BU CS 332 – Theory of Computation

https://forms.gle/PaSZkLFEFkaohxmL9

Lecture 16:

- More on Reductions

Reading:

Sipser Ch 5.1

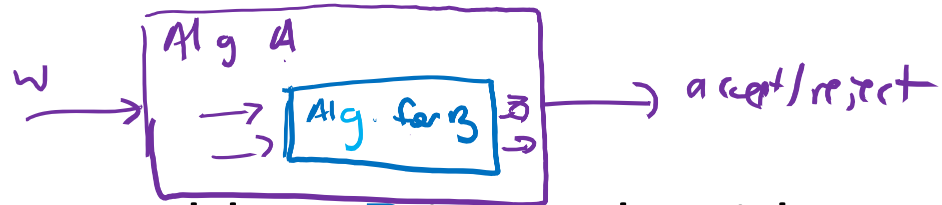
HW 7 —
Due Saturday 11:59PM
11/5

Mark Bun

November 3, 2022

Reductions

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"

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

Ex. E_{DFA} is decidable $\Rightarrow$ EQ_{DFA} is decidable

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

Ex. UD is undecidable $\Rightarrow$ A_{TM} is undecidable

Two uses of reductions

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

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

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

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

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

5

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

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 (either accept or reject) on input w?

Formulation as a language:

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

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

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

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

M' = "On input x (a natural number in binary):

For each $y = 1, 2, 3, \ldots, x$:

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

Reject."

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

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

$A_{TM} = \{\langle M, w\rangle \mid$ TM M accepts on input $w\}$

Halting Problem

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

$H(\langle M,w\rangle)$ accepts if M halts on w
rejects if M loops on w

Theorem: $HALT_{\mathrm{TM}}$ is undecidable

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

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

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.

Use H to check if M halts on w.

Simulate M on w, which terminates because we know M halts on w

Claim: If H decides $HALT_{TM}$, then V decides A_{TM}

1) $\langle M,w\rangle \in A_{TM} \Rightarrow M$ accepts on w
$\Rightarrow \langle M,w\rangle \in HALT_{TM} \Rightarrow H(\langle M,w\rangle)$ accepts
Alg. goes to step 3, where M accepts w
$\Rightarrow V$ accepts in step 4.

2) $\langle M,w\rangle \notin A_{TM}$
a) M loops on $w \Rightarrow \langle M,w\rangle \notin HALT_{TM}$
$\Rightarrow H(\langle M,w\rangle)$ rejects
$\Rightarrow V$ rejects in step 2
b) M rejects on $w \Rightarrow \langle M,w\rangle \in HALT_{TM}$
$\Rightarrow H(\langle M,w\rangle)$ accepts $\Rightarrow$ Alg. goes to step 3 where M rejects $w \Rightarrow V$ rejects.

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

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

Emptiness 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. Run R on input ??? M_2

M_2 = On input x:
If $x = w$: Run M on w. If accepts, accept. If rejects, reject.
Else: reject

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

Emptiness testing for TMs

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

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:

If $\langle M, w \rangle \in A_{TM} \Rightarrow L(N) \neq \emptyset \Rightarrow R$ rejects $\langle N \rangle$
$\Rightarrow$ decider accepts
If $\langle M, w \rangle \notin A_{TM} \Rightarrow L(N) = \emptyset \Rightarrow R$ accepts $\langle N \rangle$
$\Rightarrow$ decider rejects

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}

Emptiness testing for TMs

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

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: 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

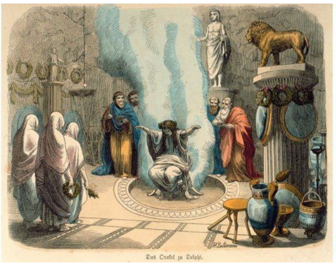
Claim: This TM decides A_{TM}

1) If $\langle M,w\rangle \in A_{TM}$
$\Rightarrow L(N) = \Sigma^*$
$\Rightarrow R(\langle N\rangle)$ rejects
$\Rightarrow$ TM overall accepts

2) If $\langle M,w\rangle \notin A_{TM}$
$\Rightarrow L(N) = \emptyset$
$\Rightarrow R(\langle N\rangle)$ accepts
$\Rightarrow$ TM overall rejects

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

Interlude: Formalizing Reductions (Sipser 6.3)



Informally: A reduces to B if a decider for B can be used to construct a decider for A

One way to formalize:

- An *oracle* for language B is a device that can answer questions "Is $w \in B$?"
- An *oracle TM* M^B is a TM that can query an oracle for B in one computational step

A is Turing-reducible to B (written $A \leq_T B$) if there is an oracle TM M^B deciding A

Equality Testing for TMs

$$EQ_{\text{TM}} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

Theorem: EQ_{TM} is undecidable

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

On input $\langle M\rangle$:

1. Construct TMs N_1, N_2 as follows:

   N_1 = N_2 =

2. Run R on input $\langle N_1, N_2\rangle$
3. If R accepts, accept. Otherwise, reject.

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

Equality Testing for TMs

What do we want out of the machines N_1, N_2?

a) $L(M) = \emptyset$ iff $N_1 = N_2$

b) $L(M) = \emptyset$ iff $L(N_1) = L(N_2)$

c) $L(M) = \emptyset$ iff $N_1 \neq N_2$

d) $L(M) = \emptyset$ iff $L(N_1) \neq L(N_2)$

On input $\langle M \rangle$: # instance of E_{TM}, i.e., want to accept $\Leftrightarrow$ $L(M) = \emptyset$

1. Construct TMs N_1, N_2 as follows:

N_1 = "On input x:
reject"

$L(N_1) = \emptyset$

N_2 = M

$L(N_2) = L(M)$

2. Run R on input $\langle N_1, N_2 \rangle$ # $R(\langle N_1, N_2 \rangle)$ accepts $\Leftrightarrow$ $L(N_1) = L(N_2)$

3. If R accepts, accept. Otherwise, reject.

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

Equality Testing for TMs

$$EQ_{\text{TM}} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

Theorem: EQ_{TM} is undecidable

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

On input $\langle M\rangle$:

1. Construct TMs N_1, N_2 as follows:

   N_1 = N_2 =

2. Run R on input $\langle N_1, N_2\rangle$
3. If R accepts, accept. Otherwise, reject.

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

Regular language testing for TMs

$$REG_{\mathrm{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is regular}\}$$

Theorem: REG_{TM} is undecidable

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

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

1. Construct a TM N as follows:

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

3. If R accepts, accept. Otherwise, reject

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

Regular language testing for TMs

$$REG_{\mathrm{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is regular}\}$$

Theorem: REG_{TM} is undecidable

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

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

1. Construct a TM N as follows:

    N = "On input x,

    1. If $x \in \{0^n 1^n \mid n \geq 0\}$, accept
    2. Run TM M on input w
    3. If M accepts, accept. Otherwise, reject."

2. Run R on input $\langle N \rangle$
3. If R accepts, accept. Otherwise, reject

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