BU CS 332 – Theory of Computation

Lecture 18:

- Test 2 Review

Mark Bun

April 2, 2025

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 (both variants of) 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

- You are not responsible for understanding any of the material involving context-free languages in this chapter

Countable and Uncountable Sets (4.2)

- Know the definitions of countable and uncountable sets

- Know how to prove that a set is countable, e.g., by describing how to list its elements using a sequence of finite stages

- Know how to prove a set is uncountable by diagonalization

Undecidability (4.2)

$\{\langle M\rangle \mid$ TM M s.t. M does not accept on input $\langle M\rangle\}$

- 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

Reducibility (5.1)

- Understand how to use a reduction (contradiction argument) to prove that a language is undecidable

- Be familiar with the reductions showing that $HALT_{TM}$, $E_{TM}, REGULAR_{TM}, EQ_{TM}$ are undecidable

- You are not responsible for understanding the computation history method described in this chapter

Mapping Reducibility (5.3)

$\exists$ TM M $\forall w \in \Sigma^{*}$
M <u>halts</u> w/ $f(w)$ on its
tape

- Understand the definition of a <u>computable function</u>

- Understand the definition of a mapping reduction

- Know how to use mapping reductions to prove decidability, undecidability, recognizability, and unrecognizability

$A \leq_m B$
B decidable $\Rightarrow$ A decidable
A undecidable $\Rightarrow$ B undecidable

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/recognizers

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)

How to initialize simulating -M

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.

"distinct"

- 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)

It suffices to show that there is no surjection $B: \mathbb{N} \to \mathcal{F}$. Let $B: \mathbb{N} \to \mathcal{F}$ be an arbitrary function and for each $i \in \mathbb{N}$ let $f_i = B(i)$. Define the function $g \in \mathcal{F}$ as follows. For every $i = 1, \dots$, let $g(i) = f_i(i) + 1$. For every $i = 0, -1, -2, \dots$, 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 every i, so B is not a surjection.

- 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) *Reduction from A_{TM} to Y*

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:

"On input $\langle M, w \rangle$: *String representing an input to M* *does $\langle M, w \rangle$ is an instance of A_{TM}* *decider for A_{TM}*

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

   $M' = $ "On input x: *Represents an arbitrary input to M'*
   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 $\langle M' \rangle$ *Use decider for Y as a subroutine*

3. If R accepts, *reject*. If R rejects, *accept*." *Explanation of correctness*

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.

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

Practice Problems

CS332 - Theory of Computation

CS332 - Theory of Computation

Turing Machines and TM Variants

A TM with reset is the same as a basic single-tape TM, except at any point it its computation, it may "reset" resulting in both of the following:
 1) The entire contents of the tape are erased and replaced with the machine's original input, and
 2) The head is returned to the left end of the tape.
Show that TMs with reset recognize the class of Turing-recognizable languages

Describe a *nondeterministic* Turing machine recognizing the language $L = \{k\#s_1\#s_2\# \ldots \#s_n \mid$
there exists a string of length k containing every s_i as a substring$\}$

CS332 - Theory of Computation

Decidability and Recognizability

Let

$A = \{\langle R, S \rangle \mid R, S$ are regexes such that R generates some string not generated by $S \}$

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 function $f : \mathbb{N} \to \mathbb{N}$ is decreasing if $f(i) > f(i+1)$ for every $i \in \mathbb{N}$. Show that the set of all decreasing functions 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.

Undecidability and Unrecognizability

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

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

Σ' is whatever the input alphabet of M is

Proof by mapping reductions:

TM B:
On input x: If $x = 0$, reject
else accept.
$L(B) = \Sigma'^* \setminus \{0\}$

Is $\langle B \rangle \in ALL_{\text{TM}}$?
No

Idea: Give a mapping reduction $A_{\text{TM}} \leq_m ALL_{\text{TM}}$
Suffices because know A_{TM} is undecidable and $[A \leq_m B + A$ undecidable $\Rightarrow B$ undecidable$]$

I.e. want to give a computable function $f : \Sigma_1^* \rightarrow \Sigma_1'^*$ s.t.

- $\forall \langle M, w \rangle \in A_{\text{TM}}$ {ie., TM M accepts input w}
$\langle N \rangle = \underline{f(\langle M, w \rangle) \in ALL_{\text{TM}}}$, i.e. N is a TM s.t. $L(N) = \Sigma_1^*$

- $\forall \langle M, w \rangle \notin A_{\text{TM}}$ {ie. TM M does not accept w} $\leftarrow$
$\langle N \rangle = \underline{f(\langle M, w \rangle) \notin ALL_{\text{TM}}}$ i.e. N is a TM s.t. $L(N) \neq \Sigma_1^*$

Note: $ALL_{\text{TM}} = \{\langle N \rangle | \forall$ strings $x \in \Sigma_1^*,$ TM N accepts $x\}$

The following TM computes f:

On input $\langle M, \omega \rangle$:   // instance of A_{TM} problem

    1. Construct TM N:

        "On input x: // Ignore x

          1. Run M on input ω.

          2. If accepts, accept. If rejects, reject."

    2. Output $\langle N \rangle$ // instance of ALL_{TM} problem

Correctness:

- $\forall \langle M, \omega \rangle \in A_{TM}$, $\langle N \rangle \in ALL_{TM}$

    $\Rightarrow$ M accepts ω $\Rightarrow$ N accepts all strings x $\Rightarrow$ $L(N) = \Sigma^{*} \Rightarrow \langle N \rangle \in ALL_{TM}$

- $\forall \langle M, \omega \rangle \notin A_{TM}$, $\langle N \rangle \notin ALL_{TM}$

    $\Rightarrow$ M does not accept ω $\Rightarrow$ N fails to accept every x $\Rightarrow$ $L(N) = \emptyset \Rightarrow \langle N \rangle \notin ALL_{TM}$

Assume for the sake of contradiction $\exists$ a TM V deciding ALL_{TM}

Construct TM H deciding A_{TM} as follows:
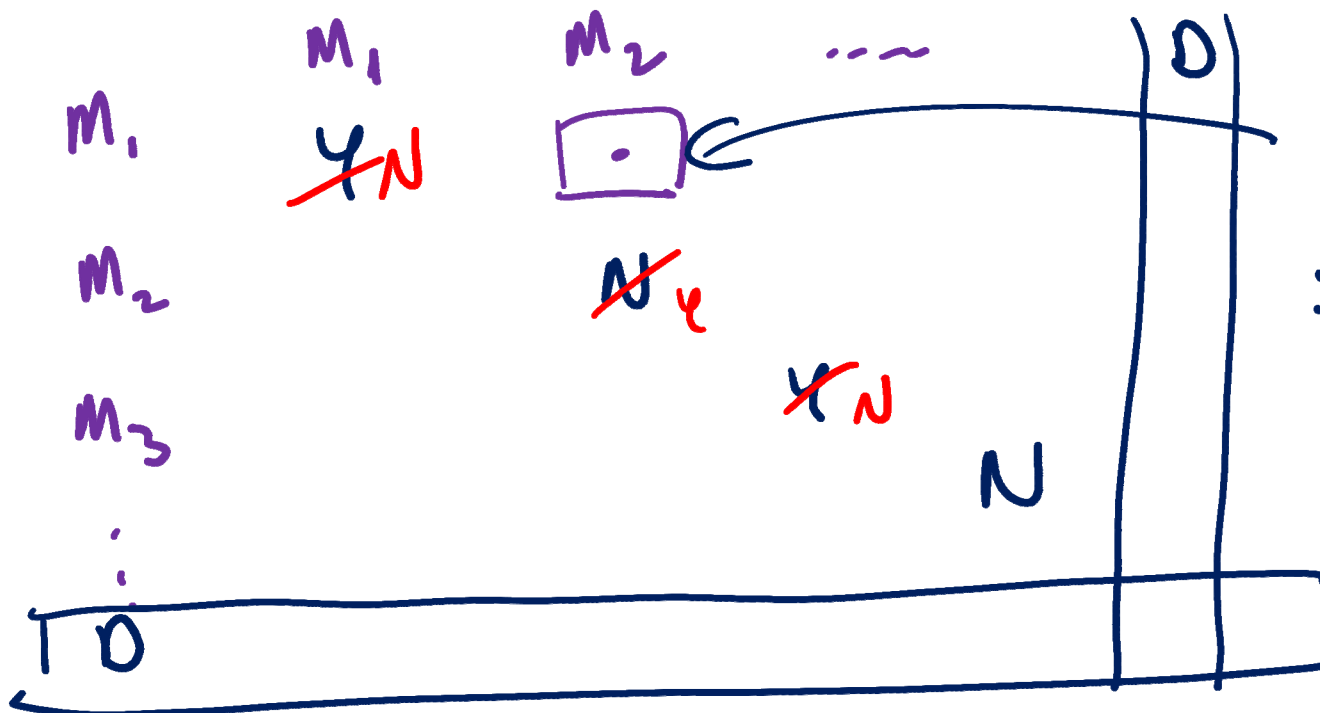
On input $\langle M, w \rangle$:

1. Construct N:
   "on input x:
      Run M on w, return its output"

2. Run V on input $\langle N \rangle$

3. Return output of V.

This decides A_{TM} which contradicts A_{TM} being undecidable,

hence ALL_{TM} undecidable.

M_1 M_2 $\cdots\sim$ D

M_1 $\overset{\text{Y}}{\cancel{}}N$ $\boxed{\cdot}$

What happens if I run M_1 on input $\langle M_2 \rangle$. If accepts, write "Y". Else, write "N"

M_2 $\cancel{N}_Y$

M_3 $\overset{\text{Y}}{\cancel{}}N$

N

$\vdots$

D

$UD = \{ \langle M_i \rangle \mid$ "Y" appears as (invoked) answer in entry (M_i, M_i) of the table $\}$

$= \{ \langle M \rangle \mid M$ does not accept input $\langle M \rangle \}$

$\forall$ TMs D:

- D accepts $\langle D \rangle \implies \langle D \rangle \notin UD$
- D rejects $\langle D \rangle \implies \langle D \rangle \in UD$

$\implies D$ does not decide UD.