Lecture 4

Frank

February 2, 2022

1 General (Turing) Reductions

1.1 Definition of Reduction

- P **(Turing)-reduces** to Q:

  $\exists$ algo for P that uses algo for Q as a "black box"

  Black box can also be interpreted as a **subroutine**

  Represented by $P \leq_T Q$ (Turing) reduction

- Formalizing reduction (no need to worry about):

  TM with oracle for Q:

  - 2-tape TM
  - States $q_?, q_Y, q_N$

    Enter $q_?$ upon call to Q, then immediate returns q_Y if the current input string is a string in the language of the TM, and q_N otherwise, this is our oracle.

2 Halting Problem

2.1 Halting Problem (Language)

- $H = \{\langle M, x\rangle : \text{TM } M \text{ halts on input } x\}$
- **Thm 4.1**: H is a) recognizable, but b) not decidable

  Proof a)

  Use M_u to simulate H on input x, if M_u halts, then H recognizes x

  Proof b) Show that $U \leq H$

  Given H-decider TM M_1, construct U-decider TM M_2

  $M_2 =$ on input $\langle M, x\rangle$

  1. Run M_1 on $\langle M, x\rangle$
  2. If M_1 accepts (M halts on x), then run M_u on $\langle M, x\rangle$.

     If M_u accepts, accept

     Else reject
  3. Else (M doesn't halt on x) reject

  Thus, M_2 decides U, but by Thm 3.x, U is not decidable, thus M_2 does not exist, but M_2's existence depends on $M_1 \implies M_1$ does not exist

  $\therefore H$ is undecidable

Example of M_2

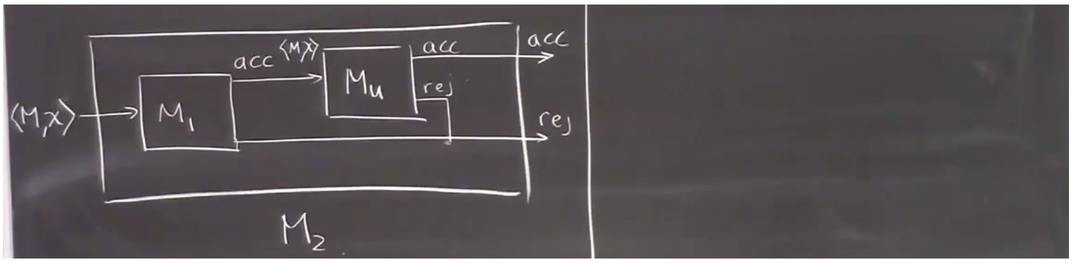

Figure 1: l41

2.2 Different reduction of b) (Alt. proof)

- Given $H-$decider M_3, construct a $U-$decider M_4.

  $M_4 =$ on input $\langle M, x\rangle$.

  1. Modify M to M' by changing every transition of M to the reject state into an inf. loop.

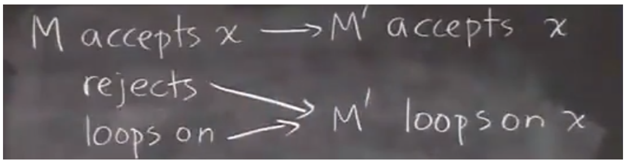

Figure 2: l42

  2. Run M_3 on $\langle M', x\rangle$
  3. If M_3 accepts, then accept
  4. Else reject

  M_4 is a decider for U, because M_4 accepts $\langle M, x\rangle \iff M_3$ accepts $\langle M', x\rangle \iff M'$ halts on x [M_3 is H-decider] $\iff M$ accepts x (M' will loop if not accept)

  Then M_4 is a $U-$recognizer that will always halt

  $\therefore M_4$ is a $U-$decider

  $\implies$ contradiction.

Example of M_4

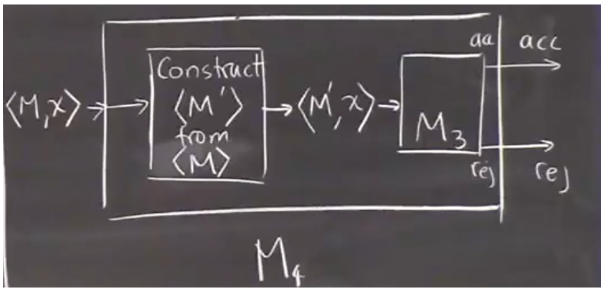

Figure 3: l43

2.3 Digression

- Can also show $H \leq U$ similarly (similar to the second reduction)
- Given input $\langle M, x \rangle$ to H, construct input $\langle M', x \rangle$ to U as follows

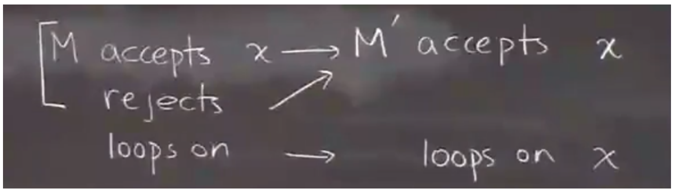

Figure 4: l44

We can also correctly show that $H \leq U$.

- However, this does not prove that H is undecidable (wrong direction!)

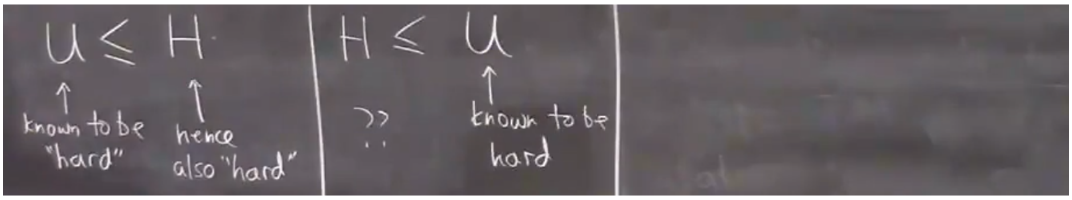

Figure 5: l45

2.4 Cor 4.2

- **Cor 4.2**: $\bar{H} = \{\langle M, x \rangle : M \text{ does not halt on } x\}$ is **unrecognizable**, this is by the fact that H is recognizable but not decidable.

3 Mapping Reductions

3.1 Definition

- Let $P, Q \subseteq \Sigma^\star$ be languages. P is mapping reducible to Q, $P \leq_m Q$, iff $\exists$ **computable** function $f : \Sigma^\star \to \Sigma^\star$ s.t. $x \in P \iff f(x) \in Q$

  Example:

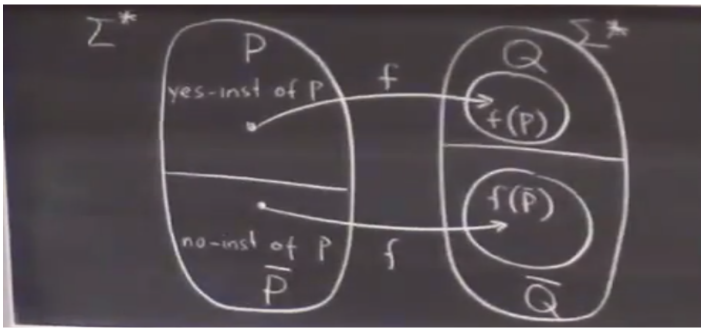

Figure 6: l46

- Important:
  - f must be computable (theres a TM that can compute it). Describe an algorithm that computes f
  - f maps yes-instances of P to yes-instances of Q **and** no-instances of P to no-instances of Q
  - f need not be (and usually is not) onto.

3.2 Example

- Consider $\bar{D} = \{\langle M \rangle : M \text{ accepts } \langle M \rangle\}$

  We actually did $\bar{D} \leq_m U$

  $f : \langle M \rangle \to \langle M, \langle M \rangle \rangle$

- The second reduction we did for $U \leq H$ is actually $U \leq_m H$

  Given $\langle M, x \rangle$, constructed $\langle M', x \rangle$ s.t. M accepts $x \iff M'$ halts on x.

  $f : \langle M, x \rangle \in U \to \langle M', x \rangle \in H$

- Note that turing reduction is a more general form of reduction, so the first reduction we did was a turing reduction, but not a mapping reduction.

3.3 Aside

- Mapping reduction is sometimes called many-to-one reduction

3.4 Properties of Mapping Reductions

- **Thm 4.3**: Suppose $P \leq_m Q$

  If Q is decidable, then P is decidable

  $\iff$

  If P is undecidable, then Q is undecidable ($\star$)

- Proof:

  - Assume that $P \leq_m Q$ and P is undecidable.
  - Suppose for contra, Q is decidable.

    Let D_Q be a decider for Q

    Since $P \leq_m Q$, $\exists$ computable f s.t. $x \in P \iff f(x) \in Q$

    Then the following algorithm is a decider for P

    $D_P =$ On input x do:

    1. Compute $f(x)$
    2. Run D_Q on $f(x)$

       If D_Q accepts, then accept

       Else reject

    Note that D_P halts on all inputs $\implies$ it's a decider

    D_P accepts $x \iff D_Q$ accepts $f(x) \iff f(x) \in Q$ [D_Q is $Q-$decider] $\iff x \in P$ [because f is a mapping reduction of P to Q] $\iff D_P$ is $P-$decider

    Which contradicts the fact that P is undecidable

    $\therefore Q$ is undecidable

- **Thm 4.4**: Suppose $P \leq_m Q$

  If Q is recognizable, then P is recognizable

  $\iff$

  If P is unrecognizable, then Q is unrecognizable

  The proof is very similar to the decidability proof, except that D_Q is now R_Q, a recognizer, which might loop. However, this is not a problem because we are building a recognizer, so R_Q will get stuck on the inputs that R_P will get stuck on.

- **Thm 4.5**: If $P \leq_m Q$ then $\bar{P} \leq_m \bar{Q}$

  Proof is very simple, f is the mapping function here again:

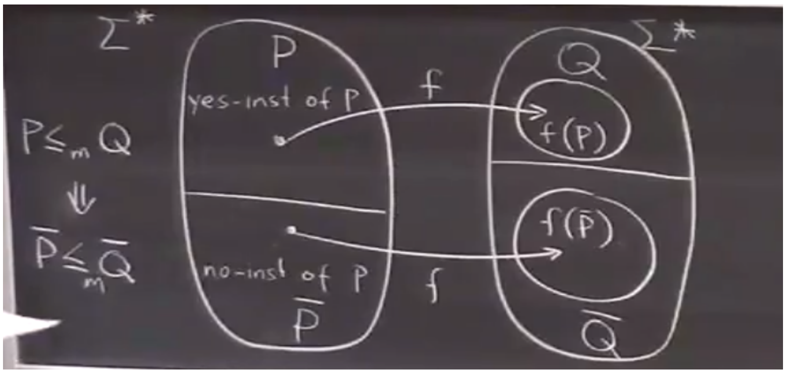

Figure 7: l47

- **Thm 4.6** If $P \leq_m Q$ and $Q \leq_m R$, then $P \leq_m R$

  Proof: Exercise (on tb too)

4 Examples

4.1 Proving undecidability

- To prove P is undecidable (unrecognizable), it suffices to prove $U \leq_m P : U$ undecidable ($\bar{U} \leq_m P : \bar{U}$ unrecognizable) By thm 4.3 and 4.4.

4.2 Ex 1

- Define $E = \{\langle M \rangle : L(M) = \emptyset\}$
- **Thm 4.7**: E is unrecognizable

  Proof:

  - It suffices to prove that $\bar{U} \leq_m E$

    Given $\langle M, x\rangle$ [input to $\bar{U}$], construct $\langle M' \rangle$ [input to E] s.t. M does not accept x ($\langle M, x\rangle \in \bar{U}$) $\iff L(M') = \emptyset$ ($\langle M' \rangle \in E$)

    Build M' s.t. if M doesn't accept x, M' accepts nothing, and if M accepts x, M' accepts everything

– $f =$ on input $\langle M, x \rangle$:

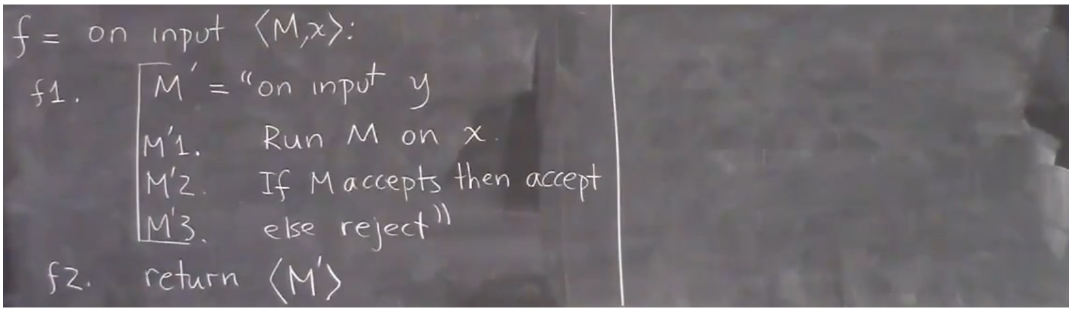

Figure 8: l48

Note that M''s input y is ignored, and M''s outcome is solely dependent on whether M accepts x

If M does not accept x, then $L(M') = \emptyset$

If M accepts x, then $L(M') = \Sigma^\star$

- **Claim**: f is a mapping reduction of $\bar{U}$ to E

  Verify that $\langle M, x \rangle \in \bar{U} \iff \langle M' \rangle \in E$

- $(\implies)$

  $\langle M, x \rangle \in \bar{U} \implies M$ does not accept $x \implies M'$ accepts no input $\implies L(M') = \emptyset \implies \langle M' \rangle \in E$

- $(\impliedby)$

  $\langle M, x \rangle \notin \bar{U} \implies M$ accepts $x \implies M'$ accepts all inputs $\implies L(M') \neq \emptyset \implies \langle M' \rangle \notin E$

4.3 Ex 2

- **Thm 4.8**: $\bar{E} = \{\langle M \rangle : L(M) \neq \emptyset\}$ is a) undecidable, but b) recognizable

  Proof a)

  Suppose for contra, that $\bar{E}$ is decidable.
  $\implies \bar{\bar{E}} = E$ is decidable. (Thm 3.3)
  $\implies E$ is recognizable, contradicting Thm 4.7

Proof b)

Idea: dovetail through all pairs (i, j), until we find an input that M accepts, at which point, $\langle M \rangle$ belongs to $\bar{E}$

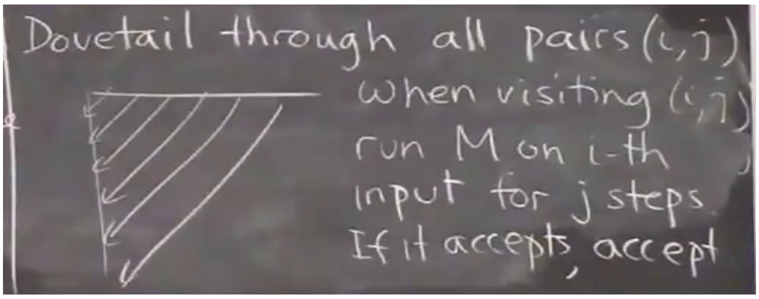

Figure 9: l49

Note we don't care about going on forever, we want recognizability (being able to accept)

Proof b) (non-determinism)

A NTM recognizes $\bar{E}$ as follows:

On input $\langle M \rangle$

1. nondeterministically "guess" a string x
2. Use universal TM to run M on x
3. If M accepts x, then accept

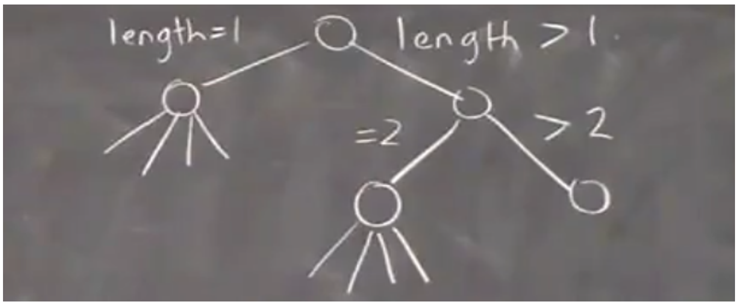

Figure 10: l410

4.4 Ex 3

- Let $REG = \{\langle M \rangle : L(M) \text{ is regular}\}$
- **Thm 4.9**: REG is undecidable

  Proof: Suffices to show $U \leq_m REG$

  Given $\langle M, x \rangle$ [input to U]

  Construct $\langle M' \rangle$ [input to REG] s.t. M accepts $x \iff L(M')$ is regular

  Or $\langle M, x \rangle \in U \iff \langle M' \rangle \in REG$

- M accepts $x \implies M'$ accepts reg language.
  ex. $L(M') = \Sigma^\star$ (regular)
- M does not accept $x \implies M'$ accept a non-reg language.
  ex. $\{0^n 1^n : n \in \mathbb{N}\}$ (not regular)
- $f =$ on input $\langle M, x\rangle$:

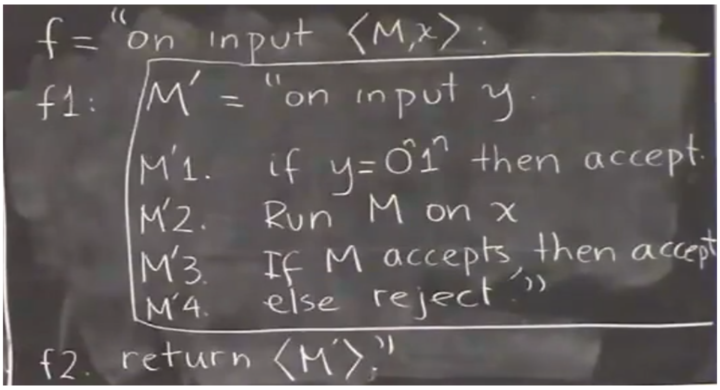

Figure 11: l411

Verify $\langle M, x\rangle \in U \iff \langle M'\rangle \in REG$

$(\implies)$ $\langle M, x\rangle \in U$

$\implies M$ accepts x

$\implies M'$ accepts all inputs y [in line M'1, if $y = 0^n 1^n$; or in line M'3 o.w.]

$\implies L(M') = \Sigma^\star$

$\implies \langle M'\rangle \in REG$

$(\impliedby)$ $\langle M, x\rangle \notin U$

$\implies M$ does not accept x

$\implies M'$ accepts all and only strings of the form $0^n 1^n$

$\implies L(M')$ is not regular

$\implies \langle M'\rangle \notin REG$

4.5 Exercise

- Ex: Show $U \leq_m R\bar{E}G$, where $R\bar{E}G = \{\langle M\rangle : L(M)$ is not regular $\}$
- Note $U \leq_m REG$ (Thm 4.9) $\implies \bar{U} \leq_m R\bar{E}G$ (Thm 4.5)

$\implies R\bar{E}G$ is unrecognizable

By exercise: $U \leq_m R\bar{E}G$

$\implies \bar{U} \leq_m REG$

$\implies REG$ is unrecognizable