Today's learning goals

• Distinguish between a DFA, NFA, PDA, and a Turing machine
• Distinguish between the formal definition of a Turing machine, an implementation-level description, and a high-level description
• Trace the computation of a Turing machine using its transition function and configurations
• Describe the language of a TM
Turing machines

- Unlimited input
- Unlimited (read/write) memory
- Unlimited time
Turing machine computation

- Read/write head starts at leftmost position on tape
- Input string written on leftmost squares of tape, rest is blank
- Computation proceeds according to transition function:
  - Given current state of machine, and current symbol being read
    - the machine
      - transitions to new state
      - writes a symbol to its current position (overwriting existing symbol)
      - moves the tape head L or R
  - Computation ends if and when it enters either the accept or the reject state.
Language of a Turing machine

L(M) = \{ w \mid \text{computation of } M \text{ on } w \text{ halts after entering the accept state} \}

i.e. \ L(M) = \{ w \mid M \text{ is accepted by } w \}

Comparing TMs and PDAs, which of the following is true:
A. Both TMs and PDAs may accept a string before reading all of it.
B. A TM may only read symbols, whereas a PDA may write to its stack.
C. Both TMs and PDAs must read the string from left to right.
D. States in a PDA must be either accepting or rejecting, but in a TM may be neither.
E. I don't know.
Why is this model relevant?

• Division between program (CPU, state space) and data (memory) is a cornerstone of all modern computing.

• Unbounded memory is outer limits of what modern computers (PCs, quantum computers, DNA computers) can implement.

• Simple enough to reason about (and diagonalize against), expressive enough to capture modern computation.
Formal definition of TM

A Turing machine is a 7-tuple \((Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})\) where \(Q, \Sigma, \Gamma\) are all finite sets and

1. \(Q\) is the set of states
2. \(\Sigma\) is the input alphabet (not containing blank symbol)
3. \(\Gamma\) is the tape alphabet (including blank symbol as well as all symbols in \(\Sigma\))
4. \(\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}\) is the transition function
5. \(q_0 \in Q\) is the start state
6. \(q_{\text{accept}} \in Q\) is the accept state
7. \(q_{\text{reject}} \in Q\) is the reject state

\(q_{\text{reject}} \neq q_{\text{accept}}\)
Formal definition of TM

A Turing machine is a 7-tuple consisting of finite sets and

1. \( Q \) is the set of states
2. \( \Sigma \) is the input alphabet (\( \Sigma \) \( \subseteq \) \( \Gamma \))
3. \( \Gamma \) is the tape alphabet (\( \Gamma \) \( \subseteq \) \( \Sigma \))

4. \( \delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\} \) is the transition function
5. \( q_0 \in Q \) is the start state
6. \( q_{\text{accept}} \in Q \) is the accept state
7. \( q_{\text{reject}} \in Q \) is the reject state

What is the input of the transition function?

A. Current state and current character read
B. Current state and current character to write
C. Current state and next state
D. Current state and whether came from L or R
E. I don't know.
Formal definition of TM

A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where:

1. $Q$ is the set of states
2. $\Sigma$ is the input alphabet (finite sets and
3. $\Gamma$ is the tape alphabet (including blank symbol as well as all symbols in $\Sigma$)
4. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function
5. $q_0 \in Q$ is the start state
6. $q_{\text{accept}} \in Q$ is the accept state
7. $q_{\text{reject}} \in Q$ is the reject state

Are Turing machines deterministic or not?

A. Deterministic  
B. Nondeterministic  
C. I don't know
Configurations of a TM

- Current state
- Current tape contents
- Current location of read/write head

Current state is \( q \)
Current tape contents are \( uv \) (and then all blanks)
Current head location is first symbol of \( v \)
Configurations of a TM

- Current state
- Current tape contents
- Current location of read/write head

current state is q

current tape contents are uv (and then all blanks)

current head location is first symbol of v

Start configuration on w:
q₀ w

Accepting configuration:
u q_{acc} v

Rejecting configuration:
u q_{rej} v

Halting configuration: any configuration that is either rejecting or halting.
Transitioning between configurations

w is input, read/write head over the leftmost symbol of w

$q' = \delta(q, v_1)$

How does $uv$ compare to $u'v'$?
**Language of a TM**

\[
L(M) = \{ w \mid M \text{ accepts } w \} = \{ w \mid \text{there is a sequence of configurations of } M \\
\text{where } C_1 \text{ is start configuration of } M \text{ on input } w, \\
\text{each } C_i \text{ yields } C_{i+1} \text{ and } C_k \text{ is accepting configuration} \}
\]

"The language of M"

"The language recognized by M"
Deciders and recognizers

- L is **Turing-recognizable** if some Turing machine recognizes it.

- M is a **decider** TM if it halts on all inputs.

- L is **Turing-decidable** if some Turing machine that is a decider recognizes it.
An example

\[ L = \{ \text{w#w} | \text{w is in \{0,1\}}^* \} \]

We already know that \( L \) is

- not regular
- not context-free

We will prove that \( L \) is

Turing-decidable and therefore also Turing-recognizable
An example

\[ L = \{ w#w \mid w \text{ is in } \{0,1\}^* \} \]

Idea for Turing machine

• Zig-zag across tape to corresponding positions on either side of '#' to check whether these positions agree. If they do not, or if there is no '#', reject. If they do, cross them off.

• Once all symbols to the left of the '#' are crossed off, check for any symbols to the right of '#': if there are any, reject; if there aren't, accept.
An example

$L = \{ w#w | w \text{ is in } \{0,1\}^* \}$

Idea for Turing machine

- Zig-zag across tape to corresponding positions on either side of '#' to check whether these positions agree. If they do not, or if there is no '#', reject. If they do, cross them off.
- Once all symbols to the left of the '#' are crossed off, check for any symbols to the right of '#': if there are any, reject; if there aren't, accept.

Is this machine a **decider**?

A. Yes, because it reads the input string exactly once.
B. Yes, because it will halt (and either accept or reject) no matter what the input is.
C. No, because it sometimes rejects the input string.
D. No, because it will go in an infinite loop if there's no '#'.
E. I don't know.
Idea for Turing machine

Zig-zag across tape to corresponding positions on either side of '#' to check whether these positions agree. If they do not, or if there is no '#', reject. If they do, cross them off.

Once all symbols to the left of the '#' are crossed off, check for any symbols to the right of '#': if there are any, reject; if there aren't, accept.
**Fig 3.10 in Sipser**

*Some transitions omitted for readability*
Describing TMs

- **Formal definition**: set of states, input alphabet, tape alphabet, transition function, state state, accept state, reject state.

- **Implementation-level definition**: English prose to describe Turing machine head movements relative to contents of tape.

- **High-level description**: Description of algorithm, without implementation details of machine. As part of this description, can "call" and run another TM as a subroutine.
Deciding vs. recognizing

"If the input string is finite, then at some point, the TM has to be able to finish reading it. Therefore, infinite looping can only happen when the input takes up the whole TM tape (which is infinitely long)."

A. True
B. False
C. I don't know.
An example

Which of the following is an implementation-level description of a TM which decides the empty set?

M = "On input w:
A. reject."
B. sweep left across the tape until find a non-blank symbol. Then, reject."
C. sweep right across the tape until find a non-blank symbol. Then, reject."
D. If the first tape symbol is blank, accept. Otherwise, reject."
E. I don't know.
Closure

**Theorem**: The class of decidable languages over fixed alphabet $\Sigma$ is closed under union.

Proof: Let …

WTS …
Closure

**Theorem:** The class of decidable languages over fixed alphabet $\Sigma$ is closed under union.

Proof: Let $L_1$ and $L_2$ be languages and suppose $M_1$ and $M_2$ are TMs deciding these languages. We will define a new TM, $M$, via a high-level description. We will then show that $L(M) = L_1 \cup L_2$. $M$ is a decider.
Closure

**Theorem:** The class of decidable languages over fixed alphabet $\Sigma$ is closed under union.

Proof: Let $L_1$ and $L_2$ be languages and suppose $M_1$ and $M_2$ are TMs deciding these languages. Construct the TM $M$ as "On input $w$,
1. Run $M_1$ on input $w$. If $M_1$ accepts $w$, accept. Otherwise, go to 2.
2. Run $M_2$ on input $w$. If $M_2$ accepts $w$, accept. Otherwise, reject."

Proof of correctness soon, but first …
Theorem: The class of decidable languages over fixed alphabet $\Sigma$ is closed under union.

Proof: Let $L_1$ and $L_2$ be languages and suppose $M_1$ and $M_2$ are TMs deciding these languages. Construct the TM $M$ as "On input $w$, 
1. Run $M_1$ on input $w$. If $M_1$ accepts $w$, accept. Otherwise, go to 2.
2. Run $M_2$ on input $w$. If $M_2$ accepts $w$, accept. Otherwise, reject."

Could the same construction give us a proof that the class of recognizable languages is closed under union?

A. Yes, just replace $M_1$ and $M_2$ by TMs (instead of deciders)
B. Yes, but need to consider the case of $M_1$, $M_2$ rejecting $w$
C. No, but a different construction will work.
D. No, the class of recognizable languages is not closed under U.
E. I don't know.

Next class!
Closure

**Theorem:** The class of decidable languages over fixed alphabet $\Sigma$ is closed under union.

Proof: Let $L_1$ and $L_2$ be languages and suppose $M_1$ and $M_2$ are TMs deciding these languages. Construct the TM $M$ as "On input $w$,

1. Run $M_1$ on input $w$. If $M_1$ accepts $w$, accept. Otherwise, go to 2.
2. Run $M_2$ on input $w$. If $M_2$ accepts $w$, accept. Otherwise, reject."

**Correctness of construction:**

WTS $L(M) = L_1 \cup L_2$ and $M$ is a decider.
Reminders

• **HW5** due Friday May 6
• **RQ6** due Monday May 9
• **Exam 2** Wednesday May 11 (one week from tomorrow)
  • *Same time* 8pm-10pm *Same place* SOLIS 107
  • *New* seating chart: see Ted
  • *Review* session Monday May 9, 8pm-10pm, PETERSON 108