Lecture 5

CSE 431
Intro to Theory of Computation

Recall:

- $L(M) = \{ w \in \Sigma^* \mid q_0 w \vdash_M^* u q_{acc} v \quad u, v \in \Gamma^* \}$
  is the language recognized by M

- M is decider iff M either accepts or rejects each string in Σ^*

- A language L is $\boxed{\text{Turing-recognizable (or recursively enumerable)}}$ abbreviated as "r.e."
  Today we see why $\rightarrow$
  iff $L = L(M)$ for some TM M

- A language L is decidable
  iff $L = L(M)$ for some decider M

- Two machines are equivalent iff they have the same I/O behavior
  - e.g. they have the same input alphabet and they accept/reject the same strings

Last time:

Thm For every k-tape TM there is an equivalent 1-tape TM

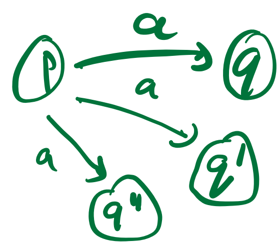
Note: k was a fixed constant independent of the input. Need to know k to define the 1-tape TM.

Nondeterministic TMs (NTMs)

conceptual, not practical model

Recall DFAs (deterministic) vs NFAs (nondeterministic)

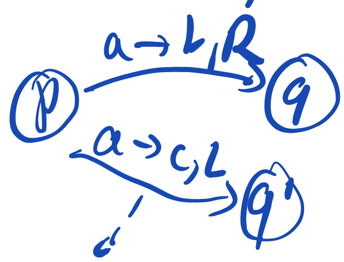
one possible move given p, a



many possible moves given p, a (or none)

We see that NFA's were convenient but for every NFA there was an equivalent DFA (though it requires exponentially more states in the worst case)

With NTMs we get the same option



of multiple possible moves given p, a.

Formally only change is that now

$$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$$

power set

i.e. a set of possible moves, not just one

Execution of NTM's:

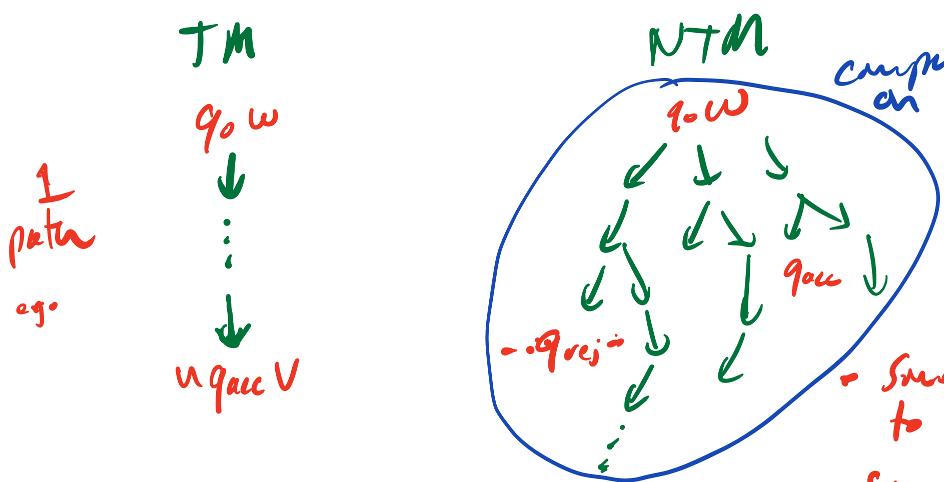
start configuration $q_0 w$

$C \vdash_M D$ iff there some move in δ that takes C to D in one step.

$L(M) = \{ w : q_0 w \vdash_M^* u\, q_{acc}\, v \quad u, v \in \Gamma^* \}$

is the language recognized by M.

Use $\rightarrow$ to denote $\vdash_M$



computation tree on input w

many paths in a tree

- Some may lead to q_{reject}
- Some may run forever
- Some may just die off

accept iff there is some q_{acc} somewhere

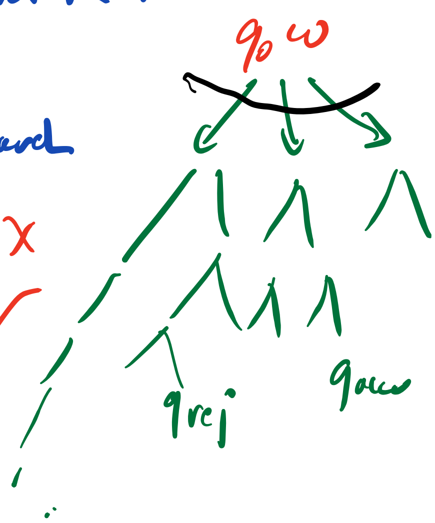
View of Nondeterminism:

- "God's eye" view — accept iff q_{acc} somewhere in infinite tree
- Perfect guesser — at each step if there is a move to lead to q_{accept}, M will take one
- Parallel exploration — M explores all branches in parallel

Theorem: For every NTM there is an equivalent TM

Proof idea: Graph search: explore every node in computation tree, stopping early if q_{accept} is found

Computation tree



Graph Search Options

DFS ✗

bad because might get stuck on infinite path

BFS ✓

fan-out of every node is at most

$b = |Q \times \Gamma \times \{L,R\}|$
$= 2 \cdot |Q| \cdot |\Gamma|$

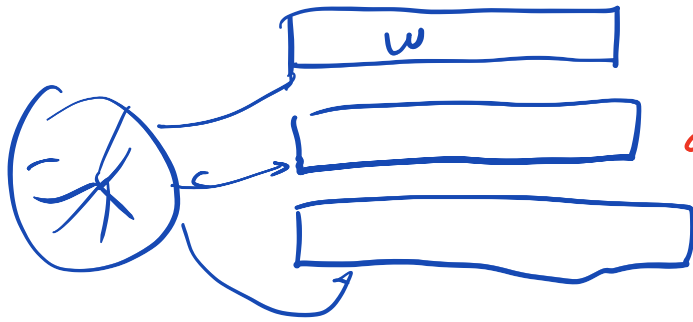
which is the max # of possible next moves for a configuration.

Associate each move with a number from $0, \ldots, b-1$

- Each node v in tree at level t associated with a base b string of length t address of v.
- Some addresses might not be reachable nodes

Overall idea: loop through all addresses (strings, base b) figuring out what the configuration would be at that node and stop & accept iff that accepts

Implementation: 3 tapes



input tape

address tape

work tape

Repeat forever

1. Copy input from input tape to work tape
2. Use address on tape 2 to see which sequence of moves to try
   - Execute each move on work tape if legal for M
   - If not legal abort this address
   - if q_{accept} reached then halt & accept
3. Erase work tape
4. Run machine to convert address on tape "2" to next address (just as with HW problem 2 but for bigger b.)

$\varepsilon, 0, 1, \ldots, b-1, 01, 02, \ldots, 0(b-1), \ldots$

This will find a q_{acc} iff there is one

Note: If at some address length t all addresses abort or reject, can reject.

Suppose original NTM accepted w in T steps

New TM will explore $\sim b^T$ addresses (actually $\sum_{t=0}^{T} b^T$)

This is $2^{O(T)}$ since b is constant.

P vs NP question: can we reduce $2^{O(T)}$ to $\text{poly}(T)$?

<u>Enumerator TMs</u>

- no input
- read/write work tape initially blank
- print [• write-only 1-way output tape alphabet $\Sigma \cup \{\$\}$ $\$ \notin \Sigma$
  $\$ = $ "\n"

Language enumerated by M is the set of strings $w \in \Sigma^*$ that M eventually prints (between two $\$$ on output tape) $E(M)$

<u>Thm</u> L is Turing-recognizable $\iff$ there is an enumerator M s.t. $L = E(M)$ recursively enumerable

Proof ($\Leftarrow$) Suppose there is an enumerator M s.t. $L = E(M)$

we build M into M' but modify it a bit

3-tape TM M' recognizing L:

- On input w, run M (using 2 other tapes)
- Every time M prints a \$ Compare the string it just produced to w. If they are equal accept. If not just continue

Clearly M' will accept w iff M prints w.

($\Rightarrow$) this is the harder direction

Suppose we have a TM M'' with $L = L(M'')$

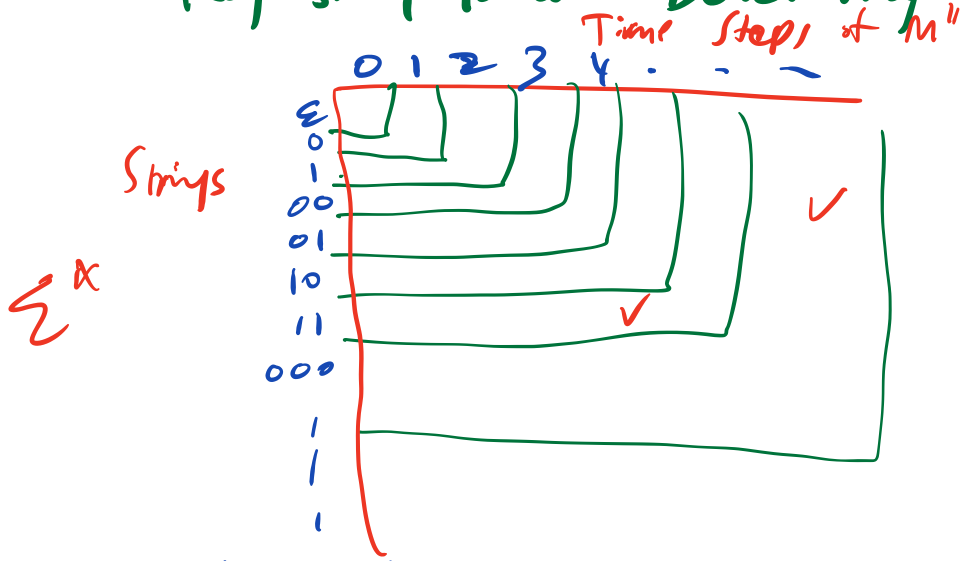
- We need to build an enumerator for L that runs M'' on all possible strings in Σ^*

- We can generate all the strings in Σ^* one after another

using the counter method we used for addresses

That's an infinite # of strings but just one computation may run forever!

Key Saving idea: "Dovetailing"



We need to try every string for all possible # of time steps (find all the accepts)

Enumerator M''':

For $t = 0 \ldots \infty$ do
- For each of the first t strings $w \in \Sigma^*$
  - Run M'' on input w for t steps
  - If M'' accepts print w

Has a modified version of M'' inside it

Eventually, will explore every point in above infinite table so will find and print all accepted strings

$\therefore E(M''') = L$. ∎