

BU CS 332 – Theory of Computation

Lecture 18:

- Test 2 Review

MASSACHUSETTS ACTUALLY HAS 12 SEASONS

Winter
Fool's Spring
Second Winter
Spring of Deception
Third Winter ← you are here
The Pollening
Actual Spring
Summer
Hell's Front Porch
False Fall
Second Summer
Actual Fall

Alexander Poremba & Mark Bun

April 7, 2026

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)

- 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)

- Understand the definition of a computable function
- Understand the definition of a mapping reduction
- Know how to use mapping reductions to prove decidability, undecidability, recognizability, and unrecognizability

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 TM

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 to 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 #.

Implementation-level description of 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 $i = 0, 1, 2, \dots$ 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 $\mathcal{F} = \{f : \mathbb{Z} \rightarrow \mathbb{Z}\}$ 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 $B: \mathbb{N} \rightarrow \mathcal{F}$. Let $B: \mathbb{N} \rightarrow \mathcal{F}$ 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)

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 A_{TM} in terms of alleged TM deciding Y

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_{TM} which is a contradiction, since A_{TM} is undecidable. Hence Y must be undecidable as well.

Conclusion to contradiction argument

Practice Problems

Turing Machines and TM Variants

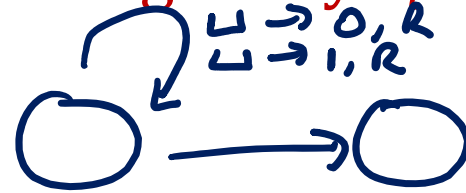
A TM with reset is the same as a basic single-tape TM, except at any point in its computation, it may “reset” resulting in both of the following:

- 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 $L = \{k\#s_1\#s_2\#\dots\#s_n \mid \text{there exists a string of length } k \text{ containing every } s_i \text{ as a substring}\}$

NM Turing deciding L :



On input x :

1. Scan x to check it has the form $k\#s_1\#\dots\#s_n$.
If not, reject.
2. Nondeterministically guess a string z of length k .
3. For each s_1, \dots, s_n :
Scan z to check that s_i is a substring of z .
4. If all checks pass, accept. Else reject.

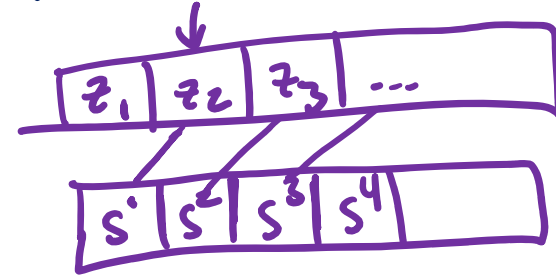
Correctness: If $x \in L$: $\exists z$ of length k s.t. s_1, \dots, s_n all substrings of z
 \Rightarrow branch of computation where this z was guessed leads NM Turing to accept

If $x \notin L$: There is no choice of z s.t. all s_1, \dots, s_n appear as substrings of z
 \Rightarrow All 2^k branches of computation lead to rejection \Rightarrow NM Turing rejects overall.

Implementation-level description on Multitape TM

On input x :

1. Copy length n to tape 2 in unary , i.e. 1^n
 2. Nondeterministically guess string $z \in \{0,1\}^n$ by snapping down its length reaches n (on tape 3) appearing on tape 2
 3. Scan next, copying next s_i to tape 4
 4. Scan tape 3 and tape 4 in parallel to test if s_i matches corresponding content in z .
- If not a match, advance one cell right on tape 3, go back to step 4
5. If no match found in 3, reject.
Else, go back to step 3
 6. If reach end of input, accept.



Show $AS_M = \{ \langle M, w \rangle \mid \exists z \text{ s.t. TM } M \text{ accepts input } wz \}$
is Turing-recognizable

$\langle M, w \rangle \in AS_M$ if at least one of the following happens

- M accepts w
 - M accepts $w0$
 - M accepts $w1$
 - M accepts $w00 \dots$
-

First attempt: Try running M on every possible string wz , i.e.

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

1. For each string $z \in \{0,1\}^*$:

Run M on input wz

If accepts, accept. Else, continue

Problem:

M might loop forever on some wz , which will prevent it from being run on any subsequent wz'

Detailing to the issue!

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

1. For each $i = 1, 2, 3, \dots$

For each z st. $|z| \leq i$:

Run M on wz for i steps.

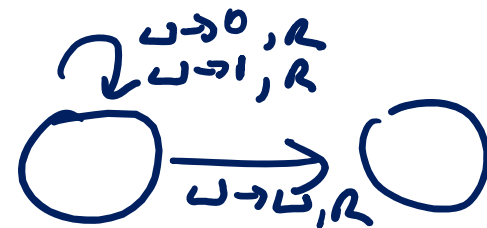
If accepts, accept.

Alternative solution on an NTM:

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

1. Nondeterministically guess $z \in \{0, 1\}^*$ ← legitimate for an NTM recognizer,

2. Run M on wz . If accepts, accept. If rejects, reject. but not for a decider, because it's possible to enter an infinite loop.



Decidability and Recognizability

Let

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

Show that A is decidable

Prove that $\overline{E_{TM}}$ 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 $f : \mathbb{N} \rightarrow \mathbb{N}$ 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 $ALL_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \Sigma^*\}$ is undecidable