EQUIVALÊNCIA ENTRE A.P. E GLC

TEOREMA: UMA LINGUAGEM É LIVRE DE CONTEXTO SE E SOMENTE SE É ACEITA POR UM AUTÔMATO DE PILHA.

- VAMOS CONSTRUIR UM A.P. QUE ACEITA A LINGUAGEM GERADA POR UMA GLC DADO; E
- VAMOS CONSTRUIR UMA GLC QUE GERA A LINGUAGEM QUE É ACEITA POR UM A.P. DADO

1) GLC → AP

- SIMULA AS DERIVAÇÕES MAIS À ESQ.

SEJA $G = (T, V, S, R)$

COMO CONSTRUO O AUTÔMATO $M = (\Sigma, \Gamma, Q, q_0, F, \Delta)$?

- $\Sigma = T$
- $\Gamma = T \cup V$

1) GLC → AP

- Simula as derivações mais à esq.

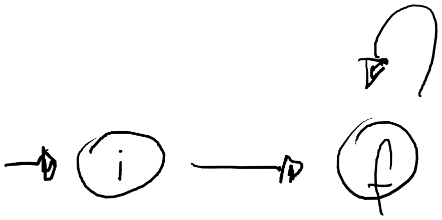
Seja $G = (T, V, S, R)$

Como construo o autômato $M = (\Sigma, \Gamma, Q, q_0, F, \Delta)$?

→ $\Sigma = T$

→ $\Gamma = T \cup V$

→ $Q = \{i, f\}$, $q_0 = i$, $F = \{f\}$



- $\Delta(i, \varepsilon, \varepsilon) = \{(f, S)\}$

- Se temos a regra $X \to \alpha$, criamos a transição

$$(f, \alpha) \in \Delta(f, \varepsilon, X)$$

→ símbolo lido na entrada

→ símbolo no topo da pilha

→ colocar na pilha

→ estado para onde vou

→ estado atual

EX: $S \rightarrow Sc \mid aSb \mid \varepsilon$

$\Delta =$

EST	ENTRADA	PILHA	TRANS.
i	ε	ε	(f, S)
f	ε	S	(f, Sc) (f, aSb) (f, ε)

EST	ENTRADA	PILHA	TRANS.
f	a	a	(f, ε)
f	b	b	(f, ε)
f	c	c	(f, ε)

abc^2 É DERIVADA POR $S \Rightarrow Sc \Rightarrow Sc^2 \Rightarrow aSbc^2 \Rightarrow abc^2$

A COMPUTAÇÃO FICA

$(i, abc^2, \varepsilon) \vdash (f, abc^2, S) \vdash (f, abc^2, Sc)$

$\vdash (f, abc^2, Sc^2)$

$\vdash (f, abc^2, aSbc^2)$

PRECISAMOS COMEÇAR → A COMPARAR

$\vdash (f, bc^2, Sbc^2)$

$\vdash (f, bc^2, bc^2) \vdash^* (f, \varepsilon, \varepsilon)$

MAIS FORMALMENTE

PRECISAMOS PROVAR QUE ESTE AUTÔMATO ACEITA TODAS AS PALAVRAS GERADAS POR G.

$\Sigma = T$

$\Gamma = T \cup V$

$Q = \{i, f\}$

$q_0 = i$

$F = \{f\}$

$$\Delta : \{i, f\} \times (T \cup \{\varepsilon\}) \times (T \cup V \cup \{\varepsilon\}) \longrightarrow \{i, f\} \times (T \cup V)^*$$

$$\Delta(q, \sigma, \gamma) = \begin{cases} (f, S) & \text{se } q = i \text{ e } \sigma = \gamma = \varepsilon \\ (f, u) : X \rightarrow u \in R & \text{se } q = f, \sigma = \varepsilon \text{ e } \gamma = X \in V \\ (f, \varepsilon) & \text{se } q = f \quad \sigma = \gamma \in T \end{cases}$$

PROVA: DADA $w \in L(G)$, VAMOS MOSTRAR QUE EXISTE COMPUTAÇÃO $(i, w, \varepsilon) \vdash^* (f, \varepsilon, \varepsilon)$ DE M.

- SABEMOS QUE HÁ DERIVAÇÃO $S \Rightarrow^* w$

SUPONHA QUE APÓS ALGUMAS ETAPAS DA DERIVAÇÃO

$$S \Rightarrow \cdots \Rightarrow \alpha X v \xRightarrow{X \to u} \alpha u v = \alpha \alpha' Y v' \Rightarrow w = \alpha \alpha' w''$$

ONDE $\alpha \in T^*$ E X É A VARIÁVEL MAIS À ESQ.

ISSO QUER DIZER QUE $w = \alpha w'$ PARA ALGUM $w' \in T^*$

OBS: EM CADA PASSO DA DERIVAÇÃO SUBSTITUÍMOS UMA VARIÁVEL $X \in V$ POR UMA PALAVRA $u \in (T \cup V)^*$ USANDO A REGRA $X \to u \in R$

ISSO CORRESPONDE A UMA SEQUÊNCIA NA COMPUTAÇÃO

NOSSA COMPUTAÇÃO TERÁ O TRECHO

$$(i, w, \varepsilon) \Rightarrow (f, w, S) \Rightarrow^* (f, \alpha w', \alpha X v) \Rightarrow^* (f, w', X v) \Rightarrow (f, w', u v) \Rightarrow^* (f, w'', Y v')$$

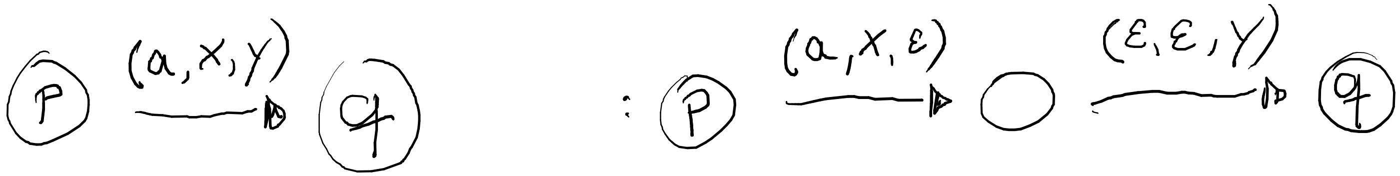
2) AP → GLC

- AUTÔMATO CORDATO :

  1) POSSUI APENAS UM ÚNICO ESTADO FINAL

  3) CADA TRANSIÇÃO OU COLOCA OU REMOVE ALGO DA PILHA, MAS NÃO AMBOS.

     ↳ PODEMOS FAZER ISSO DIVIDINDO CADA TRANSIÇÃO E ADICIONANDO UM ESTADO INTERMEDIÁRIO



PARA PAR DE ESTADOS p,q CRIAMOS UMA VARIÁVEL $A_{p,q}$ QUE GERA TODAS AS PALAVRAS QUE LEVAM M DE p A q INICIANDO E ACABANDO COM PILHA VAZIA

- A PRIMEIRA OPERAÇÃO TEM QUE SER DE COLOCAR X NA PILHA
- A ÚLTIMA OPERAÇÃO É DE REMOÇÃO

HÁ DUAS POSSIBILIDADES

1) O SÍMBOLO COLOCADO NO INÍCIO SOMENTE É REMOVIDO AO FINAL

REGRA: $A_{pq} \to a A_{rs} b$, ONDE a É O SÍMBOLO LIDO AO INCLUIR X
E b É O SÍMBOLO LIDO AO REMOVER X
r É O ESTADO APÓS p, s É O ESTADO ANTES DE q.

2) O SÍMBOLO É REMOVIDO AO FINAL É DIFERENTE DO INICIAL
MA O SÍMBOLO INICIAL É REMOVIDO EM ALGUM MOMENTO INTERMEDIÁRIO

REGRA: $A_{pq} \to A_{pr} A_{rq}$ ONDE r É O ESTADO NO QUAL A PILHA FICA VAZIA.

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 s
e um de rejeição n

$F = \{s, n\}$

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

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

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

Na fita no início da computação e com o cabeçote na casa à direita de $\triangleright$

- Como M é decisor, isso implica que

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

DEF: UMA LINGUAGEM $L \subseteq \Sigma_0$ É DECIDÍVEL (RECURSIVA - REC) SE EXISTE
EXISTE UM DECISOR QUE DECIDE L.

MA TODA LINGUAGEM REGULAR É DECIDÍVEL

MA TODA LINGUAGEM LIVRE-DE-CONTEXTO É DECIDÍVEL

DECISOR PARA LR

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

- $\Sigma_0 = \Sigma_f$
- $\Sigma = \Sigma_f \cup \{\triangleright, \sqcup, X\}$ ONDE $X \notin \Sigma_f$
- $Q = Q_f \cup \{s, n\}$ $s, n \notin Q_f$
- $q_0 = q_{0f}$
- $F = \{s, n\}$

$$\delta = \begin{cases} \delta(q, \sigma) = (\delta_f(q, \sigma), X) & \text{SE } \sigma \notin \{\triangleright, \sqcup, X\} \\ \delta(q, X) = \delta(q, \triangleright) = (q, \rightarrow) & \\ \delta(q, \sqcup) = (s, \sqcup) & \text{SE } q \in F_f \\ \delta(q, \sqcup) = (n, \sqcup) & \text{SE } q \notin F_f \end{cases}$$

DEF: DIZEMOS QUE UMA MÁQUINA M ACEITA UMA LINGUAGEM $L \subseteq \Sigma_0$

SE, $\forall w \in \Sigma_0^*$ TEMOS QUE

$w \in L \iff M$ PARA (ALCANÇA ESTADO FINAL)

COM w NO COMEÇO DA FITA

ISSO IMPLICA QUE

$w \notin L \iff M$ NÃO PARA (NUNCA ALCANÇA ESTADO FINAL)

COM w NO COMEÇO DA FITA

NESTE CASO, DIZEMOS QUE M É UM ACEITADOR.

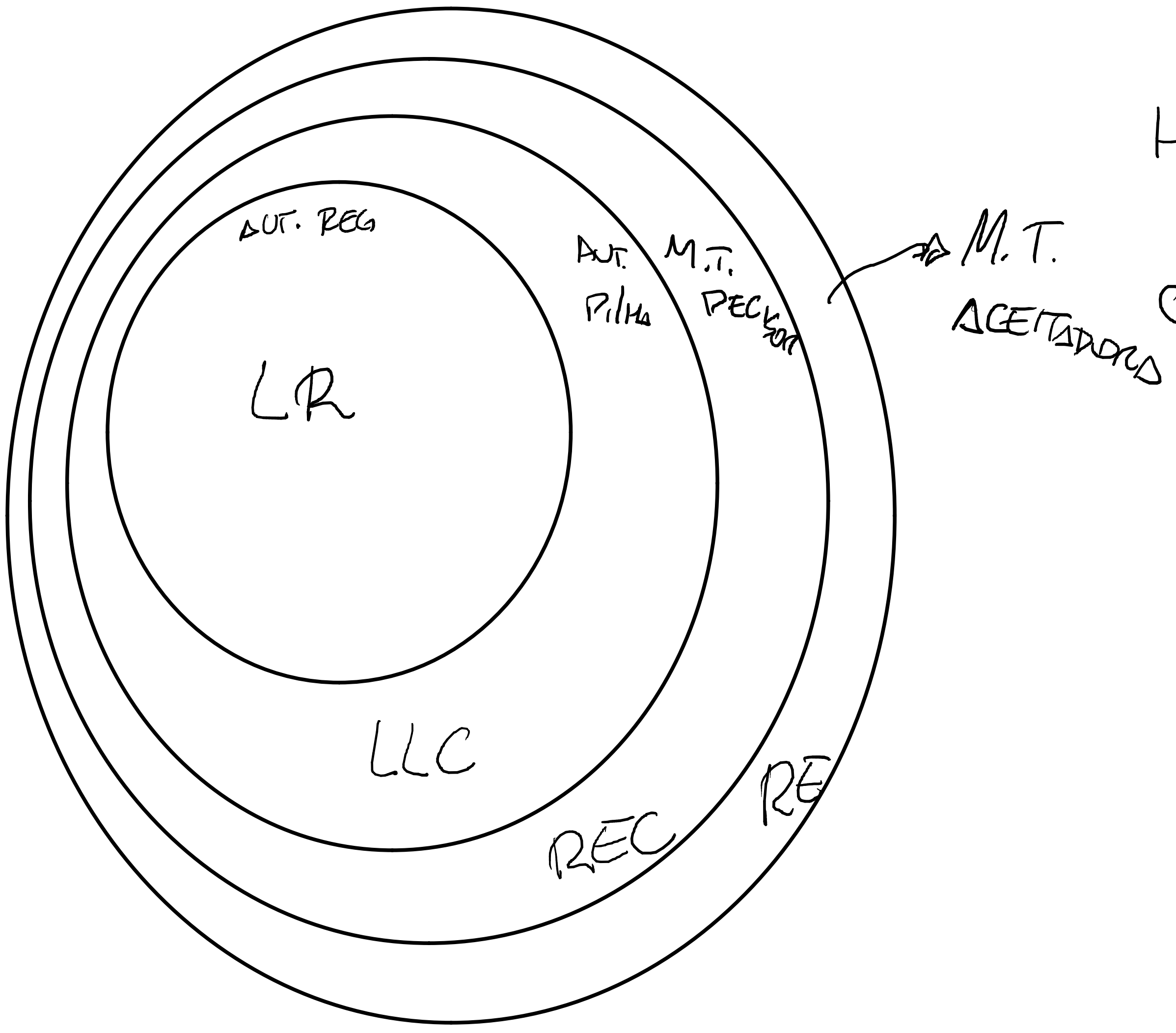
(RE)

DEF: DIZEMOS QUE UMA LINGUAGEM L É UMA LINGUAGEM RECURSIVAMENTE ENUMERÁVEL OU UMA LINGUAGEM SEMI-DECIDÍVEL SE EXISTE UMA M.T. QUE ACEITA L.

⇝ TODA LINGUAGEM DECIDÍVEL É SEMI-DECIDÍVEL (REC ⊆ RE)

↳ BASTA MODIFICAR O ESTADO DE REJEIÇÃO, CRIANDO UM LOOP.

OU MODIFICAR AS TRANSIÇÕES QUE LEVAM A n



HIERARQUIA DE CHOMSKY

⇝ PODEMOS USAR M.T.s PARA RESOLVER PROBLEMAS QUE NÃO SÃO DE DECISÃO.

- SEJA $f: \Sigma_0^* \rightarrow \Sigma_0^*$ UMA FUNÇÃO

- DIZEMOS QUE UMA M.T. 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 (APÓS $\triangleright$), ELA TERMINA PARANDO COM $f(w)$ NA FITA.

- DIZEMOS QUE f É UMA FUNÇÃO COMPUTÁVEL SE EXISTE M.T. QUE COMPUTA f.

Diagramas de Composição

- Escrever todas as transições pode ser trabalhoso
- Usaremos diagramas
  - Máquinas básicas
  - Regras de composição

1) Máquinas básicas

D = Move cabeçote p/ direita

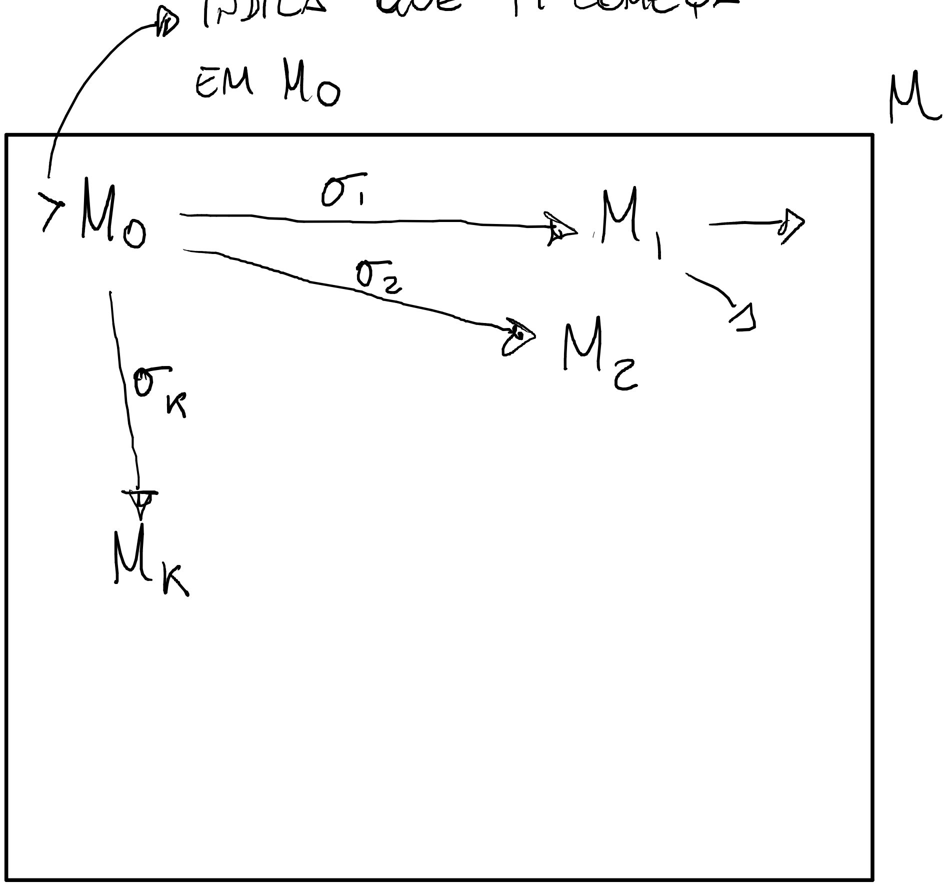
E = Move cabeçote p/ esquerda

W_a = escreve a

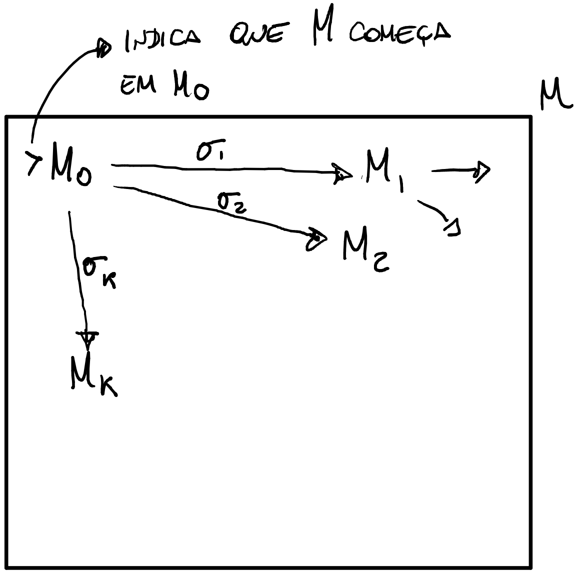
P = Para sem realizar ação

S = Para no estado de aceitação

N = Para no estado de rejeição


Indica que M 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 p/ o estado inicial da máquina corresp.

  ↳ se o símbolo lido não está presente, M para.

⇝ Permite escrever uma M.T. "por partes"

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

(2) $M_1 \longrightarrow M_2$ indica que executamos M_2 imediatamente após M_1

⇝ $M_1 M_2$

(3) $M_1 \xrightarrow{a \neq \sigma} M_2$ Vai para M_2 quando lê símbolo diferente de σ

⇝ $M_1 \xrightarrow{\overline{\sigma}} M_2$

- E_{σ} = MOVE P/ ESQ ATÉ ENCONTRAR σ

- D_{σ} = MOVE P/ DIR ATÉ ENCONTRAR σ

- $E_{\bar{\sigma}}$ = MOVE P/ ESQ ATÉ ENCONTRAR símbolo DIFERENTE DE σ

- $D_{\bar{\sigma}}$ = MOVE P/ DIR ATÉ ENCONTRAR símbolo DIFERENTE DE σ

EX: $f(x) = X + 1$

$▷1\,0000 \rightsquigarrow ▷10000$

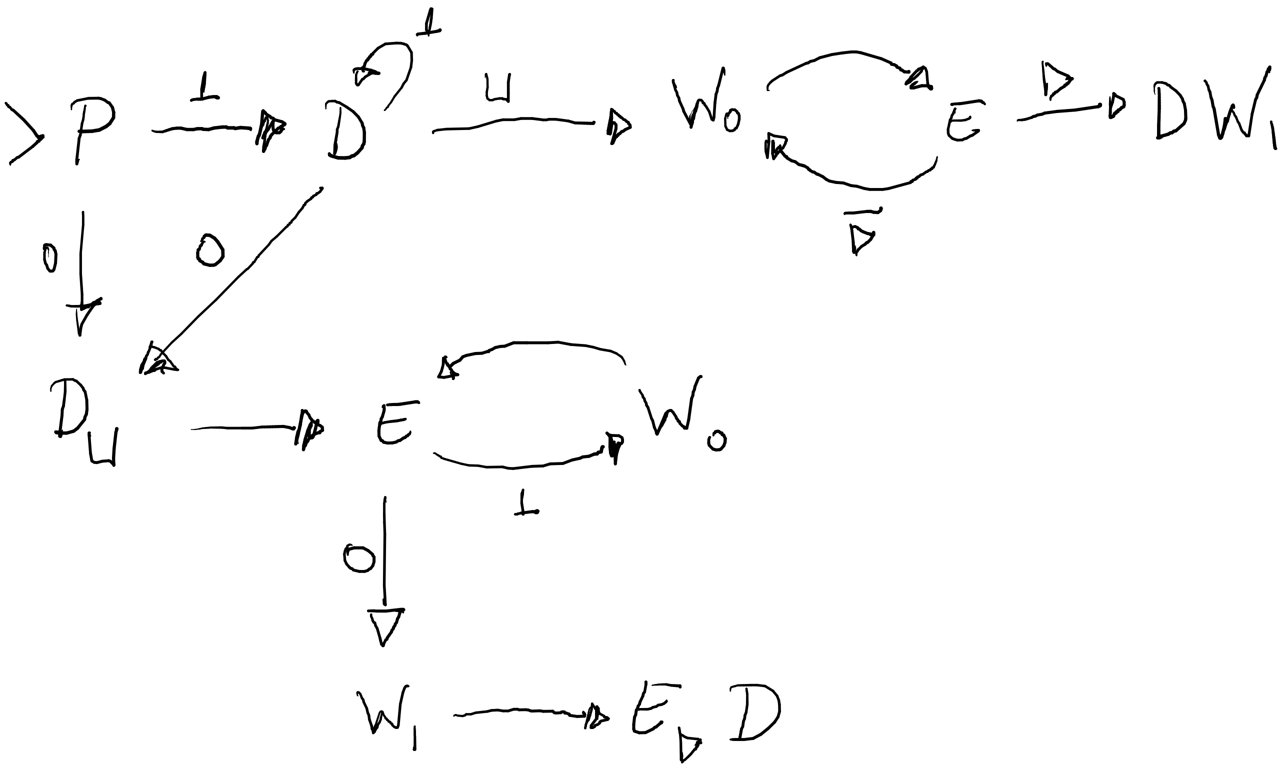
$▷\underline{1}1\;000\,0\;⊔$

$\hookrightarrow 1\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 0 = 22$

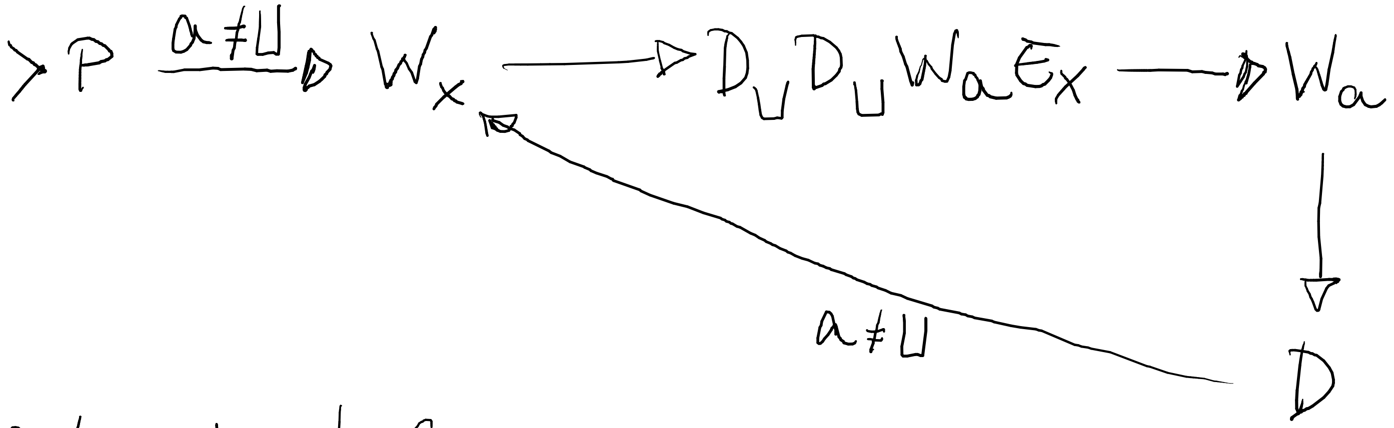
$31 = \sum_{i=0}^{4} i \cdot 2^i$

$\rightsquigarrow 32 = 1\cdot 2^5$

$▷10111 \rightsquigarrow ▷11000$

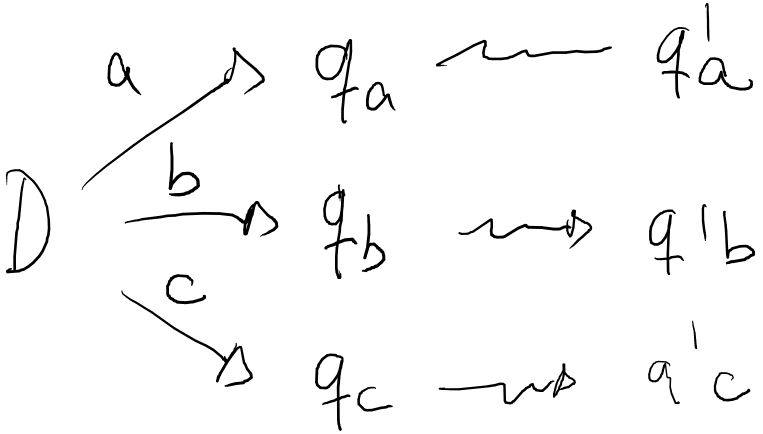


EX: MÁQUINA QUE COPIA : $f(w) = w \underline{\sqcup} w$



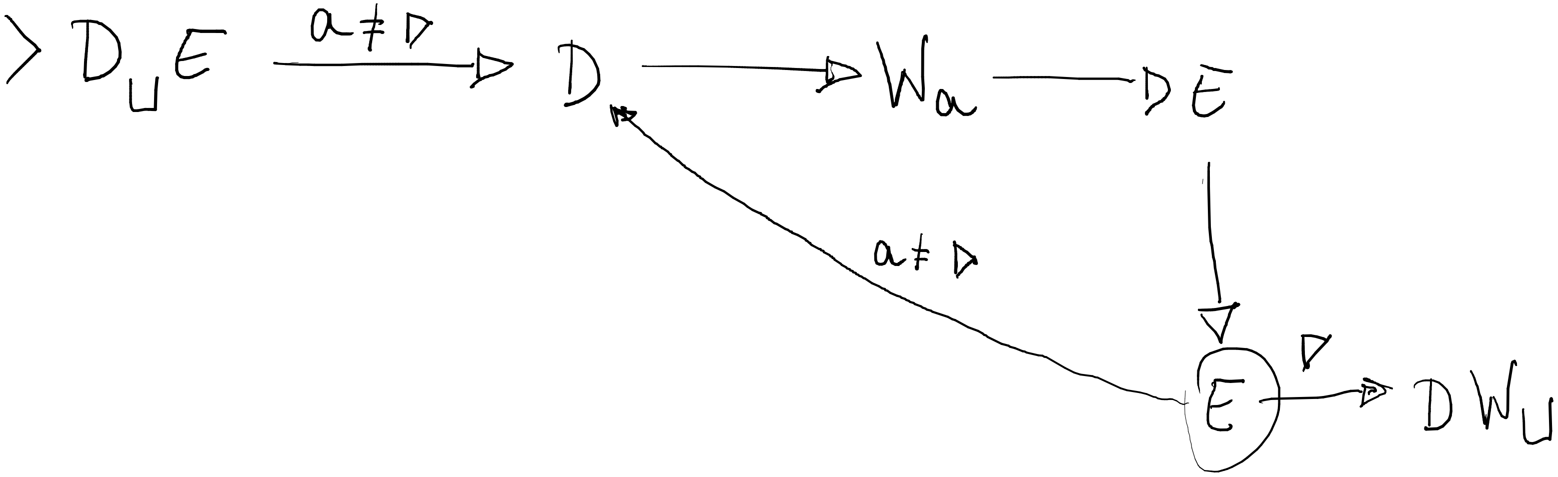
$a b c \underline{\sqcup} a b c$

$q_a \quad q_b$



EX: MÁQUINA QUE FAZ SHIFT : $f(w) = \underline{\sqcup} w$

$\rhd abc \rightsquigarrow \rhd \sqcup abc$



$\rhd \underline{a} bc \vdash \rhd abc\underline{\sqcup} \vdash \rhd ab\underline{c} \vdash \rhd abc\underline{c}$

$\vdash \rhd ab\underline{c}c \vdash \rhd a\underline{b}cc \vdash \rhd ab\underline{c}c$

$\vdash \rhd ab\underline{b}c \vdash \qquad \vdash \rhd \underline{a}bbc$

$\vdash \rhd a\underline{b}bc \vdash \underline{\rhd} aabc \vdash \rhd \underline{\sqcup} abc$

GENERALIZAÇÕES DE M.T.

- VEREMOS QUE EM VÁRIAS DIREÇÕES, NÃO GANHAMOS PODER COMPUTACIONAL AO CONSIDERARMOS M.T. MAIS POTENTES

EX:
- FITA INFINITA DOS DOIS LADOS
- MULTIPLAS FITAS
- MULTIPLOS CABEÇOTES DE LEITURA OU ESCRITA
- MOVIMENTAÇÃO MENOS RESTRITA DO CABEÇOTE

⟿ QUALQUER LINGUAGEM DECIDÍVEL, SEMI-DECIDÍVEL, OU FUNÇÃO COMPUTÁVEL POR UM MODELO DE M.T. TAMBÉM O É EM OUTRO MODELO.