CSE 105
Theory of Computation

- Professor Jeanne Ferrante
Today’s Agenda

– Descriptions of TM’s
– A Variant of TM’s: Enumerators

Announcements and Reminders:
• HW 5 Due Fri May 6, 11:59 pm
• RQ 6 Due Mon May 9, 11:59 pm
• Exam 2 Review Session Mon May 9, 8 pm - 9:50 pm in Peterson 108
• Exam 2 on Wed May 11, 8:00 pm - 9:50 pm, emphasizing all material since Exam 1
  • Study guide will be out tomorrow
  • Not in your usual classroom: WLH 2001
  • We’ll have BETTER seat assignments!
Review: The Turing Machine

- Given a current state $q$, current tape symbol $a$, $\delta \rightarrow$ new state $q'$, new tape symbol $z$ (replaces $a$), and after write, move “head” L or R (unless try to move off left end of tape)

- On input $w$, $M$ either:
  1. Enters the accept state $q_{\text{acc}}$ and accepts $w$, or
  2. Enters the reject state $q_{\text{rej}}$ and rejects $w$, or
  3. Does neither 1 nor 2, in which case we say $M$ does not halt on input $w$, and the input is not accepted

- Language of $M = \{ w \mid w \text{ is accepted by } M \}$

- If a TM always halts, it’s a decider.
Review: Formal Definition of TM

$TM \ M$ is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$:

• $Q$ is a finite set of states
• $\Sigma$ is a finite input alphabet (no blank $\square$)
• $\Gamma$ is a finite tape alphabet (includes blank $\square$) with $\Sigma \subseteq \Gamma$
• $\delta: Q \times \Gamma \to Q \times \Gamma \times \{L,R\}$ transition function
• $q_0$ is the start state
• $q_{\text{acc}}$ is the accept state
• $q_{\text{rej}} (\neq q_{\text{acc}})$ is the reject state
Using the previous definition: Are TM’s as defined Deterministic or Nondeterministic?

A. Deterministic
B. Nondeterministic
C. Sometimes Deterministic
D. Neither Deterministic nor Nondeterministic
E. Don’t Know
Review: Configurations of TM

• A configuration of $M$ is a string $uqv$ where
  – $q$ is a state in $Q$
  – String $uv$ is the current (nonblank) tape contents
  – $M$’s head is reading the first symbol of $v$

• Start configuration: $q_0w$ (w the input)

• Accepting configuration: $uq_{acc}v$

• Rejecting configuration: $uq_{rej}v$

• A halting configuration is an accepting or rejecting configuration
Executing a Transition
Suppose we have at TM with

• \( \Gamma = \{a, \ b, \ c, \ d, \ \square \} \)
• \( Q = \{q_1, \ q_2, \ q_3, \ q_4, q_{\text{acc}}, q_{\text{rej}}\} \)
• \( \delta(q_2, c) = (q_3, d, R) \) and \( \delta(q_2, d) = (q_3, c, L) \).

Let strings \( u, v, x, y \) be in \( \Gamma^* \). Which configuration does the current configuration, \( xqcq_2dy \), yield?

A. \( xq_3ddy \)
B. \( xcdq_3y \)
C. \( xq_3ccy \)
D. \( xq_3ccdy \)
E. None of the above or more than one of the above
Review: Acceptance of a TM

TM $M$ accepts input $w$ if there is a sequence of configurations $c_1 \ldots c_k$ with

1. $c_1$ is the start configuration
2. $c_i$ yields $c_{i+1}$ by following $\delta$ one step
3. $c_k$ is an accepting configuration

*Note that all the input $w$ need not be read in order to accept $w*$

$L(M) = \{ w \mid M$ accepts $w\}$
Review: Deciders and Recognizers

• A is *Turing-recognizable* if $A = L(M)$ for some TM $M$

• If $M$ always halts, $M$ is a *decider*

• $A$ is *decidable* if $A = L(M)$ for a decider TM $M$. We say $M$ *decides* $A$. 
The Turing-Recognizable Languages are Countable.

A. True

B. False

C. Don’t Know
Some TM’s don’t stop!

TURING MACHINE DESCRIPTIONS
Turing Machine Descriptions

• For Turing Machines, we will often omit the state-transition diagram, or fully specifying the transition function.

• Alternative: An implementation-level description (in words) of how the machine functions.
TM Descriptions

• A *formal* description gives the states, transition function, etc.

• An *implementation-level* description is an English description of how the TM moves its head, stores data on tape, accepts, rejects.

• We’ll be using *implementation-level* descriptions of TM’s often.
Formal Description of TM $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, q_{\text{acc}}, q_{\text{rej}})$

$Q = \{q_{\text{acc}}, q_{\text{rej}}, q_1, \ldots q_{14}\}$

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

$\Gamma = \{0, 1, \#, x, _\}$

$\delta$ is given* by diagram

$q_1$ is the start state

$q_{\text{acc}}$, $q_{\text{rej}}$ the accept and reject states

* The diagram is missing the state $q_{\text{rej}}$ and transitions to it

**FIGURE 3.10**
State diagram for Turing machine $M_1$
Implementation-level description of TM M1

M1 = “On input w:

1. Scan the input from left to right to check whether it is of form \{0,1\}^*\#\{0,1\}^*. If not, *reject*. If the input consists of only #, *accept*.

2. Return the head to the left hand end of tape.

3. Zig-zag across the tape, checking that the first unmarked symbol to the left of the # is the same as the corresponding unmarked symbol following #. If the corresponding symbols do not match, or there is no unmarked symbol left after the #, *reject*. Otherwise if the symbols match, mark them and continue.

4. If all symbols to the left of # have been marked, check for unmarked symbols after the #. If any unmarked symbols remain to the right of the #, *reject*; if none are found, *accept*. ”
Qu: What is $L(M_1)$?

$M_1 = \text{“On input } w:\text{ “}

1. Scan the input from left to right to check whether it is of form $\{0,1\}^*#\{0,1\}^*$. If not, reject. If the input consists of only #, accept.

2. Return the head to the left hand end of tape.

3. Zig-zag across the tape, checking that the first unmarked symbol to the left of the # is the same as the corresponding unmarked symbol following #. If the corresponding symbols do not match, or there is no unmarked symbol left after the #, reject. Otherwise if the symbols match, mark them and continue.

4. If all symbols to the left of # have been marked, check for unmarked symbols after the #. If any unmarked symbols remain to the right of the #, reject; if none are found, accept.”

A. $\{0,1\}^*#\{0,1\}^*$

B. $\{w#w | w \text{ in } \{0,1\}^*\}$

C. $\{#\}^*$

D. $\{0^n#1^n | n \geq 0\}$

E. None of the above
TM $M$ that accepts $\{0,1\}^*$

An implementation-level description is given by $M = \text{“On input } w:\text{”}$

A. Accept $w$

B. Sweep left across the tape, checking for the condition that $w$ consists of only 0’s and 1’s, until a blank is reached. If the condition is satisfied, accept $w$, otherwise reject $w$

C. If the first tape symbol is blank, accept. If not, sweep right across the tape, checking that for the condition that $w$ consists of only 0’s and 1’s. If so accept $w$; if not, reject $w$

D. None of the above
TM’s: High-level Descriptions

• As part of a high-level description of a TM M, M can call and run another already defined TM N as a subroutine (procedure)
• The input to TM N inside M should be specified, and should match what is expected by N
• You can also use FA’s and PDA’s as subroutines
The Decidable Languages are closed under Union.

Proof: Let \( M_1 \) and \( M_2 \) be deciders for \( L_1 \) and \( L_2 \). We show there is a decider \( M \) that decides \( L_1 \cup L_2 \) by giving a \textit{high-level} description of TM \( M \).

Construction: Let \( M = \) “On input \( w \):
1. Run \( M_1 \) on \( w \). If \( M_1 \) accepts \( w \), accept. If \( M_1 \) rejects \( w \), then go to 2.
2. Run \( M_2 \) on \( w \). If \( M_2 \) accepts \( w \), accept. If \( M_2 \) rejects \( w \), reject.”

• Correctness: Show \( M \) accepts \( w \) IFF \( M_1 \) accepts \( w \) or \( M_2 \) accepts \( w \)
• Conclusion: \( M \) is a TM that decides \( L_1 \cup L_2 \), therefore the Turing-decidable languages are closed under union. QED.
The Decidable Languages are closed under Intersection.

Proof: Let $M_1$ and $M_2$ be deciders for $L_1$ and $L_2$. We show there is a decider $M$ that decides $L_1 \cap L_2$ by giving a high-level description of TM $M$.

Construction: Let $M =$ “On input $w$:
1. Run $M_1$ on $w$. If $M_1$ accepts $w$, go to 2. If $M_1$ rejects $w$, then reject.
2. Run $M_2$ on $w$. If $M_2$ accepts $w$, accept. If $M_2$ rejects $w$, accept.”

• Correctness: $M$ accepts $w$ IFF $M_1$ accepts $w$ and $M_2$ accepts $w$

• Conclusion: $M$ is a TM that decides $L_1 \cap L_2$, therefore the Turing-decidable languages are closed under $\cap$. QED.

A. This proof is: 
B. Correct
C. Incorrect
D. Don’t know
Are we there yet?

INFINITE LOOPS
A Given Turing Machine $M$, run on a given string $w$, has 3 possible outcomes:

1. $M$ accepts $w$
2. $M$ rejects $w$
3. $M$ never halts on $w$, i.e., it “loops” forever

• Why do we have this 3$^{\text{rd}}$ behavior now, but didn’t with DFAs, NFAs nor PDAs?
Group exercise: Construct a TM that NEVER halts

1. Explain what it does in words, giving an implementation-level description.

2. Construct a state diagram.
Smart Printers

ENUMERATORS
Enumerators

TM with attached “printer”

- TM that starts with blank tape
- At any point, it may send string to printer to print
- \( L(E) = \) the set of strings that \( E \) eventually prints out
  - Strings may be printed in any order, and with repetitions
- If \( E \) does not halt, its language may be infinite
True or False?

• There is an enumerator E' whose language is the set of all strings of $\Sigma$, for any alphabet $\Sigma$.

  a) TRUE
  b) FALSE
  c) Don’t know
Th. 3.21. A language L is Turing-recognizable iff some enumerator enumerates L.

To Show if and only if:

1. If a language L is Turing-recognizable then some enumerator enumerates L.

2. If enumerator E enumerates language L then L is Turing-recognizable.