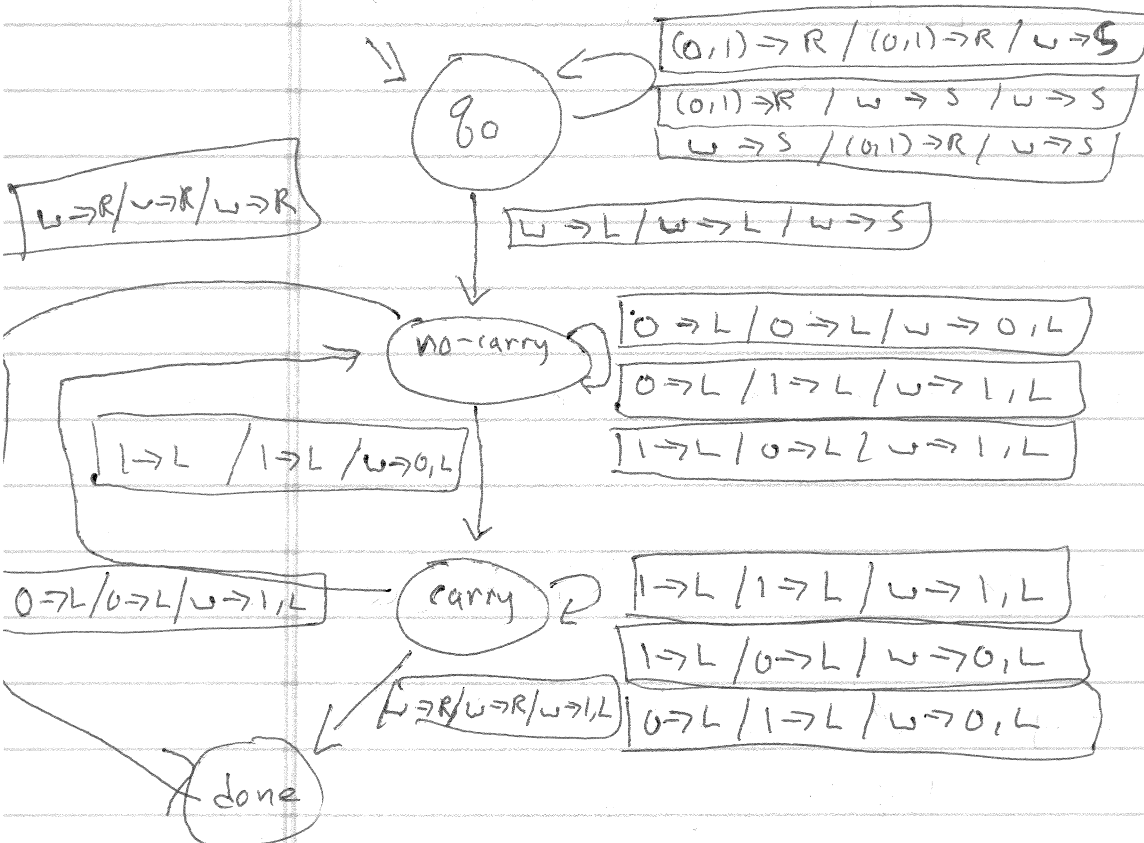
20-1/ 3-tape machine (tape 1 = LHS, tape 2 = RHS, tape 3 =

$$\left[\; q_0 \;\right] \begin{array}{l} 0110 \\ 1010 \\ \varepsilon \end{array} \quad \Rightarrow^* \quad \left[\; q_{done} \;\right] \begin{array}{l} 0110 \\ 1010 \\ 10000 \end{array}$$



0 or ⊔

20-2/ Non-deterministic TM

$$Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L, R\})$$

$$\begin{array}{ccc} & ua[q_i]bv & \\ \swarrow & \downarrow & \searrow \\ uac[q_x]bv & uac'[q_y]v & u[q_z]ac''v \end{array}$$

NFA
- oracle ⟶ can't compile
- forking ⟶ <u>can</u> compile through forkmachine
- back-tracking
- set of fringe states ⟶ can't work because of infinite paths

Fork-TM : $Q \times \Gamma \rightarrow (Q \times \Gamma \times \{L, R\}) + (Q \times Q)$

forked

fork-config = set of config $\cancel{\delta(q_i, b) = (q_x, q_y)}$ $\cancel{u[q_i]bv}$

$\{ua[q_i]bv\} \cup S \Rightarrow S \cup \{uac[q_j]v\}$
and $\delta(q_i, b) = (q_j, c, R)$

$\{ua[q_i]bv\} \cup S \Rightarrow S \cup \{ua[q_x]bv\} \cup \{ua[q_y]bv\}$
and $\delta(q_i, b) = (q_x, q_y)$

$$L(ftm) = \{ w \mid \{[q_0]w\} \Rightarrow^* S \cup \{u[q_a]v\}\}$$

If $\delta_{ND}(q_i, b) = \{(q_j, c, L), (q_k, c', R)\}$
then $\delta_F(\hat{q}_i, b) = (\hat{q}_{i1}, \hat{q}_{i2})$
$\delta_F(\hat{q}_{i1}, b) = (\hat{q}_j, c, L)$
$\delta_F(\hat{q}_{i2}, b) = (\hat{q}_k, c', R)$

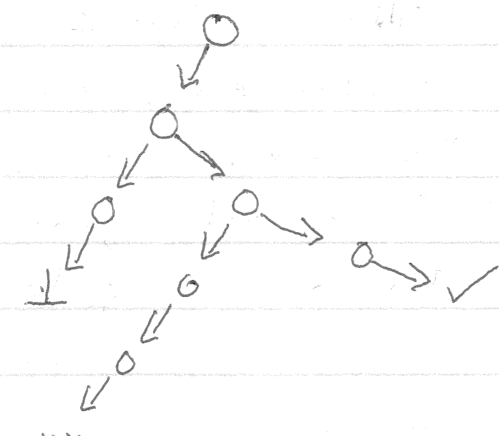
20-3/

Fork $\Rightarrow$ Normal

$\{u[q_i]v \;,\; x[q_j]y \;,\; z[q_k]h\}$

$\Rightarrow [start]\; u\hat{[}q_i\hat{]}v \;\hat{;}\; x[q_j]y \;\hat{;}\; z[q_k]h$

Back-tracking



depth-search
(try until failure
then go back)
$\Rightarrow$ fails because of divergence

breadth-first

$\delta(q_i, b) = \text{Options} \in [0, 2^{|Q| \times |\Gamma| \times 2}]$

$|opts| = 1$ $\quad$ $|opts| = 2$ $\quad$ $|opts| = 4$

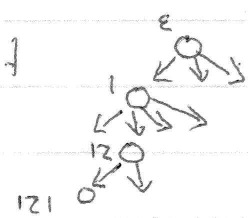
Some maximum option-size $= M$

if we have M symbols $= 1 \ldots m = \mathbb{M}$

then $\mathbb{M}^* =$ a string of options

ε = no options

address $\rightarrow$ 121 = first, second, first



3443

(sometimes addresses don't name configurations)

ordering of addresses:

$\varepsilon, 1, 2, 3, 4, 11, 12, 13, 14, 21, 22, 23, 24, \ldots$

20-4 / 3-tape machine

input = w

tape 1 : input

tape 2 : address

tape 3 : simulation

1) copy input to tape 3 with $[q_0]$ as state
   tape 3 = $[q_0] w$
2) simulate machine based on address in tape 2
3) if address ends, fail, but if machines accept, then accept
4) if we fail, generate next address and go to 1

Enumerator = TM

Have Enum $\Rightarrow$ Get TM :

TM (input w)
  Run Enum
  If it outputs w, accept

Have TM $\Rightarrow$ Get Enum :

Enum ()
  For $i \geq 0$,
    Run TM on s_i for only i steps (where s_i is the ith string in lexicographic order)
  if accepts, then print s_i