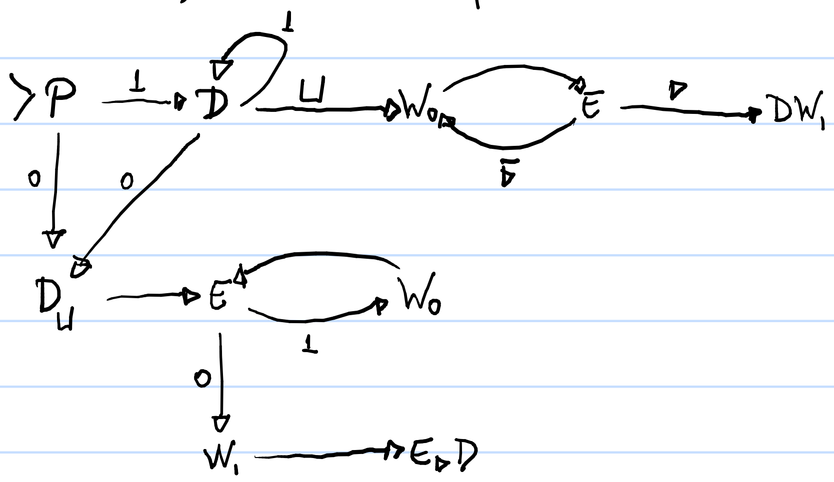
- $E_{\bar{\sigma}}$ = move p/ esq. até encontrar símbolo diferente de σ : $\begin{tikzcd} E \arrow[loop right, "\sigma"] \end{tikzcd}$

- $D_{\bar{\sigma}}$ = move p/ dir até encontrar símbolo diferente de σ : $\begin{tikzcd} D \arrow[loop right, "\sigma"] \end{tikzcd}$

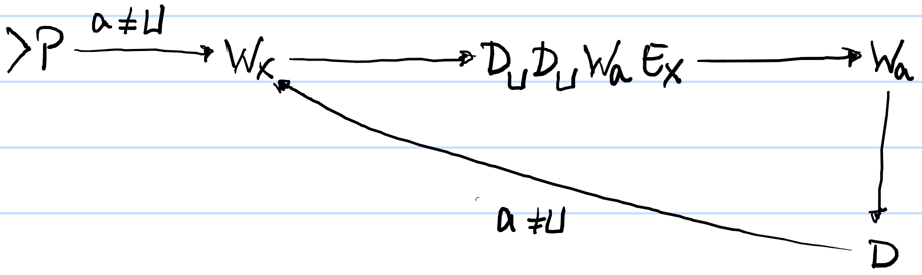
Exemplo: $f(x) = x+1$

- Varre o número da direita para esquerda trocando 1 por 0. Para quando encontra o primeiro 0 e troca por 1

- Ou, quando $w = 1 \cdots 1$, escreve $f(w) = 10 \cdots 0$



Ex: Máquina que copia: $f(w) = w \sqcup w$



Ex: Máquina que faz shift