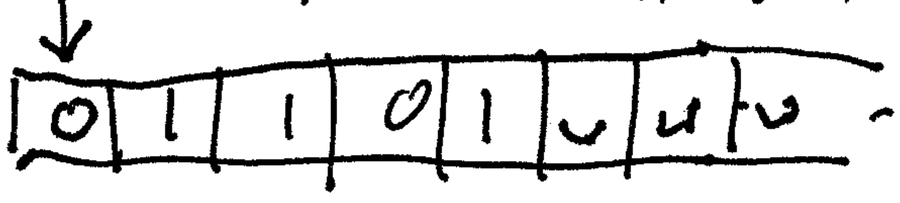
Lecture Thu 4/9: Two main issues with all the notation differences.

1. A TM halts only when it encounters a char c in a state q where it has no instruction for c.

   Note: For ~~DFA or~~ NFA this was a "crash": don't accept, even if $q \in F$

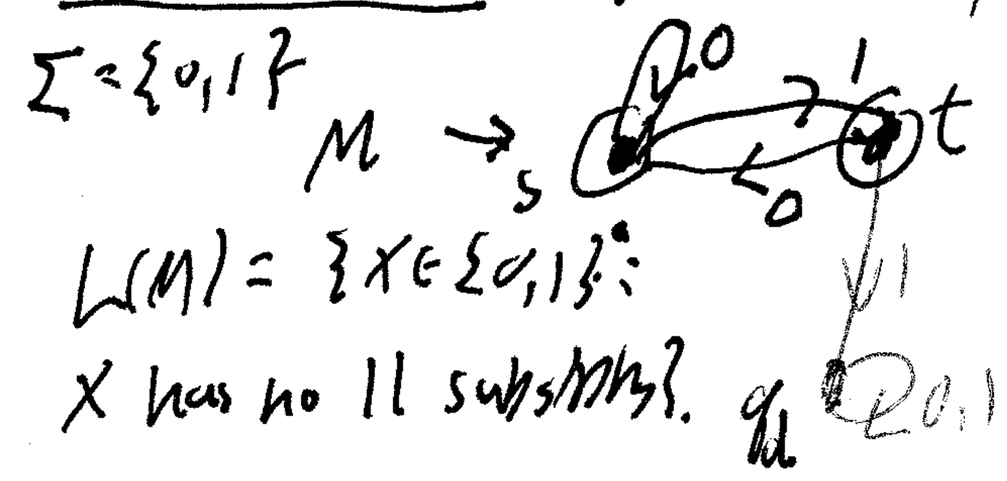
   For TM in the text, $\{q_{acc}, q_{rej}\}$ form, these states have no instructions, while every state $q \in Q \setminus \{q_{acc}, q_{rej}\}$ has instruction(s) for all chars. $\therefore$ q_{acc} and q_{rej} are the only places in which a TM can halt.

2. Is the tape infinite in both directions? Sipser, Kit text, lecture: yes



Also Point 3: We want to say that a DFA or NFA "Is-A" TM.

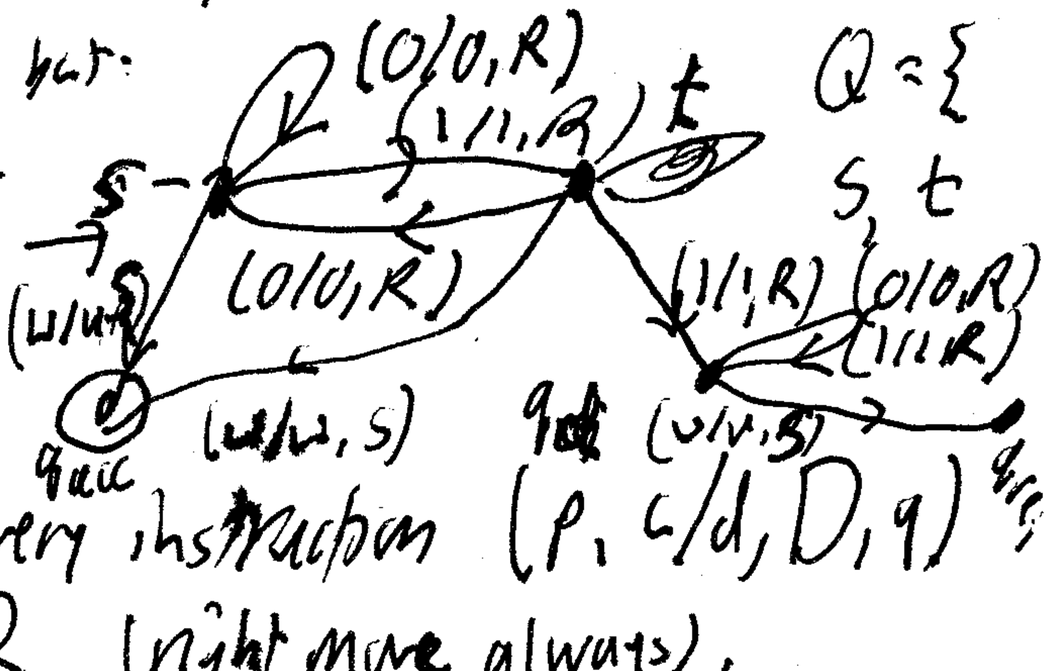
$\Sigma = \{0,1\}$



$L(M) = \{x \in \{0,1\}^* :$ x has no 11 substring$\}$.

$\Sigma = \{0,1\}$ but:

TM $M' =$



$\Gamma = \{0,1,u\}$

A finite automaton "Is-A" TM in which every instruction $(p, c/d, D, q)$ has $d = c$ (no change) and $D = R$ (right move always).

Notation for Transitions.

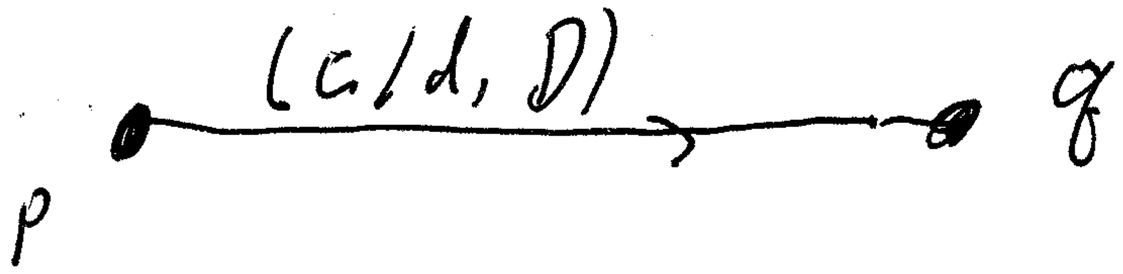
Nondeterministic TM

General (NTM): $\delta \subseteq \underbrace{Q \times \Gamma}_{\text{read}} \times \underbrace{\Gamma \times \{L,R,S\} \times Q}_{\text{write, move, next state}}$ (write pushed first.)

Instruction (aka "tuple") format: $(p, c/d, D, q)$

OK to use normal comma for "/"

In diagrams

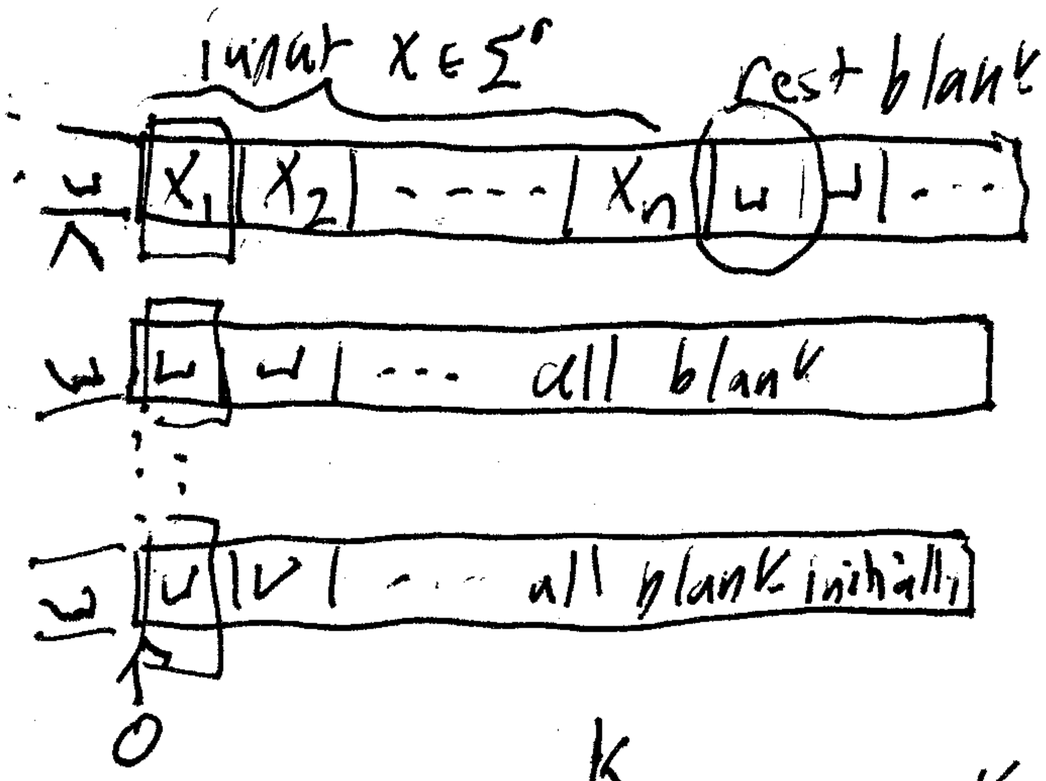


An NTM program has some state(s) p with 2 or more instructions beginning $(p, c/\cdots)$.

Otherwise, δ is a partial function on $Q \times \Gamma$. Textbook convention makes δ a total function, $\delta : (Q \setminus \{q_{acc}, q_{rej}\}) \times \Gamma \to \Gamma \times \{L,R,S\} \times Q$ for DTMs.

Abstractly $\delta(p,c) = (d, D, q)$. (Text: $= (q, d, D)$.)

Multiple Tape TMs:



Input tape: (can be read-only)

$K-1$ work tapes initially all blank.

All tapes initially all blank to left of our origin point.

Now transitions have $\delta \subseteq Q \times \Gamma^K \times \Gamma^K \times \{L,R,S\}^K \times Q$

(DTM: $\delta : (Q \setminus \{q_{acc}, q_{rej}\}) \times \Gamma^K \to \Gamma^K \times \{L,R,S\}^K \times Q$)

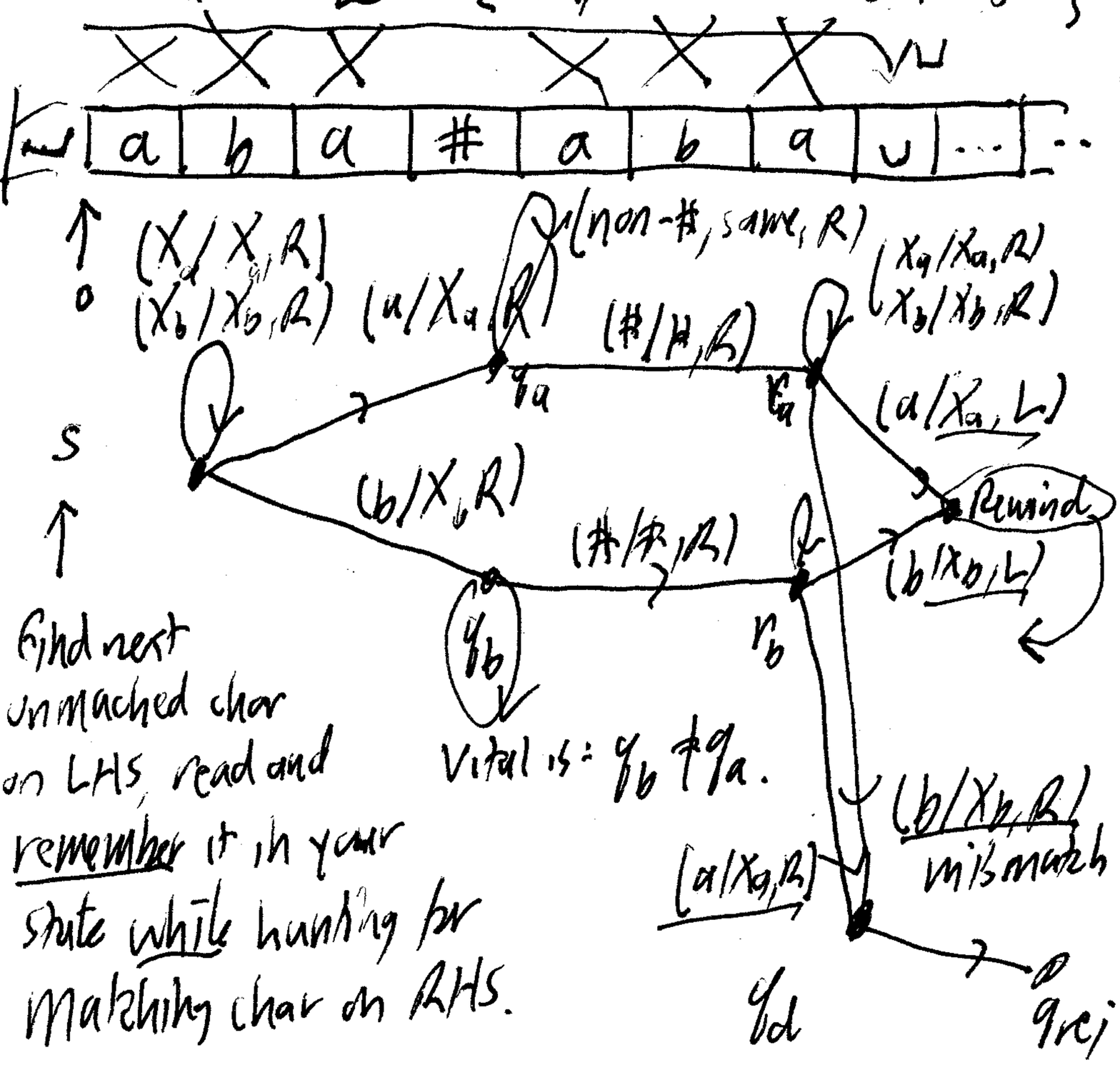
Instructions $\left(p, \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_K \end{pmatrix} / \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_K \end{pmatrix}, \begin{pmatrix} D_1 \\ D_2 \\ \vdots \\ D_K \end{pmatrix}, q\right)$ (non-left, if $D_1 \neq L$)

- Input tape is read-only if $d_1 = c_1$ in all instructions, right-only if $D_1 = R$ always.
- Tape 2 is a stack if $D_2 = L \Rightarrow d_2 = \sqcup$. "2-PDA" PDA if tape 1 is non-left, tape 2 = stack.

Example: $L = \{ w\#w : w \in \{a,b\}^* \}$

$\Sigma = \{a, b, \#\}$
↑ divider.

$\Gamma = \{\sqcup, X_a, X_b\} \cup \Sigma$.



Strategy: X-out the next char on the left, and see if the first non-X-ed char on the right matches it.

Find next unmached char on LHS, read and remember it in your state while hunting for matching char on RHS.

If you have a second tape for work, then you can copy X ~~received~~ on it and match it in one sweep (but not as a stack).

Extra:

The machine still needs a component coming out of the "Rewind" state that ~~finds~~ seeks the leftmost unmatched char (a or b), and if everything between the start and the '#' is X-ed, tests whether everything to the right of the # is X-ed as well. If so, then the input was a w#w for which so the TM accepts. If not — or if it couldn't match a char earlier — then it was w#v where $v \neq w$, so it rejects. We also need code to reject if there is no '#' or if there are 2 or more '#'-es.

* The 2-tape TM is not a PDA because it rewinds on the second tape without erasing.

Analytically, the machine has a big loop that executes up to $|w|$ times, and each iteration of the body takes at least $|w|$ steps, giving $\geq |w|^2$ time in the worst case. But if we have a second tape, we can copy the RHS underneath the LHS (in $O(|w|+|v|)$ time) and then do the whole attempted matching in one sweep ($O(|w|+|v|)$ time again, or less). So the time is linear not quadratic. But