Today's learning goals

- Trace high-level descriptions of algorithms for computational problems.
- Use counting arguments to prove the existence of unrecognizable (undecidable) languages.
- Use diagonalization in a proof of undecidability.

Reminder - Exam 2 next class (Friday)

* Review Session tonight (podcast)
* Exam2 practice Q solutions to be posted tmrw
* Seat assignments on Piazza
Computational problems

A computational problem is **decidable** iff the language encoding the problem instances is decidable.

*In Siper 4.1: The computational problems below*  

\[ A_{DFA}, A_{NFA}, A_{REX}, A_{CFG} \]  
\[ E_{DFA}, E_{NFA}, E_{REX}, E_{CFG} \]  
\[ EQ_{DFA}, EQ_{NFA}, EQ_{REX}, EQ_{CFG} \]

are all decidable
Undecidable?

• There are many ways to prove that a problem *is* decidable.

• How do we find (and prove) that a problem *is not* decidable?
Before we proved the Pumping Lemma …

We proved there was a set that was not regular because

Counting arguments

All sets of strings

$\mathcal{L}(\Sigma^*)$

All Regular Sets

Countable

Uncountable
Reminder: countable/ uncountable

Sets A and B have the **same size** $|A| = |B|$ means there is a function between them that is both one-to-one and onto.

### Countable (finite or same size as N)

- $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$
- $\Sigma^*$
- The set of all TMs

### Uncountable

- $\mathbb{R}$
- \{ infinite sequences over $\Sigma$\}
- $P(\Sigma^*)$

Sipser p. 202-204
Reminder: countable/ uncountable

Sets A and B have the same size $|A| = |B|$ means there is a function between them that is both one-to-one and onto.

What type of elements are in the set $P(\Sigma^*)$?

A. Strings  
B. Regular expressions  
C. Languages  
D. Sets of regular expressions  
E. I don’t know

Sipser p. 202-204
Why is the set of Turing-recognizable languages countable?

A. It's equal to the set of all TMs, which we showed is countable.
B. It's a subset of the set of all TMs, which we showed is countable.
C. Each Turing-recognizable language is associated with a TM, so there can be no more Turing-recognizable languages than TMs.
D. More than one of the above.
E. I don't know.
Counting arguments

All sets of strings

Countable

All Turing-recognizable sets

Turing-dec

Uncountable

Is the set of Turing-decidable sets countable?

\[ \{ \text{Turing-dec} \} \subseteq \{ \text{Turing-rec} \} \]
Satisfied?

- Maybe not …

- What's a specific example of a language that is not Turing-recognizable? or not Turing-decidable?

  \[\text{Cantor's diagonalization}\]

  - Idea: consider a set that, were it to be Turing-decidable, would have to "talk" about itself, and contradict itself!
Recall $A_{DFA} = \{ <B, w> \mid B \text{ is a DFA and } w \text{ is in } L(B) \}$

$A_{TM} = \{ <M, w> \mid M \text{ is a TM and } w \text{ is in } L(M) \}$

What is $A_{TM}$?

A. A Turing machine whose input is codes of TMs and strings.
B. A set of pairs of TMs and strings.
C. A set of strings that encode TMs and strings.
D. Not well defined.
E. I don't know.
Define the TM \( N = \) "On input \(<M,w>\):
1. Simulate \( M \) on \( w \).
2. If \( M \) accepts, accept. If \( M \) rejects, reject."

Which of the following statements is true?

A. \( N \) decides \( A_{TM} \)
B. \( N \) recognizes \( A_{TM} \)
C. \( N \) always halts
D. More than one of the above.
E. I don't know
$A_{TM}$

$A_{DFA} = \{ <B,w> \mid B \text{ is a DFA and } w \text{ is in } L(B) \}$

Decider for this set simulates arbitrary DFA

$A_{TM} = \{ <M,w> \mid M \text{ is a TM and } w \text{ is in } L(M) \}$

Decider for this set simulates arbitrary TMs.

What happens when it simulates itself? WTHS uh ohh
Diagonalization proof: $A_{TM}$ not decidable

Assume, *towards a contradiction*, that it is.

Call $M_{ATM}$ the decider for $A_{TM}$:

For every TM $M$ and every string $w$,

- Computation of $M_{ATM}$ on $<M, w>$ halts and accepts if $w$ is in $L(M)$.
- Computation of $M_{ATM}$ on $<M, w>$ halts and rejects if $w$ is not in $L(M)$.

*Goal: find contradiction*
Diagonalization proof: $A_{TM}$ not decidable

Assume, towards a contradiction, that $M_{ATM}$ decides $A_{TM}$

Define the TM $D = \text{"On input } <M>:\text{"}$

1. Run $M_{ATM}$ on $<M, <M>>$.  
2. If $M_{ATM}$ accepts, reject; if $M_{ATM}$ rejects, accept.
Diagonalization proof: $A_{\text{TM}}$ not decidable \textit{Sipser 4.11}

Assume, towards a contradiction, that $M_{\text{ATM}}$ decides $A_{\text{TM}}$.

Define the TM $D = "\text{On input } <M>:"$

1. Run $M_{\text{ATM}}$ on $<M, <M>>$.
2. If $M_{\text{ATM}}$ accepts, reject; if $M_{\text{ATM}}$ rejects, accept.

Which of the following computations halt?

A. Computation of $D$ on $<X>$ where $X$ is TM which accepts string $w$ if first character is 0 and loops otherwise.
B. Computation of $D$ on $<Y>$ where $Y$ is TM with $L(Y) = \Sigma^*$
C. Computation of $D$ on $<D>$
D. All of the above.
Diagonalization proof: $A_{TM}$ not decidable

Assume, towards a contradiction, that $M_{ATM}$ decides $A_{TM}$

Define the TM $D =$ "On input $<M>$:
1. Run $M_{ATM}$ on $<M, <M>>$.
2. If $M_{ATM}$ accepts, reject; if $M_{ATM}$ rejects, accept."

Consider running $D$ on input $<D>$. Because $D$ is a decider:
- either computation halts and accepts …
- or computation halts and rejects …
Case 1: D accepts $\langle D \rangle$

Running D: in step 1, Marn gets input $\langle D, \langle D \rangle \rangle$ and accepts because $\langle D \rangle \in L(D)$ so in step 2 D rejects $\langle D \rangle$.

Case 2: D rejects $\langle D \rangle$

Running D: in step 1, Marn gets input $\langle D, \langle D \rangle \rangle$ and rejects because $\langle D \rangle \notin L(D)$ so in step 2 D accepts $\langle D \rangle$. 
Diagonalization proof: $A_{TM}$ not decidable

Assume, towards a contradiction, that $M_{ATM}$ decides $A_{TM}$.

Define the TM $D =$ “On input $<M>$:
1. Run $M_{ATM}$ on $<M, <M>>$.
2. If $M_{ATM}$ accepts, reject; if $M_{ATM}$ rejects, accept.”

Consider running $D$ on input $<D>$. Because $D$ is a decider:
- either computation halts and accepts …
- or computation halts and rejects …

Diagonalization???

Self-reference

"Is $<D>$ an element of $L(D)$?"
$A_{TM}$

- Recognizable
- Not decidable

**Fact** (from discussion section): A language is decidable iff it and its complement are both recognizable.

**Corollary**: The complement of $A_{TM}$ is **unrecognizable**.
Do we have to diagonalize?

• Next time (after exam): comparing difficulty of problems.
Next time: Exam 2

Bring ID, note card

Check seat maps before coming to class