# BU CS 332 – Theory of Computation

## Lecture 16:

- Test 2 Review

Logistics

- Test 2 on Monday 4/1
- No office hours Wed afternoon, Fri morning

(I'll stick around after class + try to schedule extra OH this Friday)

- HW 7 now due next Thursday 4/4

Mark Bun

March 27, 2024

# 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

To show TM variant A is "at least as powerful" as TM variant B, wts that anything TM B can do, A can also do.

To show TM variant A is "no more powerful" than B, wts that anything A can do, B can also do.

# 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

Ex: we showed that $EQ_{DFA}$ is decidable by reducing to $E_{DFA}$

i.e., given a decider for $E_{DFA}$, we constructed a decider for $EQ_{DFA}$

# 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 "Dovetailing trick"
- Understand the diagonalization-based proofs of uncountability we saw in class
  - Uncountability of the real numbers
  - The power set of a set is always "larger" than the set itself.
- I will not ask you to prove that some set is uncountable on our in-class test. But you may get a conceptual question about how diagonalization works and what kinds of statements it can be used to show.

$A^*$

$A \subseteq B$ ⟸ uncountable → uncountable

# Undecidability (4.2)

L decidable ⇔ L recognizable and $\overline{L}$ recognizable

L undecidable ⇔ L unrecognizable or $\overline{L}$ unrecognizable

- 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

conclude: If L undecidable and $\overline{L}$ recognizable, then L is unrecognizable

$UD = \{\langle M \rangle \mid$ M is a Turing machine s.t. M does not accept when run on $\langle M \rangle\}$

Proof that UD is undecidable:

Assume for contradiction that D decides UD. Then either:

1) $D(\langle D \rangle)$ accepts $\Rightarrow \langle D \rangle \in UD$ [definition of D being a decider for UD] $\Leftarrow \langle D \rangle \notin UD$ [definition of UD] ↯

2) $D(\langle D \rangle)$ does not accept $\Rightarrow \langle D \rangle \notin UD$ [def. of decider] $\Leftarrow \langle D \rangle \in UD$ [definition of UD] ↯

$\Rightarrow$ D can't exist.

# 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

- I will not ask you to prove that something is undecidable by reduction on our in-class test. But you may get a conceptual question about how reductions work / what they can be used to show.

# 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.

Q: If A is finite and B is regular, then $A \cap B$ is regular.

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)

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 #.

shows that "TMs with reset" are no more powerful that two-tape TMs

Describe how to initialize simulating Machine

Describe 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 $n$th 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)

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 $B$.

conclude uncountability by contradiction

explain why it is not hit

construct some element of $\mathcal{F}$ that is not hit by $B$

- 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

Reduce from some known undecidable language, e.g. $A_{TM}$

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

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

"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*."

Using an alleged decider for Y to build a decider for $A_{TM}$

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.

Explanation of correctness

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

Conclusion by contradiction

# Practice Problems

# 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 $L = \{k\#s_1\#s_2\# \ldots \#s_n \mid k, n \in \{0,1,\#\}^*$ there exists a string of length $k$ containing every $s_i$ as a substring$\}$

(written in binary)

Example: $\underbrace{101}_{5}\#01\#10\#1000 \in L$ because 10001 is a string of length 5 s.t. 01, 10, 1000 all appear as substrings.

High-level description of NTM: On input $k\#s_1\#\ldots\#s_n$

1) Parse $k$ from input. [Nondeterministically guess a string $y$ of length $k$.] For every string $y$ of length $k$:
Deterministic version of this alg.

2) For each $i = 1, \ldots, n$:
Scan input and $y$ to check that $s_i$ appears as a substring of $y$

If all checks of $s_i$'s pass, accept. If any check fails, reject.

Correctness: If $w = k\#s_1\#\ldots\#s_n \in L$, $\exists$ $y$ s.t. $|y| = k$ and $s_1, \ldots, s_n$ are substrings of $y$
$\Rightarrow \exists$ a branch of computation leading to accept.

If "$w \notin L \Rightarrow$ no branch leads to accept

# 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 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.

E.g. $T =$ "How can mirrors be real if our eyes aren't", "Yay decidability!", ....
($\leq 140$; $s_1^1$; $s_1^2$)

For intuition: Assume this is countable, with bijection $f : \mathbb{N} \to \{$ celeb. Twitter feeds $\}$

$f(1) = T_1 = [s_1^1]\ s_1^2\ s_1^3 \ldots$ ($\leq 140$ char, $\leq 140$ char)

$f(2) = T_2 = s_2^1\ [s_2^2]\ s_2^3 \ldots$

$f(3) = T_3 = s_3^1\ s_3^2\ [s_3^3] \ldots$

$\vdots$

Goal: Construct some feed $T$ which is
- a valid celeb. Twitter feed
- nowhere in this enumeration

Define $T = (t^1, t^2, \ldots)$ where each $t^i$ is a $\leq 140$ char string as follows

Each $t^i = \begin{cases} \text{"a"} & \text{if } s_i^i \text{ is empty string} \\ s_i^i & \text{except with last character changed to something else.} \end{cases}$

$\Rightarrow$ For every $i$, $T$ differs from $T_i$ in its $i$'th Tweet $\Rightarrow$ For every $i$, $T$ differs from $T_i$
$\Rightarrow T$ cannot be $f(i)$ for any $i$
$\Rightarrow f$ is not a bijection $\lightning$.

# 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