Today's learning goals

- Design TMs using different levels of descriptions.
- Determine whether a Turing machine is a decider.
- Prove properties of the classes of recognizable and decidable sets.

Announcements
- Group HW4 due Saturday
- Review Quiz “due” Sunday
- No class on Monday ← I’m still holding office hours
- Indiv HW5 due Tuesday
- Miles Jones Subbing Wednesday & Friday
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.
Language of a TM

$L(M) = \{ w \mid M \text{ accepts } w \}$

- If $w$ is in $L(M)$ then the computation of $M$ on $w$ halts and accepts.
- If the computation of $M$ on $w$ halts and rejects, then $w$ is not in $L(M)$.
- If the computation of $M$ on $w$ doesn't halt, then $w$ is not in $L(M)$.
Deciders and recognizers \( \text{Sipser p. 170 Defs 3.5 and 3.6} \)

- \( L \) is **recognized** by Turing machine \( M \) if \( L(M) = L \).

\* \( M \) is a **decider** if it is a Turing machine and halts on all inputs.

- \( L \) is **decided** by Turing machine \( M \) if \( M \) is a decider and \( L(M) = L \).

\( \{ \text{deciders} \} \subsetneq \{ \text{TMs} \} \)
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 right across the tape until find a non-blank symbol. Then, reject."
C. If the first tape symbol is blank, accept. Otherwise, reject."
D. More than one of the above.
E. I don't know.
Extension

• Give an implementation-level description of a Turing machine which recognizes (but does not decide) the empty set.

• Give a high-level description of this Turing machine.
Another example

Suppose $M_1$ and $M_2$ are Turing machines.

Consider the new TM $M = \text{"On input } w,\\n1. \text{ Run } M_1 \text{ on } w. \text{ If } M_1 \text{ rejects, rejects. If } M_1 \text{ accepts, go to 2.}\\n2. \text{ Run } M_2 \text{ on } w. \text{ If } M_2 \text{ accepts, accept. If } M_2 \text{ rejects, reject.}\"

What kind of construction is this?
A. Formal definition of TM
B. Implementation-level description of TM
C. High-level description of TM
D. I don't know.

What's $L(M)$?

Is $M$ a decider?
Classifying languages

A language $L$ is

**Turing-recognizable** if there is a TM $M$ such that $L(M) = L$ in other words, if there is some TM that recognizes it.

**Turing-decidable** if there is a TM $M$ such that $M$ is a decider and $L(M) = L$ in other words, if there is some TM that decides it.
Turing recognizable languages
Turing decidable languages
Context-free languages
Regular languages
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 over \( \Sigma \) 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 \( L(M) = L_1 \cup L_2 \) and that \( M \) always halts.

Conclude: \( L_1 \cup L_2 \) is decidable.
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.
For an arbitrary string \( w \),

\[ \text{WTS}_1 \quad \text{Assume} \quad \text{WE} L_1 \in U_L_2. \quad \text{WTS}_5 \quad M \text{ accepts } w \]

Run \( M \) on \( w \). That is, in step 1 we run \( M_1 \) on \( w \). If \( \text{WE} L_1 \), then \( \text{b/c } \text{L}_1 = \text{L}(M_1), \quad M_1 \text{ accepts } w \).

So step 1 says \( M \) also accepts \( w \). Otherwise \( \text{WE} L_2 \), so \( \text{WE} L_2 \). (b/c Case is \( \text{WE} L_1 \in U_L_2 \)) So \( M \) rejects \( w \). \( M \) moves to steps 2, runs \( M_2 \) on \( w \), and accepts \( w \).

\[ \text{WTS}_2 \quad \text{Assume} \quad \text{WE} L_1 \in U_L_2. \quad \text{WTS}_5 \quad M \text{ rejects } w \]

(yourself)
### Closure

<table>
<thead>
<tr>
<th>The class of decidable languages is closed under</th>
<th>The class of recognizable languages is closed under</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Union ✓</td>
<td>• Union</td>
</tr>
<tr>
<td>• Concatenation</td>
<td>• Concatenation</td>
</tr>
<tr>
<td>• Intersection</td>
<td>• Intersection</td>
</tr>
<tr>
<td>• Kleene star</td>
<td>• Kleene star</td>
</tr>
<tr>
<td>• Complementation</td>
<td></td>
</tr>
</tbody>
</table>

Good exercises – can’t use without proof! (Sipser 3.15, 3.16)
For next time

**Group HW4 due** Saturday, February 17

For Wednesday, pre-class reading: Section 3.2, pp. 181