§1 2020-05-25

§1.1 Reductions

P reduces to Q means if I can solve Q I can solve P. This is roughly equivalent to Q is "harder than" P.

Algorithm for solving P. First transform the problem into a Q problem. Then feed the problem to the Q solver.

If you know P is undecidable than the putative Q solver cannot exist, so Q is also undecidable.

$\leq$ is a preorder. Transitive so reductions can be chained. Not partial order because it doesn't have anti-symmetry.

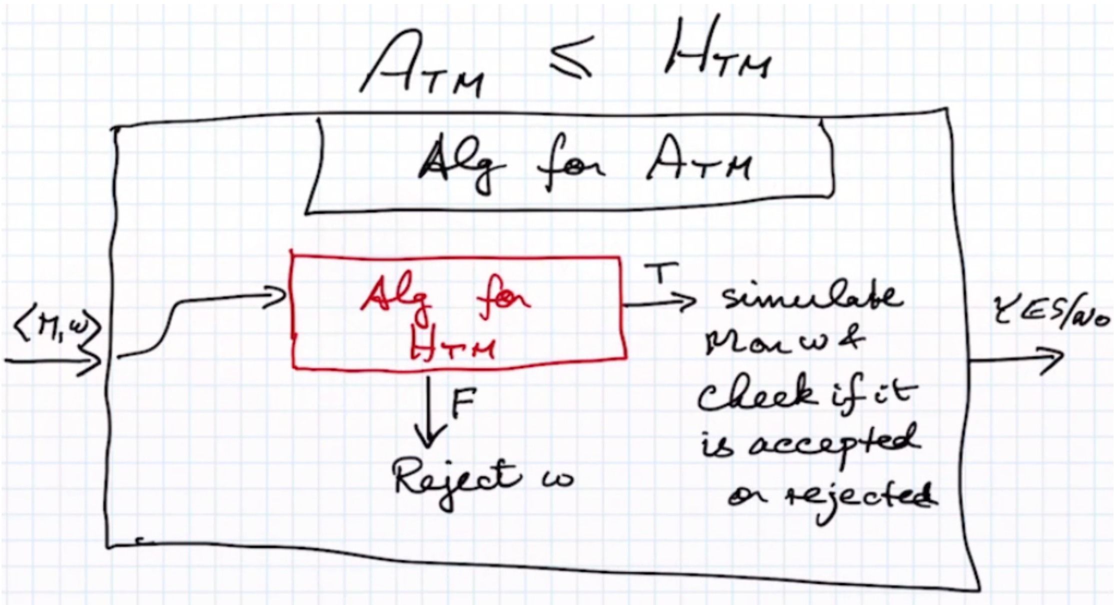
Notation. $\langle M \rangle$ is encoding of a machine. $\langle M, w \rangle$ is a machine and its input. $\langle M_1, M_2 \rangle$ is two machines. $\langle G \rangle$ encodes a CFG.

$$H_{TM} = \{\langle M, w \rangle \mid M \text{ halts on } w\}$$
$$A_{TM} = \{\langle M, w \rangle \mid M \text{ accepts } w\}$$

H_{TM} accepts the set of machines and input that halt. A_{TM} accepts the set of machines and inputs that those machines accept. $H_{TM} \leq A_{TM}$.

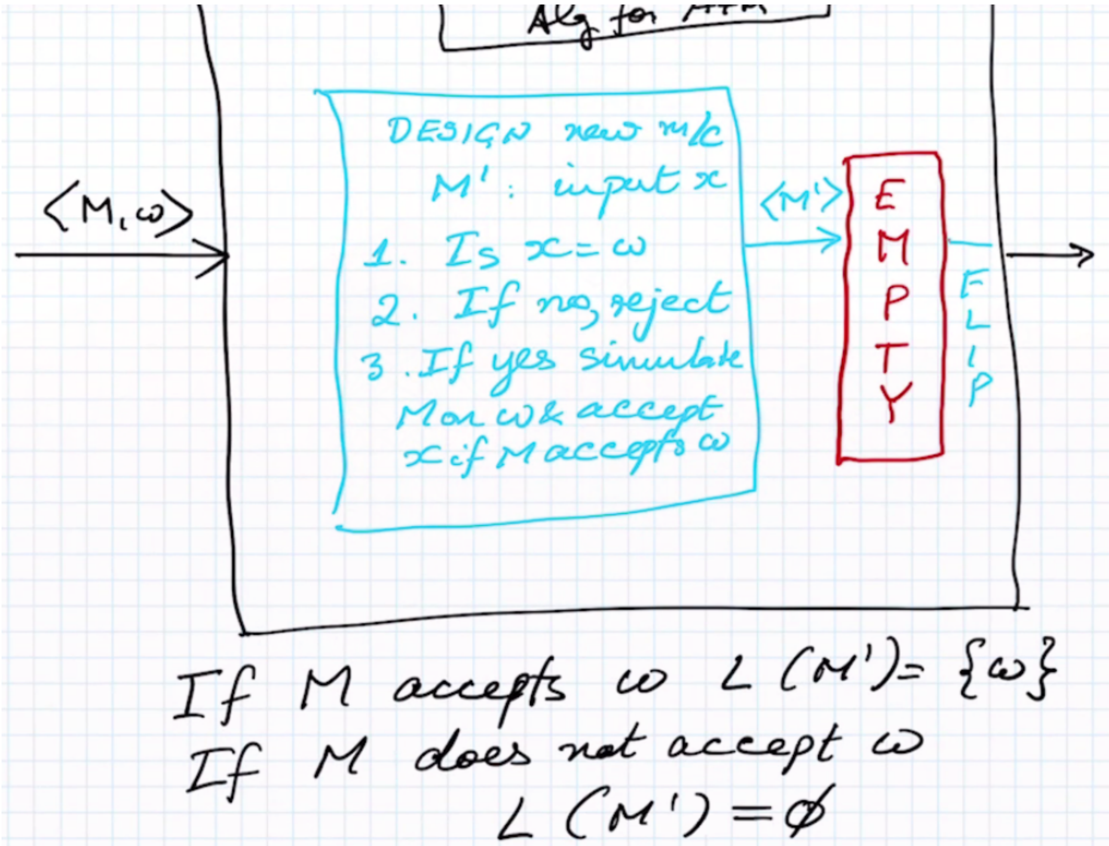
Note 1.1. Constructing a machine should be thought of as writing the code for the machine.

Algorithm for A_{TM} cannot exist because H_{TM} reduces to it.



$EMPTY_{TM} = \{\langle M \rangle \mid L(M) = \varnothing\}$.

$A_{TM} \leq EMPTY_{TM}$. Machine:



REG, is $L(M)$ a regular language? Construct machine.

Note 1.2. The more powerful a gadget is, the easier it is to build a reduction to that gadget.

$EQ_{TM} = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$. $EMPTY_{TM} \leq EQ_{TM}$. Construct a machine that rejects all words, and compare equality of input to $EMPTY$ with this machine using EQ machine. If they are equal, then M is empty.

Exercise 1.3. Show that the following are undecideable.

1. $L(M) = L(M')$ where M' always halts.
2. $L(M)$ is context free.
3. $|L(M)| < \infty$.
4. $L(M) = \Sigma^*$.

§1.2 Sharper Notion of Reduction

Mapping reduction. Suppose $L_1, L_2 \subseteq \Sigma^*$. We say L_1 is mapping reducible to L_2

$$L_1 \leq_m L_2$$

if there exists a total computable function $f : \Sigma^* \to \Sigma^*$ such that $\forall w \in \Sigma^*,\ w \in L_1 \Leftrightarrow f(w) \in L_2$.

Note 1.4.

1. $L_1 \leq_m L_2$ then $\overline{L_1} \leq_m \overline{L_2}$. This is not true for general reductions.

2. $\leq_m$ has a direction because f has a direction. No guarantee that you can go both ways. $L_1 \leq_m L_2$ does not mean $L_2 \leq_m L_1$.

Fact 1.5. Facts about $\leq_m$.

1. $P \leq_m Q$ and P is undecidable then Q is undecidable

2. $P \leq_m Q$ and Q is decidable then P is decidable.

3. $P \leq_m Q$ and Q is computably enumerable then P is also computably enumerable.

4. $P \leq_m Q$ and P is not computably enumerable, then Q cannot be computably enumerable.

5. If $P \leq_m Q$ and P is not co-CE then Q cannot be co-CE. co-CE means if the algorithm is NO your algorithm will definitely tell you. CE means if the answer is YES your algorithm will definitely tell you.

   H_{TM}, A_{TM} are CE but not coCE. Run it and it will tell you if the answer is yes.

   $\overline{A_{TM}}, EMPTY_{TM}$ are co-CE. You will definitely find out if it is not empty with dove tailing.

Semi decision problem, computably enumerable set.

$A_{TM} \leq EMPTY_{TM}$ but this is not a mapping reduction. Suppose we had $A_{TM} \leq_m EMPTY_{TM}$, then $\overline{A_{TM}} \leq_m \overline{EMPTY_{TM}}$. But this is not possible because $\overline{A_{TM}}$ is co-CE while $\overline{EMPTY_{TM}}$ is CE.

§1.3 Turing Reduction

$P \leq_T Q$. I get to use a Q oracle as many times as I want and I can do any computable post processing I want.

$P \leq_m Q$. I get to do some total computable preprocessing and then ask my Q oracle 1 question and output the answer without post processing. Can't even flip the output.

> **Theorem 1.6**
> EQ_{TM} is not CE or coCE. More difficult than halting problem.

Fact 1.7. Halting problem is complete for all CE problems. CE complete.

Non-halting problem is coCE complete.

$|L(M)| = \infty$. INF $= \{\langle M \rangle \mid |L(M)| = \infty\}$

Claim: $\overline{H_{TM}} \leq_m INF$.

Reducing of non halting problem. Let input to M' be X, then run M on w $|x|$ times and reject x if it halts, accept otherwise. The language of M' is everything if w runs forever and is finite otherwise, which can be checked with INF.

Theorem 1.8 (Rice's Theorem)
$P : \mathbb{N} \to \mathbb{N}$. $[\![P]\!] := \{(x, y) \mid P(x) = y\}$.

$P_1 \sim P_2$ means $[\![P_1]\!] = [\![P_2]\!]$.

P_1, P_2 are extensionally equal.

$M_1 \sim M_2 \Leftrightarrow L(M_1) = L(M_2)$. $Q : PROG \to \{T, F\}$ is called a property of programs. Q is an extensional property if $P_1 \sim P_2 \Leftrightarrow Q(P_1) = Q(P_2)$. Q only depends on the IO behavior. Q only depends on the functional spec.

Q always true or Q always false are trivial properties.

Rice's theorem: Every non trivial extensional property of programs is undecidable. Nothing that just depends on the IO spec can possibly be decidable.

Proof. Let Q be a nontrivial property of CE sets. i.e. $\exists P$ such that $Q(P) = true$ and $\exists P'$ such that $Q(P') = false$.

Assume empty does not satisfy Q. i.e. $\forall M$ if $L(M) = \varnothing$ then $Q(M) = F$.

Let M_0 be such that $Q(M_0) = T$. Then $L(M_0) \neq \varnothing$ by our assumption.

$$L_q = \{\langle M \rangle \mid Q(M) = T\}$$

Claim: $A_{TM} \leq_m L_Q$. Have gadget to solve $x \in L_Q$.

M' with input x construction: Simulate M on w. If M accepts w then simulate M_0 on x. $\square$