CSE 105
THEORY OF COMPUTATION

Fall 2016

http://cseweb.ucsd.edu/classes/fa16/cse105-abc/
Today's learning goals

- Justify the use of encoding.
- Give examples of decidable problems.
- Use counting arguments to prove the existence of unrecognizable (undecidable) languages.
- Determine and prove whether sets are countable.
- Use diagonalization in a proof of uncountability.
- Use diagonalization in a proof of undecidability.
Is it true that the intersection of a Turing-recognizable set and a co-Turing-recognizable set is decidable?
A. Yes, by the Venn diagram.
B. Yes, by the argument we did last class.
C. No.
D. I don't know.
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.
Computational problems

A computational problem is **decidable** iff the language encoding the problem instances is decidable.
Encoding input for TMs

- By definition, TM inputs are **strings**
- To define TM M:
  ```
  “On input w ... 
  1. ...
  2. ..
  3. ..
  ```

For inputs that aren’t strings, we have to **encode the object** (represent it as a string) first.

**Notation:**

- `<O>` is the string that represents (encodes) the object O
- `<O1, ..., On>` is the single string that represents the tuple of objects O1, ..., On
Computational problems

Sample computational problems and their encodings:

- $A_{DFA}$ "Check whether a string is accepted by a DFA."
  $\{ \langle B, w \rangle \mid B \text{ is a DFA over } \Sigma, w \in \Sigma^*, \text{ and } w \text{ is in } L(B) \}$

- $E_{DFA}$ "Check whether the language of a DFA is empty."
  $\{ \langle A \rangle \mid A \text{ is a DFA over } \Sigma, L(A) \text{ is empty} \}$

- $EQ_{DFA}$ "Check whether the languages of two DFA are equal."
  $\{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFA over } \Sigma, L(A) = L(B) \}$

FACT: all of these problems are decidable!
Proving decidability

Claim: $A_{DFA}$ is decidable

Proof: WTS that \{ <B,w> | B is a DFA over $\Sigma$, w in $\Sigma^*$, and w is in $L(B)$ \} is decidable.

Step 1: construction

How would you check if w is in $L(B)$?
Proving decidability

Claim: $A_{DFA}$ is decidable

Proof: WTS that $\{ <B,w> \mid B$ is a DFA over $\Sigma$, $w \in \Sigma^*$, and $w \in L(B) \}$ is decidable.

Step 1: construction

Define TM $M$ by: $M_1 = "On \text{ input } <B,w>"

1. Check whether $B$ is a valid encoding of a DFA and $w$ is a valid input for $B$. If not, reject.
2. Simulate running $B$ on $w$ (by keeping track of states in $B$, transition function of $B$, etc.)
3. When the simulation ends, by finishing to process all of $w$, check current state of $B$: if it is final, accept; if it is not, reject."
Proving decidability

Step 1: construction
Define TM $M_1$ = "On input $<B,w>$
1. Check whether $B$ is a valid encoding of a DFA and $w$ is a valid input for $B$. If not, reject.
2. Simulate running $B$ on $w$ (by keeping track of states in $B$, transition function of $B$, etc.)
3. When the simulation ends, by finishing to process all of $w$, check current state of $B$: if it is final, accept; if it is not, reject."

Step 2: correctness proof
WTS (1) $L(M_1) = A_{DFA}$ and (2) $M_1$ is a decider.
Proving decidability

Claim: \( E_{DFA} \) is decidable

Proof: WTS that \( \{ <A> | A \text{ is a DFA over } \Sigma, \, L(A) \text{ is empty} \} \) is decidable.

Idea: give high-level description

Step 1: construction

What condition distinguishes between DFA that accept *some* string and those that don't accept *any*?

A. \(<A>\) is in \( E_{DFA} \) iff A's initial state is accepting.
B. \(<A>\) is in \( E_{DFA} \) iff A's set of accepting states is empty.
C. \(<A>\) is in \( E_{DFA} \) iff A is the empty set.
D. None of the above.
E. I don't know.
Claim: $E_{\text{DFA}}$ is decidable

Proof: WTS that $\{ <A> | A \text{ is a DFA over } \Sigma, L(A) \text{ is empty} \}$ is decidable.

Idea: give high-level description

Step 1: construction

What condition distinguishes between DFA that accept *some* string and those that don't accept *any*?
Proving decidability

Claim: $E_{\text{DFA}}$ is decidable

Proof: WTS that $\{ <A> | A \text{ is a DFA over } \Sigma, L(A) \text{ is empty} \}$ is decidable. Idea: give high-level description

Step 1: construction

Define TM $M_2$ by: $M_2 = \text{"On input } <A>:\"
1. Check whether $A$ is a valid encoding of a DFA; if not, reject.
2. Mark the start state of $A$.
3. Repeat until no new states get marked:
   i. Loop over states of $A$ and mark any unmarked state that has an incoming edge from a marked state.
4. If no final state of $A$ is marked, accept; otherwise, reject.
Proving decidability

Step 1: construction
Define TM $M_2$ by: $M_2 = "On input <A>:"
1. Check whether $A$ is a valid encoding of a DFA; if not, reject.
2. Mark the state $state$ of $A$.
3. Repeat until no new states get marked:
   i. Loop over states of $A$ and mark any unmarked state that has an incoming edge from a marked state.
4. If no final state of $A$ is marked, accept; otherwise, reject.

Step 2: correctness proof
WTS (1) $L(M_2) = E_{DFA}$ and (2) $M_2$ is a decider.
Proving decidability

Claim: $\text{EQ}_{\text{DFA}}$ is decidable

Proof: WTS that $\{ <A, B> \mid A, B$ are DFA over $\Sigma$, $L(A) = L(B) \}$ is decidable. Idea: give high-level description

Step 1: construction

Will we be able to simulate $A$ and $B$?
What does set equality mean?
Can we use our previous work?
Proving decidability

Claim: \( \text{EQ}_{\text{DFA}} \) is decidable

Proof: WTS that \( \{ \langle A, B \rangle \mid A, B \text{ are DFA over } \Sigma, L(A) = L(B) \} \) is decidable. Idea: give high-level description

Step 1: construction

Will we be able to simulate? 
What does set equality mean? 
Can we use our previous work?

\[ X = Y \iff (X \cap Y^c) \cup (Y \cap X^c) = \emptyset \]
Proving decidability

Claim: \( \text{EQ}_{\text{DFA}} \) is decidable

Proof: WTS that \( \{ <A, B> | A, B \text{ are DFA over } \Sigma, L(A) = L(B) \} \) is decidable. Idea: give high-level description

Step 1: construction

\[ X = Y \iff (X \cap Y^c) \cup (Y \cap X^c) = \emptyset \]

Very high-level:

Build new DFA recognizing symmetric difference of \( A, B \). Check if this set is empty.
Claim: \( \text{EQ}_{\text{DFA}} \) is decidable

Proof: WTS that \( \{ \langle A, B \rangle \mid A, B \text{ are DFA over } \Sigma, L(A) = L(B) \} \) is decidable. Idea: give high-level description

Step 1: construction

Define TM \( M_3 \) by: \( M_3 = \) "On input \( \langle A, B \rangle \):

1. Check whether \( A, B \) are valid encodings of DFA; if not, reject.
2. Construct a new DFA, \( D \), from \( A, B \) using algorithms for complementing, taking unions of regular languages such that \( L(D) = \) symmetric difference of \( A \) and \( B \).
3. Run machine \( M_2 \) on \( \langle D \rangle \).
4. If it accepts, accept; if it rejects, reject."

Proving decidability

Step 1: construction
Define TM $M_3$ by: $M_3 = \text{"On input } <A,B>:\$

1. Check whether $A,B$ are valid encodings of DFA; if not, reject.
2. Construct a new DFA, $D$, from $A,B$ using algorithms for complementing, taking unions of regular languages such that $L(D) = \text{symmetric difference of } A \text{ and } B$.
3. Run machine $M_2$ on $<D>$.
4. If it accepts, accept; if it rejects, reject.

Step 2: correctness proof
WTS (1) $L(M_3) = EQ_{DFA}$ and (2) $M_3$ is a decider.
Techniques

- **Subroutines**: can use decision procedures of decidable problems as subroutines in other algorithms
  - $A_{DFA}$
  - $E_{DFA}$
  - $EQ_{DFA}$

- **Constructions**: can use algorithms for constructions as subroutines in other algorithms
  - Converting DFA to DFA recognizing complement (or Kleene star).
  - Converting two DFA/NFA to one recognizing union (or intersection, concatenation).
  - Converting NFA to equivalent DFA.
  - Converting regular expression to equivalent NFA.
  - Converting DFA to equivalent regular expression.
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 show that all sets of strings are either countable or uncountable.
Counting arguments

Recall: sets $A$ and $B$ have the same size, $|A| = |B|$ means there is a one-to-one and onto function between them.

A set is countable iff it is either

- finite (has the same size as $\{0, 1, \ldots, n\}$ for some nonnegative integer $n$), or
- has the same size as $\mathbb{N}$ (can list all and only the elements of the list in a sequence)
Counting arguments

Which of the following is true?

A. Any two infinite sets have the same size.
B. If A is a strict subset of B and then A and B do not have the same size.
C. If A is a subset of B and B is countable, then A is countable.
D. If A is countable then AxA is not countable.
E. I don't know.
Countable sets

Some examples:

\( \mathbb{N} \)

\( \mathbb{Z} \)

\( \mathbb{Q} \)

\( \{0,1\}^* \)

\( \Sigma^* \) for any alphabet \( \Sigma \)

Corollary: The set of all TMs is countable.  

Proof Idea: \(|\{M: M \text{ is a TM}\}| = |\{<M>: M \text{ is a TM}\}| \) and 

\(<M> \) is a string over the alphabet \( \{0,1,\_,(,),\ldots\} \).
Uncountable sets

Some examples:

\( \mathbb{R} \)

\([0, 1]\)

\{ infinite sequences of 0s and 1s \}

\( P(\{0, 1\}^*) \). i.e. collection of all languages over \( \{0, 1\}^* \)

Diagonalization Proof: Assume towards a contradiction that the set is countable. This gives a correspondence with \( \mathbb{N} \), but we can derive a contradiction.
Countable & Uncountable

\{0,1\}^* \text{ is countable}

\mathcal{P}(\{0,1\}^*) \text{ is uncountable}

\begin{align*}
\varepsilon \\
0 \\
1 \\
00 \\
01 \\
10 \\
11 \\
000 \\
\ldots
\end{align*}

There can be no such (exhaustive) list.
Proof that $P(\{0,1\}^*)$ not countable

Diagonalization Proof: Assume towards a contradiction that the set is countable. This gives a correspondence with $\mathbb{N}$, but we can derive a contradiction.

<table>
<thead>
<tr>
<th>n</th>
<th>f(n)</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$A_1$</td>
<td>$0$ in $A$ iff $0$ is not in $A_1$</td>
</tr>
<tr>
<td>2</td>
<td>$A_2$</td>
<td>$00$ in $A$ iff $00$ is not in $A_2$</td>
</tr>
<tr>
<td>3</td>
<td>$A_3$</td>
<td>$0^3$ in $A$ iff $0^3$ is not in $A_3$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Given the function $f$, define $A$ so it couldn't be in the image of $f$. 

Define $A$ so it couldn't be in the image of $f$. 

...
Proof that $P(\{0,1\}^*)$ not countable

<table>
<thead>
<tr>
<th>n</th>
<th>f(n)</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$A_1$</td>
<td>0 in $A$ iff 0 is not in $A_1$</td>
</tr>
<tr>
<td>2</td>
<td>$A_2$</td>
<td>00 in $A$ iff 00 is not in $A_2$</td>
</tr>
<tr>
<td>3</td>
<td>$A_3$</td>
<td>$0^3$ in $A$ iff $0^3$ is not in $A_3$</td>
</tr>
<tr>
<td>..</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

A is defined by "$0^n$ is in $A$ iff $0^n$ is not in $A_n$"

BUT since $A$ is a set of strings, it is the image of some int $c$.

Is $0^c$ in $A$?
Proof that $P(\{0,1\}^*)$ not countable

A is defined by "0\(_n\) is in A iff 0\(_n\) is not in A\(_n\)"

BUT since A is a set of strings, it is the image of some int c.

Is 0\(_c\) in A?

Diagonalization???

Self-reference

"Is 0\(_c\) an element of f(c)??"
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.
Satisfied?

- Maybe not …

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

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

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

What is $A_{\text{TM}}$?

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 = \text{"On input } <M,w>:\n1. \text{ Simulate } M \text{ on } w.\n2. \text{ If } M \text{ accepts, accept. If } M \text{ rejects, reject."}
Define the TM $N = "On input \langle M, w \rangle:\"
1. Simulate $M$ on $w$.
2. If $M$ accepts, accept. If $M$ rejects, reject."

What is $L(N)$?
A. $A_{TM}$
B. Some superset of $A_{TM}$
C. $\{\langle M, w \rangle | M$ is a TM and $w$ is a string$\}$
D. I don't know.
\( A_{TM} \)

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

Define the TM \( N = \) "On input \( <M,w> \):"

1. Simulate \( M \) on \( w \).
2. If \( M \) accepts, accept. If \( M \) rejects, reject."

Which statement is true?
A. \( N \) decides \( A_{TM} \)
B. \( N \) recognizes \( A_{TM} \)
C. \( N \) always halts
D. I don't know.
\[ A_{TM} = \{ <M,w> \mid M \text{ is a TM and } w \text{ is in } L(M) \} \]

Define the TM \( N = " \text{On input } <M,w>:\)
1. Simulate \( M \) on \( w \).
2. If \( M \) accepts, accept. If \( M \) rejects, reject."

Conclude: \( A_{TM} \) is Turing-recognizable.

Is it decidable?
Diagonalization proof: $A_{TM}$ not decidable

Assume, towards a contradiction, that it is.

I.e. let $M_{ATM}$ be a Turing machine such that for every TM $M$ and every string $w$,

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

Sipser 4.11
If $N$ is TM with $L(N) = \{ w \mid w \text{ starts with } 0 \}$ and $N$ does not halt on all strings not in $L(N)$, what is result of computation of $M_{ATM}$ on $<N, 11>$?

A. $M_{ATM}$ halts and accepts.
B. $M_{ATM}$ halts and rejects.
C. $M_{ATM}$ loops.
D. I don't know.

I.e. let $M_{ATM}$ be a Turing machine such that 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 $L(M)$. 
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."
Diagonalization proof: $A_{TM}$ not decidable

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

Define the TM $D = \text{"On input } \langle M \rangle:\$

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

Is $D$ a decider?
A. Yes: it's a TM that always halts.
B. No: it's a well-defined TM but may loop.
C. No: it's not even a well-defined TM.
D. I don't know.
Diagonalization proof: $A_{TM}$ not decidable \(\text{Sipser 4.11}\)

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

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

If $M_0$ is a TM with $L(M_0) = \emptyset$, what is result of computation of $D$ with input $<M_0>$?
A. Halt and accept.
B. Halt and reject.
C. Loop.
D. I don't know.
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."

If $M_1$ is a TM with $L(M_1) = \Sigma^*$, what is result of computation of $D$ with input $<M_1>$?
A. Halt and accept.
B. Halt and reject.
C. Loop.
D. I don't know.
Diagonalization proof: $A_{TM}$ not decidable \textit{Sipser 4.11}

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 proof: $A_{TM}$ not decidable

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

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

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 $L(D)$?"
Regular
Context-Free
Turing-
Recognizable

\{a^n b^n | n \geq 0\}

\{a^n b^n a^n | n \geq 0\}

A_{TM}

??
Do we have to diagonalize?

- Next time: undecidability proofs without diagonalization (or counting).