BU CS 332 – Theory of Computation

Lecture 16:

- Test 2 Review

Mark Bun

November 2, 2021

"All flying octopuses are purple" True

$\iff \forall x \in \{\text{flying octopuses}\}$, x is purple

$\iff \forall x \in \phi$, x is purple

<u>In general</u>: "$\forall x \in \phi$, $p(x)$" is true

p ↑ boolean predicate

<u>E.g.</u>: $\forall x \in \phi$, x contains substring "zzz" is true

Test 2 Topics

Turing Machines (3.1, 3.3)

- Know the three different "levels of abstraction" for defining Turing machines and how to convert between them: Formal/state diagram, implementation-level, and high-level
- Know the definition of a configuration of a TM and the formal definition of how a TM computes
- Know how to "program" Turing machines by giving state diagrams and implementation-level descriptions
- Understand the Church-Turing Thesis

TM Variants (3.2)

- Understand the following TM variants: TM with stay-put, TM with two-way infinite tape, Multi-tape TMs, Nondeterministic TMs
- Know how to give a simulation argument (implementation-level and high-level description) to compare the power of TM variants
- Understand the specific simulation arguments we've seen: two-way infinite TM by basic TM, multi-tape TM by basic TM, nondeterministic TM by basic TM

Decidability (4.1)

- Understand how to use a TM to simulate another machine (DFA, another TM)
- Know the specific decidable languages from language theory that we've discussed, and how to decide them: A_{DFA}, E_{DFA}, EQ_{DFA}, etc.
- Know how to use a reduction to one of these languages to show that a new language is decidable

E.g. knowing E_{DFA} decidable, gave a reduction from EQ_{DFA} to E_{DFA}, concluding EQ_{DFA} is also decidable.

Undecidability (4.2)

- Know the definitions of countable and uncountable sets and how to prove countability and uncountability
- Understand how diagonalization is used to prove the existence of an explicit undecidable language UD
- Know that a language is decidable iff it is recognizable and its complement is recognizable, and understand the proof

$UD = \{ \langle M \rangle \mid M$ is a TM s.t. M does not accept input $\langle M \rangle \}$

Reducibility (5.1)

- Understand how to use a reduction (contradiction argument) to prove that a language is undecidable
- Know the reductions showing that $HALT_{TM}$, E_{TM}, $REGULAR_{TM}$, EQ_{TM} are undecidable

  Sipser, unannotated slides 15

- You are not responsible for understanding the computation history method.

- I will not ask you to prove something is undecidable by reduction on the in-class test
- You may get a conceptual question about this
- There is an undecidability by reduction question on take-home

True or False

- It's all about the justification!
- The logic of the argument has to be clear
- Restating the question is not justification; we're looking for additional insight
- False statements should be justified by a specific counterexample whenever possible.

True. If A is finite, it is regular, as shown in class. The regular languages are closed under intersection, so $A \cap B$ is also regular.

Simulation arguments, constructing deciders

How to initialize simulating machine

Give a simulation argument, using an implementation-level description, to show that TMs with reset recognize the class of Turing-recognizable languages. *Hint:* You may want to simulate using a two-tape TM. (12 points)

We simulate a TM with reset using a two-tape TM as follows. The first tape of the new machine is read-only and used the store the input. We initialize the second tape by marking the left end of the tape with a special symbol $, copying the input, and then marking the right end of the input with another special symbol #. (These special symbols are in place to allow us to know how much of the second tape is actually in use during simulation).

To simulate one ordinary step (i.e., read, write, and move) of the TM with reset, we simulate its action on the second tape of our new machine, treating the cell containing $ as the left end of the tape and moving the # symbol to the right by one cell if we ever try to overwrite it.

To simulate a reset step, we scan the second tape of the new machine between the $ symbol and the # to erase its contents and re-initialize the second tape by copying the input from the first tape, again demarcated by $ and #.

How to simulate one step of computation

- Full credit for a clear and correct description of the new machine
- Can still be a good idea to provide an explanation (partial credit, clarifying ambiguity)

Countability proofs

A *DNA strand* is a finite string over the alphabet $\{A, C, G, T\}$. Show that the set of all DNA strands is countable. (8 points)

We may list the elements of this set in stages $i = 0, 1, 2, \ldots$ as follows. In stage 0, we list the empty string, the only string of length 0. In stage 1, we list all strings of length 1, etc. In general, in stage i, we list all 4^i strings of length i. We obtain a correspondence f from the set of natural numbers into this set of strings by taking $f(n)$ to be the nth string in this list.

- Describe how to list all the elements in your set, usually in a succession of finite "stages"
- Describe how this listing process gives you a bijection from the natural numbers

Uncountability proofs

Let $\mathcal{F} = \{f : \mathbb{Z} \to \mathbb{Z}\}$ be the set of all functions taking as input an integer and outputting an integer. Show that $\mathcal{F}$ is uncountable. (10 points)

Set up contradiction

Suppose for the sake of contradiction that $\mathcal{F}$ were countable, and let $B : \mathbb{N} \to \mathcal{F}$ be a bijection. For each $i \in \mathbb{N}$, let $f_i = B(i)$. Define the function $g \in \mathcal{F}$ as follows. For every $i = 1, 2, \ldots$ let $g(i) = f_i(i) + 1$. For every $i = 0, -1, -2, \ldots,$ let $g(i) = 0$. This definition of the function g ensures that $g(i) \neq f_i(i)$ for every $i \in \mathbb{N}$. Hence, $g \neq f_i = B(i)$ for any i, which contradicts the onto property of the map B.

Construct element flipping the diagonal

Explain why construction works

- The 2-D table is useful for helping you think about diagonalization, but does not need to appear in the proof
- The essential part of the proof is the construction of the "inverted diagonal" element, and the proof that it works

Undecidability proofs

Show that the language Y is undecidable. (10 points)

We show that Y is undecidable by giving a reduction from A_{TM}. Suppose for the sake of contradiction that we had a decider R for Y. We construct a decider for A_{TM} as follows:

Set up contradiction argument

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

1. Use M and w to construct the following TM M':

   M' = "On input x:

   1. If x has even length, *accept*
   2. Run M on w
   3. If M accepts, *accept*. If M rejects, *reject*."

2. Run R on input $\langle M' \rangle$
3. If R accepts, *reject*. If R rejects, *accept*."

Describe TM deciding language you are reducing <u>from</u>

If M accepts w, then the machine M' accepts all strings. On the other hand, if M does not accept w, then M' only accepts strings of even length. Explain why TM works

Hence this machine decides A_{TM} which is a contradiction, since A_{TM} is undecidable. Hence Y must be undecidable as well. Conclude by contradiction

Practice Problems

Closure properties

Decidable languages are closed under:

- union
- intersection
- complement
- concatenation
- star
- reverse

Recognizable languages closed under: All of the above except complement.

Counterexample: A_{TM} is recognizable

$\overline{A_{TM}}$ is not recognizable

Can use closure of decidable languages to prove undecidability.

- A is decidable
- $B = A \cup C$ is undecidable
- what can we conclude about C?

<u>Claim:</u> C is undecidable

<u>Proof.</u> If C were decidable,

$B = A \cup C$ would decidable (closure under union)

$\times$

$A = \{\langle M, w\rangle \mid$ TM M accepts w within 332 steps$\}$

$B = A_{TM} = \{\langle M, w\rangle \mid M$ accepts $w\}$

A is decidable:

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

1. Run M on w for 332 steps
2. If accepts, accept, o.w. reject"

B is undecidable

$C = \{\langle M, w\rangle \mid$ TM M accepts w in > 332 steps$\}$

$B = A \cup C$, A decidable + B undecidable $\Rightarrow$ C undecidable

Decidability and Recognizability

Let

$$A = \{\langle R, S\rangle \mid R, S \text{ are regexes such that } L(R) \subseteq L(S)\}$$

Show that A is decidable

Prove that $\overline{E_{\mathrm{TM}}}$ is recognizable

Prove that if A and B are decidable, then so is $A \setminus B$

Countable and Uncountable Sets

Show that the set of all valid (i.e., compiling without errors) C++ programs is countable

A Celebrity Twitter Feed is an infinite sequence of ASCII strings, each with at most 140 characters. Show that the set of Celebrity Twitter Feeds is uncountable. = CTF.

Ex: ("covfefe", "fake news", ...) ∈ CTF
("covfefe": ≤ 140 chars)

Assume for contradiction CTF is countable w/ bijection $f: \mathbb{N} \to CTF$

n				
1	w_1^1,	w_1^2,	w_1^3,	w_1^4 ...
2	w_2^1,	w_2^2,	...	
3	w_3^1,	w_3^2,	w_3^3 ...	
4				
⋮				

Goal: Make $b^n \neq w_n^n$ for every n as follows:

If $|w_n^n| < 140$, let $b^n = w_n^n \circ a$

If $|w_n^n| = 140$, let $b^n = a$

Note: $b \in CTF$ because, by construction, b^n is a ≤140 char string $\forall n$

Want $b = (b^1, b^2, \ldots) \in CTF$, but $b \neq f(n)$ for all n.

$\Rightarrow$ f not onto, contradicting assumption

Undecidability and Unrecognizability

Prove or disprove: If A and B are recognizable, then so is $A \setminus B$

Prove that the language $ALL_{\text{TM}} = \{\langle M \rangle | M \text{ is a TM and } L(M) = \Sigma^*\}$ is undecidable

Idea: Reduce from $A_{TM} = \{\langle M, w \rangle \mid \text{TM } M \text{ accepts } w\}$ to ALL_{TM}

Proof: Assume for contradiction ALL_{TM} decided by TM R

Goal: Construct decider S for A_{TM} as follows:

$S =$ " On input $\langle M, w \rangle$:

1. Construct (using M, w) TM N
2. Run R on input $\langle N \rangle$
3. If R accepts, accept, if R rejects, reject"

How to construct N:

$\langle M, w \rangle \in A_{TM} \iff S$ accepts $\langle M, w \rangle \iff R$ accepts $\langle N \rangle \iff \langle N \rangle \in ALL_{TM}$

$\iff L(N) = \Sigma^*$

Want: $L(N) = \begin{cases} \Sigma^* & \text{if } M \text{ accepts } w \\ \text{anything else} & \text{if } M \text{ does not accept } w. \end{cases}$

Want: $L(N) = \begin{cases} \Sigma^* & \text{if } M \text{ accepts } w \\ \emptyset & \text{if } M \text{ does not accept } w \end{cases}$

N = " On input x :

1. Run M on w. If accepts, <u>accept</u>. If rejects, <u>reject</u>. "

Started w/ TM R deciding ALL_{TM}

S = " on input ⟨M, w⟩: // input to A_{TM}

1. Construct N as above.
2. Run R on input ⟨N⟩
3. If accepts, <u>accept</u>, o.w. <u>reject</u>. "

Conclusion: A_{TM} is undecidable ✳ So ALL_{TM} is undecidable

Claim: S decides A_{TM}

1) If $\langle M,w \rangle \in A_{TM}$, $L(N) = \Sigma^*$ => R accepts => S accepts
2) If $\langle M,w \rangle \notin A_{TM}$, $L(N) = \emptyset$ => R rejects => S rejects.