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.

Sipser Section 3.1

Group WA Quartet suggestion: instead of using general CFG $\leftrightarrow$ PDA transformation from look and analyze language first, i.e.

(a) $G$ CFG $\rightarrow$ $L(G) = L$ $\rightarrow$ PDA for $L$.
(b, c) $M$ PDA $\rightarrow$ $L(M) = L$ $\rightarrow$ CFG for $L$. 
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 $L(M) = L$. 

*Sipser p. 170 Defs 3.5 and 3.6*
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

Regular languages

Turing recognizable languages

Turing decidable languages
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.

• Give a formal description for each of A, B, C
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 $w$,

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

What's $L(M)$?

Is $M$ a decider? i.e. does $M$ halt on each input?

Assume $M_1, M_2$ are deciders. Then $M$ is decider too.

But didn't assume $M_1, M_2$ deciders.

$L(M) = L(M_1) \cap L(M_2)$. (no assumptions about $M_1, M_2$)
Assume \( w \in L(M_1) \cap L(M_2) \).

Claim: \( w \in L(M) \)

Pf: On \( w \), \( M \) first (in step 1) simulates \( M_1 \) on \( w \). Since assumed \( w \in L(M_1) \), \( M_1 \) halts. It accepts on input \( w \) so \( M \) goes to 2 and simulates \( M_2 \) on \( w \).

By assumption, \( w \in L(M_2) \) so \( M_2 \) accepts \( w \). \( M \) is defined to accept as well.
WTS $L(M) \subseteq L(M_1) \cap L(M_2)$

i.e. $L(M_1) \cup L(M_2) \subseteq L(M)$

Proof: Assume $w \notin L(M_1)$ or $w \notin L(M_2)$

WTS $w \notin L(M)$. $M_1$ rejects $w$.

Case 1: $w \notin L(M_1)$ $\leq$ $M_1$ loops on $w$.

When run $M$ on $w$:
- if in step 1, $M_1$ running on $w$ rejects $w$, means $M$ rejects too $\triangleright$
- otherwise, in step 1, $M_1$ running on $w$ loops so $M$ running on $w$ loops (stays in step 1) and $M$ doesn't accept $w$, i.e. $w \notin L(M)$.
Closure

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

Proof: Let \( L_1, L_2 \) be decidable langs i.e. have \( M_1, M_2 \) are deciders \( L_i = \text{L}(M_i) \)

WTS ... \( L_1 \cup L_2 \) is decidable.

Want \( M \) s.t. \( \text{L}(M) = \text{L}(M_1) \cup \text{L}(M_2) \)

\[ M_1 \cup M_2 \text{ doesn't type check} \]
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.
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.
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

**GroupHW4 due** Saturday, May 12

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