BU CS 332 – Theory of Computation

Lecture 17:

- Midterm II review

Reading for midterm:

Sipser Ch, 3, 4, 5

Ran Canetti

November 5, 2020

Format of the Exam Same as Midterm I:

- In Class, 70 minutes
- Via Gradescope
- Questions will test:
  - Knowledge of facts learned
  - Understanding of main concepts learned
  - Ability to use the tools learned
- Not meant to be hard!

Study Tips

- Review problems from HW 5-7, discussion sections 6-9, solved exercises/problems in Sipser, suggested exercises on the homework, and the practice midterm.
  - We will ask you a question from the homework exercises (or a close variant), so make sure you understand these
- While you are not required to prepare a cheat-sheet, this is a great way to study!

Study Tips

- Try to solve the practice midterm in same setting as the exam (70 minutes, one sitting). The exam will have a similar format.
- Make use of office hours
- Make use of Nathan the tutor
- If you need more practice, there are lots of problems in the book. We're happy to talk about any of these problems in office hours!

For the exam itself

- You may cite without proof any result…
  - Stated in lecture or discussion
  - Stated and proved in the main body of the text (Ch. 3-5)
  - These include worked-out solutions in the main text.
- Not included above: homework problems, (solved) exercise/problems in the book.
- Showing your work / explaining your answers will help us give you partial credit..

Midterm II 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 implementation-level descriptions

TM Variants (3.2)

- Understand the following TM variants: Multi-tape TMs, Nondeterministic TMs, Enumerators
- Know how to give a simulation argument (implementation-level description) to compare the power of TM variants
- Understand the specific simulation arguments we've seen: multi-tape TM by basic TM, nondeterministic TM by basic TM, enumerator by basic TM and basic TM by enumerator.

Universality of TMs (3.3, 4.2)

- Understand how to use a TM to simulate another machine (DFA, another TM)
  - The actual construction (program it!)
  - The concept and implications

- Understand the Church-Turing Thesis

Decidability (4.1)

- Know the specific decidable languages from language theory that we've discussed, and how to decide them: $A_{DFA}, E_{DFA}, EQ_{DFA}, A_{CFG}, E_{CFG}$, etc.
- Know how to use a reduction to one of these languages to show that a new language is 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 undecidability of A_{TM}
- Know that a language is decidable iff it is recognizable and co-recognizable, and understand the proof

Reducibility (5.1)

- Understand how to use a reduction (contradiction argument) to prove that a language is undecidable
- Know the reductions showing that $HALT_{TM}$, $E_{TM}, REGULAR_{TM}, CFL_{TM}, EQ_{TM}$ are undecidable
- You are not responsible for understanding the computation history method. However, you should know that the languages POST and EQ_{CFG} are undecidable, and reducing from them might be useful.

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

Tips for the Exam

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 some additional insight

All regular languages are Turing recognizable.

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 some additional insight

T All regular languages are Turing recognizable.

We showed in class that that non-regular languages are not always decidable.
We also showed that if both a language and its complement are recognizable then both are decidable, and regular languages are closed under complement.
It follows that regular languages are Turing recognizable.

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 some additional insight

T All regular languages are Turing-recognizable.

We showed in class that all context-free languages are decidable. Since all regular languages are context free, and all decidable languages are recognizable, it follows that all regular languages are also recognizable.

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:

"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*."

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.

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.

- The 2-D table is useful for thinking about diagonalization, but is not essential to the argument
- The essential part of the proof is the construction of the "inverted diagonal" element, and the proof that it works

Practice Problems

Decidability and Recognizability

Let $A = \{\langle D \rangle \mid D \text{ is a DFA that does not accept any string containing an odd number of } 1's\}$

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., compile 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.

Undecidability and Unrecognizability

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

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

Give a nonregular language A such that $A \leq_m L(0^*1^*)$ or prove that none exists

Give an undecidable language A such that $A \leq_m L(0^*1^*)$ or prove that none exists

Give an undecidable language A such that $L(0^*1^*) \leq_m A$ or prove that none exists