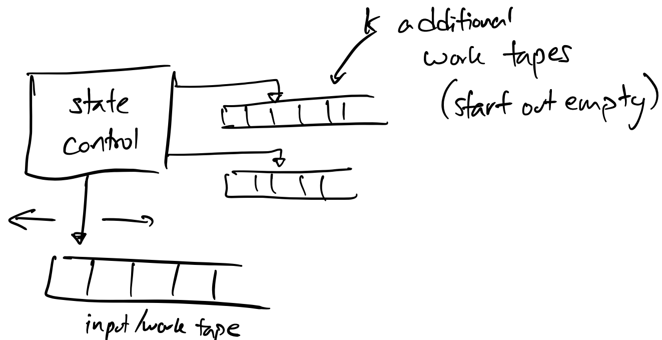
COMS W3261 – Lecture 9, Part 2: Variant TMs.

2.1) Multitape TM.



Formally, the Multitape TM is a 7-tuple like the TM but with the transition function

$$\delta : \underbrace{Q}_{\text{state}} \times \underbrace{\Gamma^k}_{\substack{k \text{ different} \\ \text{tape symbols}}} \longrightarrow \underbrace{Q}_{\text{state}} \times \underbrace{\Gamma^k}_{\substack{k \text{ write} \\ \text{symbols}}} \times \underbrace{\{L, R, S\}^k}_{\substack{k \text{ head} \\ \text{movements.}}}$$

(Looks like $\delta(q_i, a_1, a_2 \cdots a_k) = (q_j, b_1, b_2, \ldots, b_k, L, R, R, S, \ldots)$.)

Theorem: Every Multitape TM has an equivalent single-tape TM.

Proof sketch: Simulate all k tapes of a given k-tape TM on one tape. Define a new machine M:

(1) Start by writing down delimiters for the contents of k tapes onto the one work tape. On input $w_1 w_2 \cdots w_n$:

represent tape heads.

$$① \dot{w}_1 w_2 w_3 \cdots w_n ② \dot{\sqcup} ③ \dot{\sqcup} ④ \dot{\sqcup} \cdots ⓚ \dot{\sqcup}$$

marks the beginning of simulated tapes.

(2) mark virtual tape heads.

(3) simulate the transition function for the k-tape TM. If we run out of space on a virtual tape, we run a special subroutine to shift the tape contents over and add a space.

(4) We accept / reject when the simulation accepts/rejects. $\square$

Takeaway: to show something is Turing-recognizable or decidable, we can assume multiple tapes w/o loss of generality.

2.2) Nondeterministic TMs

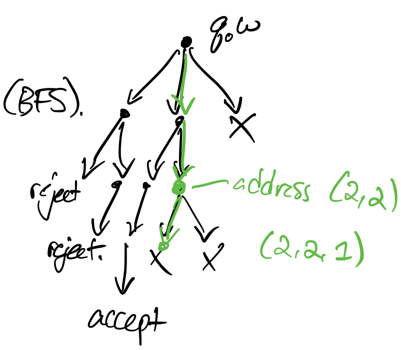
New transition function (other formal details the same):

$$\delta : \underbrace{Q \times \Gamma}_{\text{state, tape symbol}} \longrightarrow \underbrace{\mathcal{P}(Q \times \Gamma \times \{L, R\})}_{\text{set of new configurations we go to.}}$$

(Accept if any branch accepts.)

Theorem: Every Nondeterministic TM has an equivalent deterministic TM.

Proof sketch: Nondeterministic computation looks like a decision tree.



We can use a DTM to traverse all branches of this tree according to breadth-first search (BFS). Thus we eventually find any branch that reaches an accept state. Accept in this case, reject if we finish exploring the tree.

(Why not depth-first search? Infinite loops.)

Define a TM D to do this with 3 tapes:

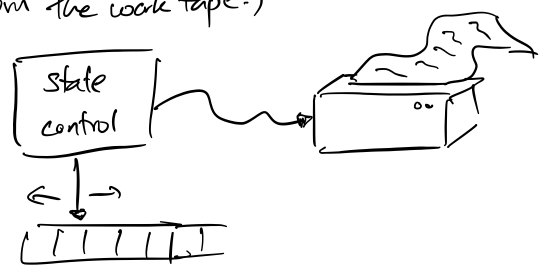
Input tape: unchanged.
Address tape: stores our position on the tree
Work tape: keeps track of our current computation.

Every time we change the address we copy a fresh input copy onto the work tape and simulate to this point. $\blacksquare$

Takeaway: To prove languages are Turing-recognizable or decidable, we can assume nondeterminism w/o loss of generality.

2.3) Enumerators.

Idea: Give a Turing Machine a printer that can write down strings. (from the work tape.)



<u>Theorem</u>. A language is Turing-recognizable if and only if some enumerator enumerates it (i.e., writes down all strings in the language.)

<u>Proof</u>. $\Rightarrow$. Suppose some enumerator E enumerates a language L. We define the following TM that recognizes L:

M = "On input w:
Simulate E. Every time E outputs a string, compare the string with w. On a match, accept."

$\Leftarrow$. Suppose a TM M recognizes a language L. We show an enumerator E that enumerates L. Let $s_1, s_2, s_3 \dots$ be an infinite sequence containing all strings over Σ^*, the input alphabet of M.

E = "For $i = 1, 2, 3, \dots$:
Simulate M for i steps on strings $s_1, s_2 \dots s_i$.
If any computation accepts, print that string."

Suppose some string $s_j \in L(M)$. M takes k steps to accept s_j. Now when the loop reaches the iteration $\max(j, k)$, we'll accept when we simulate M on s_j, and we will enumerate it.

(Note here — we simulate i strings for i steps to avoid infinite loops.) $\square$

// Historical note: the Turing-recognizable languages are sometimes called the Recursively Enumerable languages because of this Theorem. The class is often abbreviated RE.

$\underline{MIP}^* = RE.$

Next up: From TMs $\longrightarrow$ "general purpose algorithms"