def: Um aceitador é uma máquina de Turing $M = (\Sigma_0, \Sigma_1, \ldots)$ com a propriedade que existe $L \subseteq \Sigma_0^*$ tal que, para todo $w \in \Sigma_0^*$, se M é executada com w na fita (logo à direita de $\triangleright$) e o cabeçote começa na primeira casa à direita de $\triangleright$, então:

a) se $w \in L$, então a execução termina em algum momento

b) se $w \notin L$, então a execução não termina

L é chamada linguagem aceita por M, ou M aceita L.

def: Uma linguagem $L \subseteq \Sigma_0^*$ é chamada **recursivamente enumerável** (ou **semidecidível**) se existe uma máq Turing $M = (\Sigma_0, \ldots)$ que aceita L. Denotamos por R.E. a classe destas linguagens

Proposição: $Rec \subseteq R.E$

Prova: Exercício. ▨

Curiosidade: $L \subseteq \Sigma_0^*$ é R.E. sse existe uma máq. de Turing M com um estado especial com a propriedade que, ao ser executada na fita vazia, M nunca para e para cada $w \in \Sigma_0^*$ temos $w \in L \iff w$ está na fita de M em algum

momento da execução em que M está no seu estado especial

Veremos mais tarde: $RE \setminus Rec \neq \emptyset$

O exemplo "canônico" de $L \in RE \setminus Rec$ é o problema da Parada

Queremos estudar Rec & RE mais a fundo, mas o modelo "cru" de MT que vimos até aqui é pouco tratável na prática.

- Diagramas de composição
  - Diagramas de vértices & arestas
  - Dois tipos de "componentes":
    - → máquinas básicas
    - → regras de composição
  - Máquinas básicas:
    - D, a máquina que move seu cabeçote 1 casa para a direita e para.
    - E, " " " " " " esquerda " " " .

- W_a, a máquina que escreve a na célula atual da fita e para (uma máquina para cada a do alfabeto)
- P_1, a máquina que para imediatamente
- S (caso estejamos interessados em descrever um decisor), a máquina que para no estado s_1 de aceitação
- N, como S mas para rejeição

• Regras de composição
- Escrevemos $M_1 \xrightarrow{a} M_2$ para denotar

a máquina que funciona da seguinte forma:

1) funciona como M_1, até que M_1 pare
2) se M_1 parar e o cabeçote estiver sobre o símbolo a na fita, a máquina continua como M_2.

- Em um diagrama denotamos o "vértice inicial" com >
- Antes do exemplo, vamos melhorar um pouco mais a usabilidade:

1) Diagramas da forma $M_1 \overset{a}{\underset{b}{\rightleftarrows}} M_2$ podem ser escritos como $M_1 \xrightarrow{a|b} M_2$

2) Diagramas da forma $M_1 \longrightarrow M_2$ denotam que M_2 será executada após M_1 parar, independentemente de qual símbolo estiver sob o cabeçote. Alternativa de representação: $M_1 M_2$.

3) O diagrama $M_1 \xrightarrow{x \neq b} M_2$ denota a máquina

que executa M_2 após o término de M_1, desde que o símbolo sob o cabeçote não seja b. A letra x é usada como "variável" que guarda o símbolo lido. Alternativa: $M_1 \xrightarrow{\overline{b}} M_2$

4) E_a: a máquina que move o cabeçote para a esquerda até encontrar o símbolo a.

Em outras palavras, é o diagrama $E \circlearrowleft \overline{a}$

5) D_a, análogo (para a direita)

6) $E_{\bar{a}}$, análogo (para a esq enquanto encontra a)

$E \circlearrowleft a$

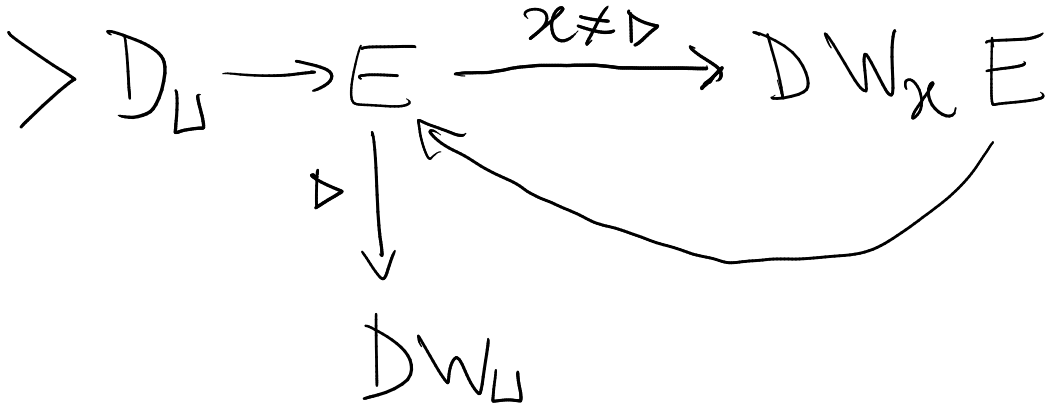
(vá para a esquerda até encontrar $x \neq a$)

7) $D_{\bar{a}}$, análogo.

Exemplo: Shift para a direita, i.e., a máquina que passa de $\triangleright a_0 a_1 a_2 \cdots a_n \sqcup\sqcup\sqcup \cdots$

até terminar em $\triangleright \sqcup a_0 a_1 a_2 \cdots a_n \sqcup\sqcup \cdots$

supondo que $\sqcup \notin \Sigma_0^*$!!



- (Não-) Generalização de Máquinas de Turing
  - ⇝ múltiplas fitas
  - ⇝ não-determinismo

def: Uma máquina de Turing com k fitas é como uma máq. de Turing usual $M = (\Sigma_0, \Sigma, Q, q, F, \delta)$, mas a função de transição δ tem tipo

função determinística

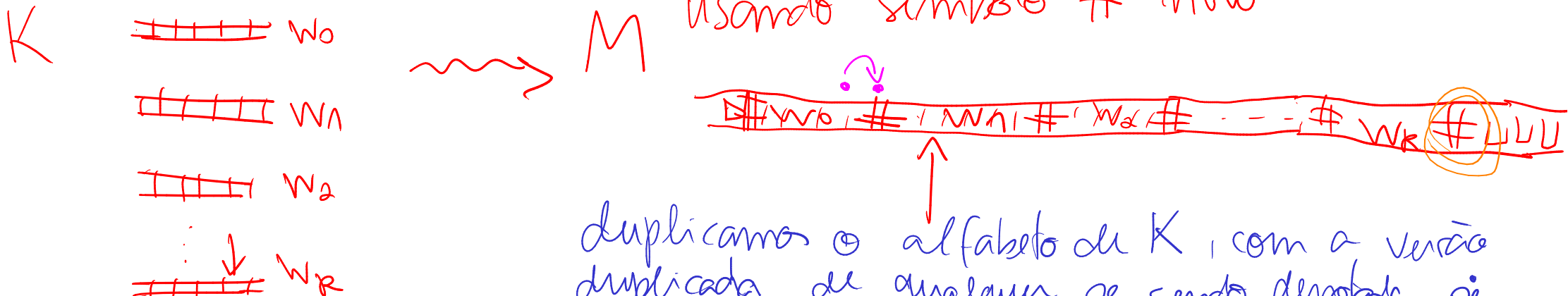
$$\delta : \left[(Q \setminus F) \times \Sigma^k \right] \to \left[Q \times \left(\Sigma \cup \{ \leftarrow, \rightarrow \} \right)^k \right]$$

Configurações & execução são definidos de forma análoga

Teorema: Máquinas de Turing "usuais" simulam exatamente as Máqs. de Turing com k fitas para qualquer $k > 0$.

Ideia da prova: Seja K uma máq. Turing com k fitas. Definimos M máq Turing usual que simula K da seguinte forma.

RASCUNHO



Passagem de configuração em M:

- "varrer" a fita até encontrar o $(k+1)$-ésimo separador #, "guardando" os símbolos da forma já vistos. Mais formalmente, temos estados de M correspondentes a cada sequência de símbolos do alfabeto de K, de tamanho $\leq k$
- Ao final da varredura, a "ação" a ser tomada está "clara", basta "copiar" o que K faria

- O caso problemático é quando uma destas instruções é "mova o cabeçote p/ a direita", mas o cabeçote de K estava no final da fita correspondente.
Assim, M "moveria o cabeçote virtual" • para o símbolo de separação de fitas #.
Neste caso, antes de mover o cabeçote virtual, fazemos um shift p/ a direita em todo o conteúdo da fita de M a partir da posição atual do cabeçote virtual ("abrindo espaço").

Não-Determinismo

def: Uma máq Turing não determinística (MTND)

$M = (\Sigma_0, \Sigma, Q, q_1, F, \Delta)$ é como uma usual, mas sua função de transição Δ tem tipo

$$\Delta : [(Q \setminus F) \times \Sigma] \longrightarrow \mathcal{P}([Q \times (\Sigma \cup \{\leftarrow, \rightarrow\})])$$

↑ conjunto das partes

tal que $\forall q \in Q \setminus F \; \forall a \in \Sigma \; (\Delta(q, a) \neq \emptyset)$

Configuração: como antes!

Mas a relação de "produção" $\vdash$ muda: cada configuração (q, w) produz um conjunto finito de configurações em 1 passo

(A maior quantidade de "possíveis produções em 1 passo" a partir de uma configuração qualquer é um número finito que só depende de M.)

Uma execução de M é uma sequência de configs de M onde cada nova config é uma configuração

que pode ser produzida pela anterior

RASCUNHO: As execuções de uma MTND a partir de uma config (q, w) podem ser vistas como uma árvore