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.

Group HW 4 Q1 suggestion

* Instead of general PDA ↔ CFG conversion
  Analyze language of specific CFG (c) or PDAs (b, c) ...
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

- L is **recognized** by Turing machine M if $L(M) = L$.
- M **recognizes** L if M is a Turing machine and $L(M) = L$.

- 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 $L(M) = L$.
- M **decides** L if M is a decider and $L(M) = L$.
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.
Context-free languages

Turing recognizable languages

Turing decidable languages

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

M = "On input w:

A. reject."  \( L(A) = \emptyset \)

B. sweep right across the tape until find a non-blank symbol. Then, reject."  \( L(B) = \emptyset \) \( L(A) = \emptyset \) \( L(B) \cup L(A) = \emptyset \) on \( \epsilon \).

C. If the first tape symbol is blank, accept. Otherwise, reject."  \( L(C) = \{ \epsilon \} \)

D. More than one of the above.

E. I don't know.
Machine $B$ on $\varepsilon$ loops.
So $\varepsilon \notin L(B)$

Machine $B$ on $w \neq \varepsilon$ halts and rejects.
So $w \notin L(B)$.
\( M \) decides the empty set means

1. \( M \) recognizes the empty set
   i.e. \( L(M) = \emptyset \)
   i.e. \( M \) does not accept any string

and

2. for all \( w \), compute \( M \) on \( w \) halts
Extension

• Give an implementation-level description of a Turing machine which \textbf{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 $M = "$On input $w$,
1. Run $M_1$ on $w$. If $M_1$ rejects, rejects. If $M_1$ accepts, go to 2.
2. Run $M_2$ on $w$. If $M_2$ accepts, accept. If $M_2$ 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.
Another example

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

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

Trace compn of $M$ on $w$. Assuming $w \in L(M_1) \cap L(M_2)$, i.e., $w \in L(M_1)$ so $M_1$ halts on $w$, and accepts. So in step 1 of $M$, our simulation at $M_1$ on $w$ will accept, go to step 2. Then similarly $M_2$ accepts $M$.

What's $L(M)$?

Is $M$ a decider?
Case \( \text{WE } L(M_1) \)

Since \( \text{WE } L(M_1) \), computation of \( M \) on \( w \) will reject loop if \( M \) rejects \( w \), so will \( M \) (in step 1).
if \( M \) loops on \( w \), so will \( M \) (in step 1).

Conclusion: if \( w \in L(M_1) \) then \( w \notin L(M) \)

Similarly: if \( w \in L(M_2) \) then \( w \notin L(M) \)
Closure

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

Proof: Let $M_1$, $M_2$ be deciders

WTS $L(M_1) \cup L(M_2)$ is decidable

Need $M < L(M) = L(M_1) \cup L(M_2)$. 

Previous: The class of recognizable languages is closed under intersection.
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.
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.

Where do we use decidability?
### Closure

**The class of decidable languages is closed under**
- Union ✓
- Concatenation
- Intersection
- Kleene star
- Complementation

**The class of recognizable languages is closed under**
- Union
- Concatenation
- Intersection ✓
- Kleene star

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

**Group HW4** due Saturday, May 12

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