Lecture 4/19: Two main issues with all the notions in 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 ∈ F.

For TM in the text, 3 cases: grey, black, form. These states have no instructions while every state q ∈ Q \ { qacc, blank } has instruction(s) for all chars. "Grey" are the only places in which a TM can halt.

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

A TM with a 2-way infinite tape cannot "crash."

Also point 3: We want to say that a DFA or NFA is a TM.

Σ = { 0, 1, z };
M = ( Q, Σ, δ, q0, F )
L(M) = { x ∈ Σ* | x has no 11 substrings. }

A finite automaton "Es-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.

Non-deterministic TM

General (NTM): \( S \subseteq (Q \times \Gamma)^* \times \Gamma \times \{L, R, S\}^* \times Q \)

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

OK to use normal comma for In diagrams \( (c/d, D) \rightarrow q \)

An NTM properly has same state(s) \( p \) with 2 or more instructions beginning \( (p, c/\ldots) \).

Otherwise, \( S \) is a partial function on \( Q \times \Gamma \). Text's convention makes \( S \) a total function \( S: (Q \times \{\text{acc, rej, }\}^* \times \Gamma) \rightarrow \Gamma \times \{L, R, S\}^* \times Q \)

for OTMs.

Abstractly \( S(p, c) = (d, D, q) \).

Multiple Tape TMs

Input tape: can be read-only

All tapes initially all blank to left

Start on left

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

(OTM: \( S: Q \times \{\text{acc, rej, }\}^K \times \Gamma^K \rightarrow \Gamma^K \times \{L, R, S\}^K \times Q \))

Instructions
\[
(p_i, c_i, d_1, D_1, q_1, \ldots, c_k, d_k, D_k, q_k)
\]

- Input tape is read-only if \( d_i = c_i \) in all instructions, right-only if \( D_1 = R \) always.
- Tape 2 is a stack if \( D_2 = L \Rightarrow d_2 = \ldots \) "PUSH" MA if type 1 is non-left, type 2 = stack.
Example: \( L = \{ w \in \{a, b\}^* \mid \text{w has an odd number of } a\text{s} \} \)

\[
\Sigma = \{ a, b, \# \}
\]

\[
\Gamma = \{ \#, X_a, X_b \} \cup \Sigma
\]

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

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

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

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 \( \Omega(|w|^3) \) 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.