CS 39
Theory of Computing

Nondeterministic Turing Machines
Amit Chakrabarti

A closure property
• If $L_1$ and $L_2$ are decidable, so is $L_1 \cup L_2$.
• Let $M_i$ be a decider for $L_i$ (i = 1, 2). Let $M$ be the following 2-tape TM:
  $$M = \text{"On input } x, \text{ 
1. Copy } x \text{ to tape 2.} 
2. \text{Simulate } M_1 \text{ on input } x, \text{ using tape 1.} 
3. \text{If } M_1 \text{ accepts, then ACCEPT} 
4. \text{Else} 
4.1 \text{Simulate } M_2 \text{ on input } x, \text{ using tape 2.} 
4.2 \text{If } M_2 \text{ accepts, then ACCEPT, else REJECT."}$$

Closure properties
• If $L_1$ and $L_2$ (over $\Sigma$) are decidable, so is
  - $L_1 \cup L_2$
  - $L_1 \cap L_2$
  - $\Sigma^* - L_1$
  - $L_1 L_2$
  - $L_1^*$
• Think how you might prove these.

Recap: configuration of a TM
• Recall: $TM = 7$-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$
  (States, InputAlph, TapeAlph, Transitions, StartState, AccState, RejState)
• $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$
• A configuration of a TM specifies three things
  - Current state
  - Tape contents
  - Head position
Recap: yielding

• If configuration $C_2$ is a successor of $C_1$, we say “$C_1$ yields $C_2$”.

• Note: TM is deterministic, so a configuration either yields a unique configuration or yields nothing.

Nondeterministic Turing Machines

• Abbreviated NDTM
• TM’s by default deterministic
  – *Always* say NDTM if you want nondeterminism
• 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$, just like a TM
  – Except, $\delta : Q \times \Gamma \rightarrow 2^Q \times \Gamma \times \{L,R\}$
• This means, a configuration can now yield several different new configurations.

NDTM computation formalized

• Consider NDTM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$
• We say $M$ accepts $x \in \Sigma^*$ if
  – $\exists$ sequence $C_0, C_1, \ldots, C_t$ of configurations of $M$ s.t.
  – $C_0 = q_0^x$
  – $C_{i-1}$ yields $C_i$ (for all $i$, $1 \leq i \leq t$)
  – $C_t$ is an accepting configuration
• Behavior for $x \notin L$
  – Happy if $M$ never enters accept state (recognizer)
  – Require $M$ to always enter reject state (decider)

Uses of nondeterminism

• Simpler solutions for $\{ww : w \in \Sigma^*\}$
  – Use nondeterminism to guess midpoint and insert a ‘#’ symbol there. Then proceed as for $\{w#w : w \in \Sigma^*\}$.
  – Better yet, use two tapes *and* nondeterminism.
• Simpler proofs of closure properties
  – For $L_1L_2$ and $L_1^*$, just use nondeterminism to decide how to “break up” the input.
• But, to prove decidability, we want TMs, not NDTMs and not $k$-tape NDTMs!
Turning a NDTM into a TM

- Let $M$ be a NDTM; design TM $M'$ to simulate $M$.
- $M'$ will try out every possible branch of $M$’s computation. A branch is a particular sequence of nondeterministic choices.
- $M'$ will have three tapes
  - Tape 1: the input to $M$, never overwritten
  - Tape 2: the tape of $M$ in the current branch
  - Tape 3: sequence of ints describing current branch

Representing the branches

The configuration $C_6$ is represented as “3,1”