

BU CS 332 – Theory of Computation

Lecture 15:

- Review mid-semester feedback
- Reductions

Reading:

Sipser Ch 5.1

Mark Bun

March 15, 2021

What helps you learn best?

- Lectures in general (13)
- In-class examples / walkthroughs (11)
- Interaction in lecture, polls (7)
- Discussion sections in general (7)
- Gradescope check-ins (5)
- Homework – useful, appropriate length/difficulty (5)
- Automata Tutor, TM simulator (4)
- Office hours (4)
- Course organization, perspective (3)
- Annotated slides (3)
- Piazza use (3)
- Homework feedback
- Reading

What hinders your learning?

- Remote format, COVID / Zoom fatigue (7)
- Course difficulty, recent increase in difficulty (4)
- Proofs, proof assignments on homework (2)
- Too theoretical / knowledge of concepts but not how to use them (2)
- Homework too long, too difficult (2)
- Automata Tutor / Morphett
- Turing machines
- Starting homework late
- Vague answers in office hours
- Transferring lecture knowledge to homework
- Delay on homework feedback
- Sipser book
- Difficulty finding collaborators
- Gradescope check-ins *more socials*
- Instructor mistakes *Piazza*
- Weekly (vs. less frequent) assignments *matchmaking*
- Instructor handwriting
- Course pace too fast
- Can't turn in late work

Suggestions for course improvement

- More office hours (3)
 - More examples (3)
 - Recommendations for what's expected on HW solutions (3)
 - More polls, interaction (2)
 - Better / more visible handwriting, or type on slides (2)
 - Faster turnaround on grades (2)
 - +12 hours on HW submissions (2)
 - Supplementary readings / videos (2)
 - Shorter breakouts in discussion
 - Releasing homework earlier
 - Guidance on how to prove things
 - Homeworks build up from easier to harder questions
 - Homework solutions
 - Tutoring sessions
 - Old exam questions during discussion
 - More ungraded practice
 - Less difficult homework
 - Slower lectures
 - Free A's
- HW 6 due Friday

Clarity of expectations

- Seems mostly clear
- Reminder of resources to take advantage of:
 - Sipser textbook
 - Lectures (slides, recordings, Gradescope check-ins)
 - Discussions (in-class meetings, solution recording, posted slides)
 - Homework feedback, posted solutions
 - Office hours
 - Piazza
- See Lecture 1, Slides 13-17 for more advice

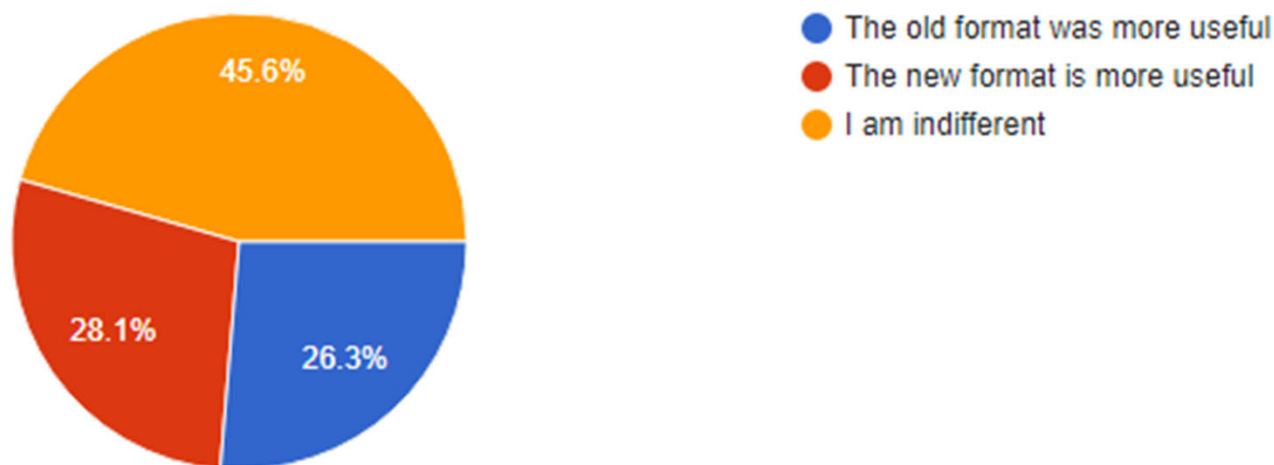
Suggestions for self-improvement

- Keep up with readings (24)

“I believe reading the textbook is more helpful than students realize. I have been reading the textbook inconsistently and I find the weeks that I do the reading, I can better understand the lecture material.”

- Time management (10)
- Review lecture / discussion materials (8)
- Attend more office hours (5)
- Participate in class more actively (5)
- Attend more discussions (2)
- Do example problems in Sipser (2)
- Find collaborators
- Remember to do Gradescope check-ins
- Participate on Piazza

Discussion format / feedback



- Nadya's awesome (9)
- Breakout rooms can be awkward / not useful (3)
- New format is more engaging, allow for solving more problems (2)
- Discussion problems too easy

Proposed Course Modifications

- Poll for more office hours, tutoring
- Discussions
 - Keeping new format (class time for breakout rooms, solution video after)
 - Nadya will kick start discussion in quieter groups
- Homework more approachable and useful
 - More explicit guidance on components of a complete solution (See also lecture slides where I try to do this)
 - Gradient from easier (mechanical) to harder (creative) questions

Other questions / concerns

- Is the final cumulative?

Yes, with an emphasis on last third of material

- Is there a curve?

Can expect grade increases for scores especially in the 50-75% range

- Specific concerns about test grades (especially relative to HW)

Come talk to us about this. Office hours can be an awkward time, so schedule an appointment

Undecidability and Reductions

Undecidability / Unrecognizability

Definition: A language L is undecidable if there is no TM deciding L

Definition: A language L is unrecognizable if there is no TM recognizing L

Last time: Two explicit undecidable languages

$UD = \{\langle M \rangle \mid M \text{ is a TM that does not accept on input } \langle M \rangle\}$

- Shown directly by diagonalization

$A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM that accepts input } w\}$

- “Reduction” from the undecidability of UD

- Recognizable via Universal TM

Scientists vs. Engineers

A computer scientist and an engineer are stranded on a desert island. They find two palm trees with one coconut on each. The engineer climbs a tree, picks a coconut and eats.



The computer scientist climbs the second tree, picks a coconut, climbs down, climbs up the first tree and places it there, declaring success.

“Now we’ve reduced the problem to one we’ve already solved.”
(Please laugh)

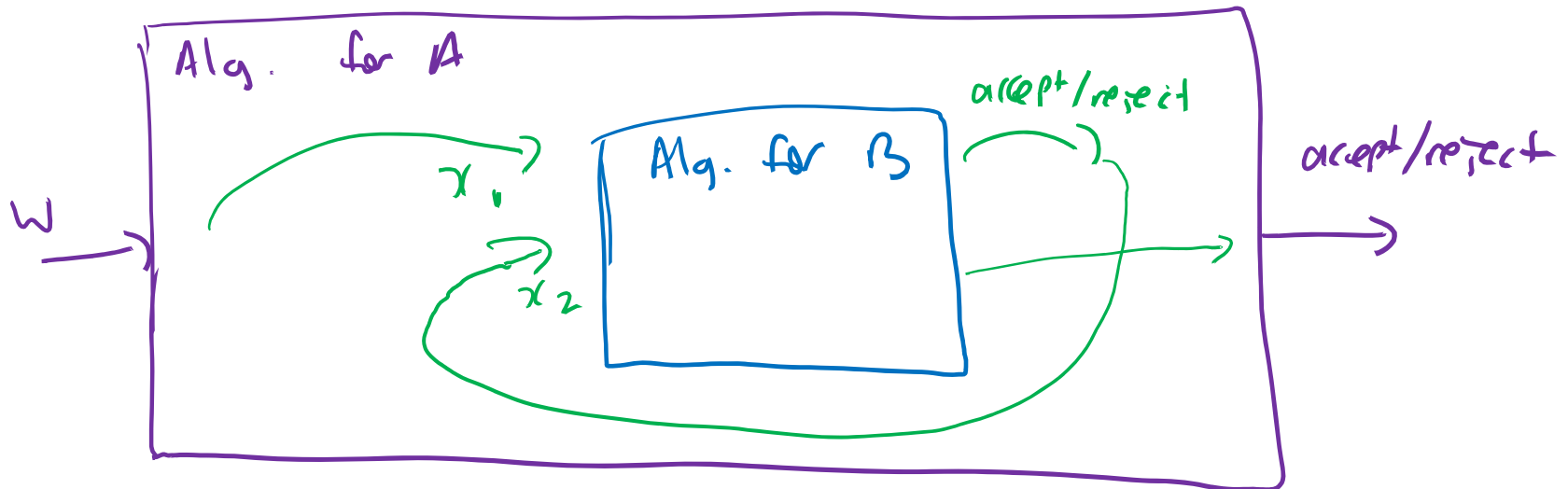
Reductions

A **reduction** from problem A to problem B is an algorithm for problem A which uses an algorithm for problem B as a subroutine

Eating coconut from tree 2

Eating coconut from tree 1

If such a reduction exists, we say “ A reduces to B ”



Reductions

A **reduction** from problem A to problem B is an algorithm for problem A which uses an algorithm for problem B as a subroutine

If such a reduction exists, we say “ A reduces to B ”

If A reduces to B , and B is decidable, what can we say about A ?

a) A is decidable

b) A is undecidable

c) A might be either decidable or undecidable



Two uses of reductions

Positive uses: If A reduces to B and B is decidable, then A is also decidable

$EQ_{\text{DFA}} = \{\langle D_1, D_2 \rangle \mid D_1, D_2 \text{ are DFAs and } L(D_1) = L(D_2)\}$

Theorem: EQ_{DFA} is decidable

Proof: The following TM decides EQ_{DFA}

A reduction from EQ_{DFA} to E_{DFA}

On input $\langle D_1, D_2 \rangle$, where $\langle D_1, D_2 \rangle$ are DFAs:

1. Construct a DFA D that recognizes the symmetric difference $L(D_1) \Delta L(D_2)$
2. Run the decider for E_{DFA} on $\langle D \rangle$ and return its output

Two uses of reductions

Negative uses: If A reduces to B and A is undecidable, then B is also undecidable

$A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM that accepts input } w\}$

Suppose H decides A_{TM}
Encoding / to String of M

Reduction from
UD to A_{TM}

Consider the following TM D .

On input $\langle M \rangle$ where M is a TM:

1. Run H on input $\langle M, \langle M \rangle \rangle$
2. If H accepts, **reject**. If H rejects, **accept**.

Claim: D decides

$UD = \{\langle M \rangle \mid M \text{ is a TM that does not accept input } \langle M \rangle\}$

Two uses of reductions

Negative uses: If A reduces to B and A is undecidable, then B is also undecidable

KNOW: A undecidable

WTS: B undecidable

Template for undecidability proof by reduction:

1. Suppose to the contrary that B is decidable
2. Using a decider for B as a subroutine, construct an algorithm deciding A
3. But A is undecidable. Contradiction!

Halting Problem

Computational problem: Given a program (TM) and input w , does that program halt (either accept or reject) on input w ?

Formulation as a language:

$$HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that halts on input } w \}$$

Ex. $M =$ "On input x (a natural number in binary):

For each $y = 1, 2, 3, \dots$:

If $y^2 = x$, **accept**. Else, continue."

$\langle M, 100 \rangle$
 $\in HALT_{TM}$
binary enc of 4

↙ 5
Is $\langle M, 101 \rangle \in HALT_{TM}$?

*M loops on 101
 \Rightarrow M does not halt on input 101
 $\Rightarrow \langle M, 101 \rangle \notin HALT_{TM}$*

- a) Yes, because M accepts on input 101
- b) Yes, because M rejects on input 101
- c) No, because M rejects on input 101
- d) No, because M loops on input 101



Halting Problem

Computational problem: Given a program (TM) and input w , does that program halt on input w ?

Formulation as a language:

$$HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that halts on input } w \}$$

Ex. $M =$ “On input x (a natural number in binary):
For each $y = 1, 2, 3, \dots$:
If $y^2 = x$, **accept**. Else, continue.”

$\langle M, 101 \rangle \notin HALT_{TM}$

$M' =$ “On input x (a natural number in binary):
For each $y = 1, 2, 3, \dots, x$:
If $y^2 = x$, **accept**. Else, continue.
Reject.”

$\langle M', 101 \rangle \in HALT_{TM}$

$\langle M', w \rangle \in HALT_{TM}$

for all w

(since M' decider)

Halting Problem

$$HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that halts on input } w \}$$

Theorem: $HALT_{TM}$ is undecidable

Proof: Suppose for contradiction that there exists a decider H for $HALT_{TM}$. We construct a decider for A_{TM} as follows:

$\{ \langle M, w \rangle \mid TM \text{ } M \text{ accepts } w \}$

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

input to A_{TM}

1. Run H on input $\langle M, w \rangle$
 2. If H rejects, **reject**
 3. If H accepts, run M on w
 4. If M accepts, **accept**
- Otherwise, **reject**.

Claim of correctness: If H decides $HALT_{TM}$ then this TM decides A_{TM}

- If $\langle M, w \rangle \in A_{TM}$, M accepts on w
 $\Rightarrow M$ halts on input w
 $\Leftrightarrow \langle M, w \rangle \in HALT_{TM} \Rightarrow H$ accepts input $\langle M, w \rangle$
 M accepts $w \Rightarrow TM$ accepts

- If $\langle M, w \rangle \notin A_{TM}$, then either:
 - 1) M rejects input $w \Rightarrow M$ halts on w
 $\Leftrightarrow \langle M, w \rangle \in HALT_{TM}$
 $\Rightarrow H$ accepts input $\langle M, w \rangle$
 - 2) M loops on $w \Rightarrow M$ does not halt on w
 $\Leftrightarrow \langle M, w \rangle \notin HALT_{TM}$
 $\Rightarrow H$ rejects input $\langle M, w \rangle$
 $\Rightarrow TM$ overall rejects

This is a reduction from A_{TM} to $HALT_{TM}$

- 2) M loops on $w \Rightarrow M$ does not halt on w
 $\Leftrightarrow \langle M, w \rangle \notin HALT_{TM}$
 $\Rightarrow H$ rejects input $\langle M, w \rangle$
 $\Rightarrow TM$ overall rejects

Halting Problem

Computational problem: Given a program (TM) and input w , does that program halt on input w ?

- A central problem in formal verification
- Dealing with undecidability in practice:
 - Use heuristics that are correct on most real instances, but may be wrong or loop forever on others
 - Restrict to a “non-Turing-complete” subclass of programs for which halting is decidable
 - Use a programming language that lets a programmer specify hints (e.g., loop invariants) that can be compiled into a formal proof of halting

Empty language testing for TMs

$$E_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

Theorem: E_{TM} is undecidable

Proof: Suppose for contradiction that there exists a decider R for E_{TM} . We construct a decider for A_{TM} as follows:

On input $\langle M, w \rangle$: input to A_{TM}

1. Run R on input ???

$$R(\langle M \rangle) = \begin{cases} \text{accept if } L(M) = \emptyset \\ \text{reject if } L(M) \neq \emptyset \end{cases}$$

↓
not informative about whether $w \in L(M)$

This is a reduction from A_{TM} to E_{TM}

Empty language testing for TMs



$$E_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

Theorem: E_{TM} is undecidable

Proof: Suppose for contradiction that there exists a decider R for E_{TM} . We construct a decider for A_{TM} as follows:

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

1. Construct a TM N as follows:

$L(N) \neq \emptyset \Leftrightarrow R(\langle N \rangle)$ rejects
 $\Leftrightarrow \langle M, w \rangle \in A_{\text{TM}}$
 $\Leftrightarrow M$ accepts w

$L(N) = \emptyset \Leftrightarrow R(\langle N \rangle)$ accepts
 $\Leftrightarrow \langle M, w \rangle \notin A_{\text{TM}}$
 $\Leftrightarrow M$ does not accept w

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

3. If R rejects, **accept**. Otherwise, **reject**

What do we want out of machine N ?

a) $L(N)$ is empty iff M accepts w

b) $L(N)$ is non-empty iff M accepts w

c) $L(M)$ is empty iff N accepts w

d) $L(M)$ is non-empty iff N accepts w

This is a reduction from A_{TM} to E_{TM}

Empty language testing for TMs

$$E_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

Theorem: E_{TM} is undecidable

Proof: Suppose for contradiction that there exists a decider R for E_{TM} . We construct a decider for A_{TM} as follows:

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

1. Construct a TM N as follows:

“On input x : *• I ignore x*

Run M on w and output
the result.”

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

3. If R rejects, **accept**. Otherwise, **reject**

$$\text{(Claim: } L(N) \neq \emptyset \iff M \text{ accepts } w)$$

$$1) M \text{ accepts } w \implies L(N) = \Sigma^*$$

$$2) M \text{ does not accept } w \implies L(N) = \emptyset$$

This is a reduction from A_{TM} to E_{TM}