BU CS 332 – Theory of Computation

Lecture 14:

- Unrecognizability
- Undecidability

Reading: Sipser Ch. 4

Ran Canetti October 27, 2020 A general theorem about set sizes

Theorem: Let X be a set. Then the power set P(X) does **not** have the same size as X.

Proof: Assume for the sake of contradiction that there is a correspondence $f: X \rightarrow P(X)$

Construct a set $S \in P(X)$ that cannot be the output f(x) for any $x \in X$:

$$S = \{x \in X \mid x \notin f(x)\}$$

If S = f(y) for some $y \in X$,

then $y \in S$ if and only if $y \notin S$

Diagonalization argument

Assume a correspondence $f: X \to P(X)$

x	$x_1 \in f(x)?$	$x_2 \in f(x)$?	$x_3 \in f(x)$?	$x_4 \in f(x)$?	
<i>x</i> ₁	Y	N	Y	Y	
<i>x</i> ₂	N	N	Y	Y	
<i>x</i> ₃	Y	Y	Y	N	
<i>x</i> ₄	N	N	Y	N	
:					*•.

Define S by flipping the diagonal: Put $x_i \in S \iff x_i \notin f(x_i)$

An Existential Proof

Theorem: There exists an unrecognizable language over {0, 1}

Proof:

Set of all Turing machines: $X \subseteq \{0, 1\}^*$ Set of all languages over $\{0, 1\}$ = all subsets of $\{0, 1\}^*$ = P(X)

There are more languages than there are TMs!

Questions

• Are there languages that are recognizable but not decidable?



Questions

- Are there languages that are recognizable but not decidable?
- Are there any languages of interest that are unrecognizable/undecidable?

A Specific Undecidable Language

 $A_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts input } w \}$ Theorem: A_{TM} is undecidable

Proof: Assume for the sake of contradiction that TM H decides A_{TM} :

$$H(\langle M, w \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ accepts } w \\ \text{reject} & \text{if } M \text{ does not accept } w \end{cases}$$

Diagonalization: Use *H* to check what *M* does when given as input its own description...and do the opposite

A Specific Undecidable Language

 $A_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts input } w \}$ Suppose *H* decides A_{TM}

Consider the following TM D.

On input $\langle M \rangle$ where M is a TM:

- 1. Run *H* on input $\langle M, \langle M \rangle \rangle$
- 2. If *H* accepts, reject. If *H* rejects, accept.

Question: What does D do on input $\langle D \rangle$?

How is this diagonalization?



How is this diagonalization?

TM M	$M(\langle M_1 \rangle)?$	$M(\langle M_2 \rangle)?$	$M(\langle M_3 \rangle)?$	$M(\langle M_4\rangle)?$	
<i>M</i> ₁	Y	N	Y	Y	
<i>M</i> ₂	N	N	Y	Y	
<i>M</i> ₃	Y	Y	Y	N	
<i>M</i> ₄	N	N	Y	N	
:					*••

D accepts input $\langle M_i \rangle \iff M_i$ does not accept input $\langle M_i \rangle$

How is this diagonalization?

TM M	$M(\langle M_1 \rangle)?$	$M(\langle M_2 \rangle)?$	$M(\langle M_3 \rangle)?$	$M(\langle M_4 \rangle)?$		$D(\langle D \rangle)?$
<i>M</i> ₁	Y	N	Y	Y		
<i>M</i> ₂	N	N	Y	Y		
<i>M</i> ₃	Y	Y	Y	Ν		
<i>M</i> ₄	N	N	Y	N		
:					*••	
D						

D accepts input $\langle M_i \rangle \iff M_i$ does not accept input $\langle M_i \rangle$

Classes of Languages: updated view



A specific unrecognizable Language

Theorem: A language L is decidable if and only if L and \overline{L} are both Turing-recognizable.

Proof:

A specific unrecognizable Language

Theorem: A language L is decidable if and only if L and \overline{L} are both Turing-recognizable.

Corollary: If *L* is Turing-recognizable and undecidable then \overline{L} is not Turing-recognizable.

A specific unrecognizable Language

Theorem: A language L is decidable if and only if L and \overline{L} are both Turing-recognizable.

Corollary: If *L* is Turing-recognizable and undecidable then \overline{L} is not Turing-recognizable.

Define:

- **R** = decidable languages
- *RE* = Turing-recognizable languages
- **co** $RE = \{L \mid \overline{L} \text{ is Turing recognizable}\}$

Classes of Languages: updated view



Enumerators

TMs are equivalent to...

- TMs with "stay put"
- TMs with 2-way infinite tapes
- Multi-tape TMs
- Nondeterministic TMs
- Random access TMs
- Enumerators

• • •

Enumerators



- Starts with two blank tapes
- Prints strings to printer
- $L(E) = \{ \text{strings eventually printed by } E \}$
- May never terminate (even if language is finite)
- May print the same string many times

Enumerator Example

- 1. Initialize c = 1
- 2. Repeat forever:
 - Calculate $s = c^2$ (in binary)
 - Send *s* to printer
 - Increment *c*

What language can an enumerator generate?



Enumerable = Turing-Recognizable

Theorem: A language is Turing-recognizable ⇔ some enumerator enumerates it

 \Leftarrow Start with an enumerator *E* for *A* and give a TM

Enumerable = Turing-Recognizable

Theorem: A language is Turing-recognizable ⇔ some enumerator enumerates it

 \Rightarrow Start with a TM *M* for *A* and give an enumerator

Reductions

Scientists vs. Engineers

A computer scientist and an engineer are stranded on a desert island. They find two palm trees with one coconut on each. The engineer climbs a tree, picks a coconut and eats.



The computer scientist climbs the second tree, picks a coconut, climbs down, climbs up the first tree and places it there, declaring success.

"Now we've reduced the problem to one we've already solved."



A reduction from problem A to problem B is an algorithm for problem A which uses an algorithm for problem B as a subroutine

If such a reduction exists, we say "A reduces to B"

Two uses of reductions

Positive uses: If A reduces to B and B is decidable, then A is also decidable

 $EQ_{\text{DFA}} = \{ \langle D_1, D_2 \rangle | D_1, D_2 \text{ are DFAs and } L(D_1) = L(D_2) \}$ Theorem: EQ_{DFA} is decidable Proof: The following TM decides EQ_{DFA}

On input $\langle D_1, D_2 \rangle$, where $\langle D_1, D_2 \rangle$ are DFAs:

- 1. Construct a DFA D that recognizes the symmetric difference $L(D_1) \Delta L(D_2)$
- 2. Run the decider for $E_{\rm DFA}$ on $\langle D \rangle$ and return its output

Two uses of reductions

Negative uses: If A reduces to B and A is undecidable, then B is also undecidable

 $A_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts input } w \}$ Suppose *H* decides A_{TM}

Consider the following TM D. On input $\langle M \rangle$ where M is a TM:

- 1. Run *H* on input $\langle M, \langle M \rangle \rangle$
- 2. If *H* accepts, accept. If *H* rejects, reject.

Claim: *D* decides $SA_{TM} = \{\langle M \rangle \mid M \text{ is a TM that accepts on input } \langle M \rangle \}$

Two uses of reductions

Negative uses: If A reduces to B and A is undecidable, then B is also undecidable

Proof template:

- 1. Suppose to the contrary that *B* is decidable
- 2. Using B as a subroutine, construct an algorithm deciding A
- 3. But *A* is undecidable. Contradiction!

Halting Problem

 $HALT_{TM} = \{ \langle M, w \rangle | M \text{ is a TM that halts on input } w \}$

Theorem: *HALT*_{TM} is undecidable

Proof: Suppose for contradiction that there exists a decider *H* for $HALT_{TM}$. We construct a decider for A_{TM} as follows: On input $\langle M, w \rangle$:

1 Due II au instat / 1

- 1. Run *H* on input $\langle M, w \rangle$
- 2. If *H* rejects, reject
- 3. If *H* accepts, simulate *M* on *w*
- 4. If *M* accepts, accept. Otherwise, reject

This is a reduction from $A_{\rm TM}$ to $HALT_{\rm TM}$

Empty language testing for TMs

$$E_{\text{TM}} = \{ \langle M \rangle | M \text{ is a TM and } L(M) = \emptyset \}$$

Theorem: *E*_{TM} is undecidable

Proof: Suppose for contradiction that there exists a decider R for E_{TM} . We construct a decider for A_{TM} as follows:

- On input $\langle M, w \rangle$:
- 1. Run *R* on input ???

This is a reduction from $A_{\rm TM}$ to $E_{\rm TM}$

Empty language testing for TMs

$$E_{\text{TM}} = \{ \langle M \rangle | M \text{ is a TM and } L(M) = \emptyset \}$$

Theorem: *E*_{TM} is undecidable

Proof: Suppose for contradiction that there exists a decider R for $E_{\rm TM}$. We construct a decider for $A_{\rm TM}$ as follows:

On input $\langle M, w \rangle$:

1. Construct a TM *M*' as follows:

2. Run *R* on input $\langle M' \rangle$

3. If *R* , accept. Otherwise, reject

This is a reduction from $A_{\rm TM}$ to $E_{\rm TM}$

Context-free language testing for TMs

 $CFL_{TM} = \{\langle M \rangle | M \text{ is a TM and } L(M) \text{ is context} - \text{free} \}$ **Theorem:** CFL_{TM} is undecidable

Proof: Suppose for contradiction that there exists a decider R for CFL_{TM} . We construct a decider for A_{TM} as follows: On input $\langle M, w \rangle$:

1. Construct a TM *M*' as follows:

2. Run *R* on input $\langle M' \rangle$

3. If *R* accepts, accept. Otherwise, reject

This is a reduction from $A_{\rm TM}$ to $CFL_{\rm TM}$

Context-free language testing for TMs

 $CFL_{TM} = \{\langle M \rangle | M \text{ is a TM and } L(M) \text{ is context} - \text{free} \}$ **Theorem:** CFL_{TM} is undecidable

Proof: Suppose for contradiction that there exists a decider R for CFL_{TM} . We construct a decider for A_{TM} as follows: On input $\langle M, w \rangle$:

1. Construct a TM *M*' as follows:

M' = "On input x, $1. \text{ If } x \in \{0^n 1^n 2^n \mid n \ge 0\}, \text{ accept}$ 2. Run TM M on input w 3. If M accepts, accept." $2. \text{ Run } R \text{ on input } \langle M' \rangle$ 3. If R accepts, accept. Otherwise, reject

This is a reduction from A_{TM} to CFL_{TM}