Today's learning goals

- Prove Turing-recognizability using
  - Turing machines
  - Enumerators
- State and use the Church-Turing thesis.
- Explain what it means for a problem to be decidable.
- Justify the use of encoding.
- Give examples of decidable problems.
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.
Recognition vs. decision

- L is **Turing-recognizable** if some Turing machine recognizes it.
- L is **Turing-decidable** if some Turing machine that is a decider recognizes it. M is a **decider** TM if it halts on all inputs.

Which of the following is **true**?

A. If X recognizable then X is decidable.
B. If X is recognizable then its complement is recognizable.
C. If X is decidable then its complement is decidable.
D. If X is decidable then its complement is recognizable.
E. I don't know.
Enumerators

- What about machines that produce output rather than accept input?

Computation proceeds according to transition function.

At any point, machine may "send" a string to printer.

$L(E) = \{ w \mid E \text{ eventually, in finite time, prints } w \}$
"There is an enumerator whose language is the set of all strings over $\Sigma$."

A. True
B. False
C. Depends on $\Sigma$.
D. I don't know.
Set of all strings

"There is an enumerator whose language is the set of all strings over \( \Sigma \)."

A. True  
B. False  
C. Depends on \( \Sigma \).  
D. I don't know.  

Lexicographic ordering: order strings first by length, then dictionary order.  
(p. 14)
Theorem: A language L is Turing-recognizable iff some enumerator enumerates L.

Proof:
A. Assume L is enumerated by some enumerator. WTS L is Turing-recognizable.
B. Assume L is Turing-recognizable. WTS some enumerator enumerates it.
Recognition and enumeration *Sipser Theorem 3.21*

A. Assume the enumerator $E$ enumerates $L$. WTS $L$ is Turing-recognizable.

We'll use $E$ as a subroutine used by high-level description of Turing machine $M$ that will recognize $L$.

Define $M$ as follows: $M =$ "On input $w$,

1. Run $E$. Every time $E$ prints a string, compare it to $w$.
2. If $w$ ever appears as the output of $E$, accept."

**Correctness?**
B. Assume $L$ is Turing-recognizable. WTS some enumerator enumerates it.

Let $M$ be a TM that recognizes $L$. We'll use $M$ as a subroutine in a high-level description of enumerator $E$.

Let $s_1, s_2, \ldots$ be a list of all possible strings of $\Sigma^*$.

Define the enumerator $E$ as follows:

$E =$ "Repeat the following for each value of $i=1,2,3\ldots$
1. Run $M$ for $i$ steps on each input $s_1, \ldots, s_i$
2. If any of the $i$ computations of $M$ accepts, print out the accepted string."

Correctness?
Variants of TMs

- Scratch work, copy input, …
- Parallel computation
- Printing vs. accepting
- More flexible transition function
  - Can "stay put"
  - Can "get stuck"
  - *lots of examples in exercises to Chapter 3*

**Multiple tapes**

**Nondeterminism**

**Enumerators**

Also: wildly different models

- \(\lambda\)-calculus, Post canonical systems, URM\(\text{s}, \text{etc.}\)
Variants of TMs

- Scratch work, copy input, …
- Multiple tapes
- Parallel computation
- Printing
- More flexibility:
  - Can "stay put"
  - Can "get stuck"
  - Lots of examples in exercises to Chapter 3

Also: wildly different models
- $\lambda$-calculus, Post canonical systems, URM$s$, etc.

All these models are equally expressive...

capture the notion of "algorithm"
Algorithm

- Wikipedia "self-contained step-by-step set of operations to be performed"
- CSE 20 textbook "An algorithm is a finite sequence of precise instructions for performing a computation or for solving a problem."

Church-Turing thesis

Each algorithm can be implemented by some Turing machine.
Some algorithms

Examples of algorithms / algorithmic problems:
1. Recognize whether a string is a palindrome.
2. Reverse a string.
3. Recognize Pythagorean triples.
4. Compute the gcd of two positive integers.
5. Check whether a string is accepted by a DFA.
6. Convert a regular expression to an equivalent NFA.
7. Check whether the language of a PDA is infinite.
Some algorithms

Examples of algorithms / algorithmic problems:
1. Recognize whether a string is a palindrome.
2. Reverse a string.
3. Recognize Pythagorean triples.
4. Compute the gcd of two positive integers.
5. Check whether a string is accepted by a DFA.
6. Convert a regular expression to an equivalent NFA.
7. Check whether the language of a PDA is infinite.

Which of the following is true?
A. All these algorithms have inputs of the same type.
B. The inputs of each of these algorithms can be encoded as finite strings.
C. Some of these problems don't have algorithmic solutions.
D. I don't know.