Comp 330 - Lecture 20 - November 9th

IE: Solita mineotrea
The usual soup
The usual routine

Showing that a problem is undecidable via REDUCTIONS

Recall Languages / Decision Problems

Def A DP is a problem where given an instance of that problem the answer is either Y or N.

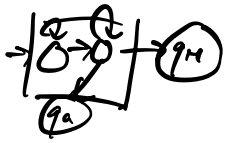
Ex 1 PRIME(n): "Given $n \in \mathbb{N}$, is n prime?"
$n = 7$, Y

HTM
↕
Ex 2 HP(M, x): "Given a TM M and an input string x, does M halt on x?"

$\Sigma = \{0,1\}$?

instance: M, x = 01101

Answer: N



(usually $\Sigma = \{0,1\}$)

Def $\Sigma \neq \emptyset$, The language of a DP, L_{DP}, is the language containing all instances of DP where the answer is yes & the instances are encoded as strings over Σ.

Ex1 $L_{PRIME} = \{ 0^n : n \in \mathbb{N}, n \text{ is prime} \}$

$x \in L_{PRIME} \iff$ x encodes a prime number n using $\Sigma = \{0,1\}$

Ex2 $\Sigma = \{0,1\}$ $L_{HP} = \{ \langle M, x \rangle : M \text{ is a TM}, x \in \Sigma^*, M \text{ halts on } x \}$

$\langle M, x \rangle$: binary representation of M & x

Def A DP is decidable iff L_{DP} is decidable

Showing undecidability

Who cares?

1. Decidability => Logical inconsistency
   Diagonalization

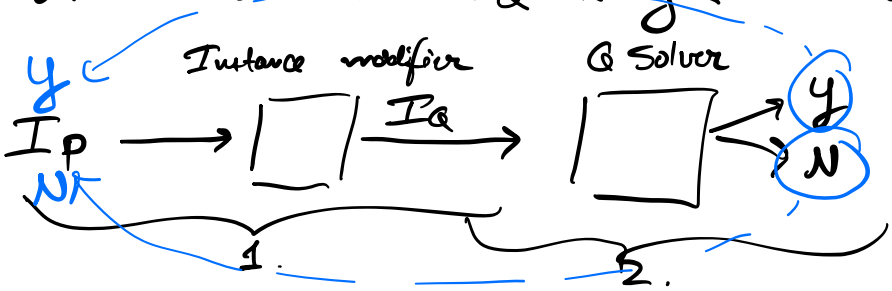
2. Reduction

Idea Given two problems P & Q. A reduction from P to Q is a method to show that Q is at least as difficult as P.

$$P \leq_m Q$$

How? Intuition: Show that if you could solve Q, you could also solve P.

1. Take instance of P, I_P, and modify it to look like an instance of Q, I_Q.
2. Answer to ~~I_Q~~ is ~~$y \Leftrightarrow$~~ Answer to I_P is y

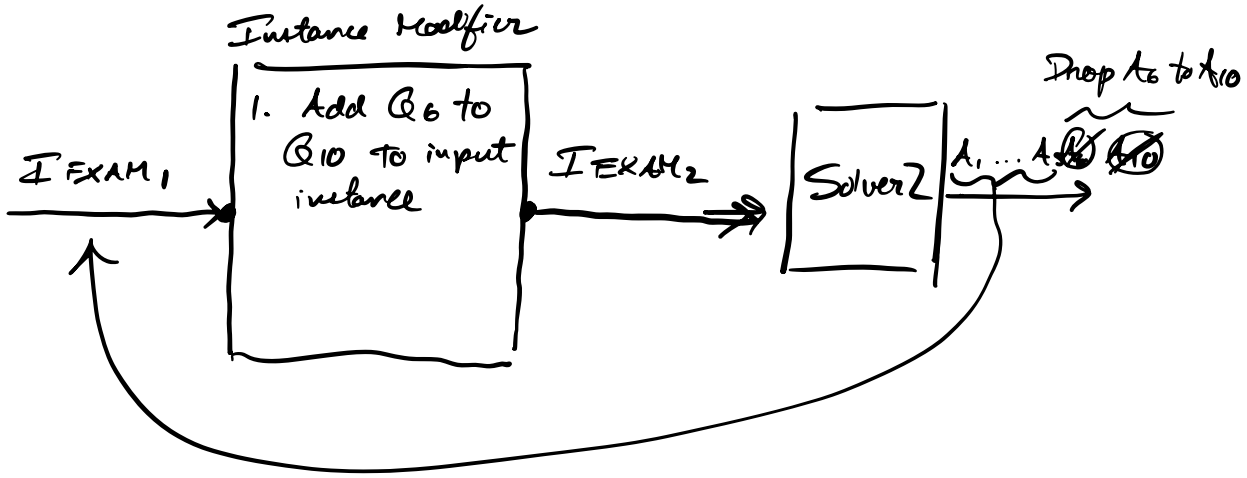


Ex For the final exam of COMP 999, there are 10 question types $Q_1, Q_2, \ldots, Q_{10}$ and two versions of the final

$EXAM_1 = Q_1\ Q_2 \ldots Q_5$

$EXAM_2 = Q_1 \ldots Q_5\ Q_6 \ldots Q_{10}$

Goal: Show $EXAM_1 \leq EXAM_2$



What's the point?

Suppose we knew EXAM1 was unsolvable, then EXAM2 must be unsolvable as well. Otherwise, we could solve EXAM1.

This is exactly the process in a mapping reduction with a few constraints

1. Instance modifier must be computable
2. No postprocessing & solver used only once

Ex A Show that the acceptance problem, AP, is undecidable by $HP \leq_m AP$.

L_{HP} L_{AP}

What is AP? AP(M, w): "Given a TM M, does M accept w?"

Mapping reduction from HP to AP

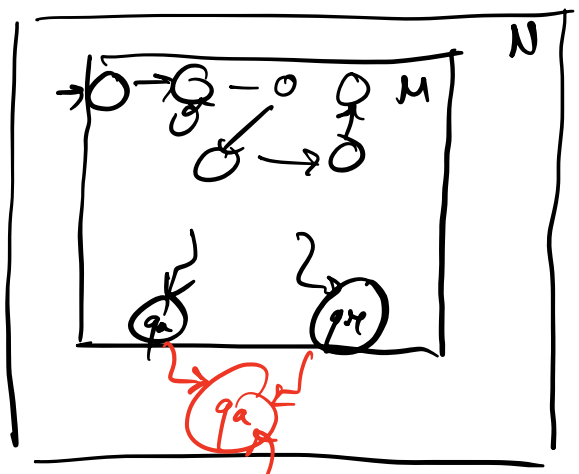
1. Convert I_{HP} to I_{AP}
2. Answer to I_{HP} is Y <=> — " — to I_{AP} is Y

1. > Given $I_{HP} := \langle M, x \rangle$, create $I_{AP} := \langle N, y \rangle$ (as a function of I_{HP})

   -> M halt on x <=> N accepts y

   -> M halt on x <=> N accepts x

$y := x$



Rewrite step 1.

N := On input string w

1. Simulate M on w
2. If M halts on w, then accept.
3. Else loop. COMMON ERROR - DON'T DO THIS

2. if M halts on x => N accepts x

if M loops on x $\Rightarrow$ N loops on x

N does not accept x

Incorrect : N rejects x

The instance modification is computable

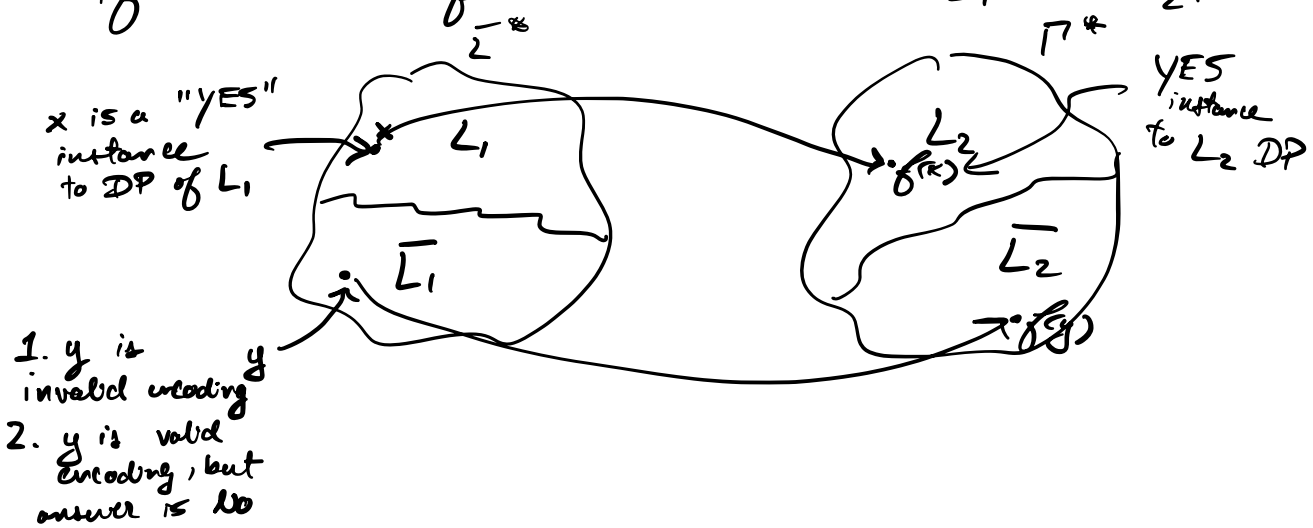
$\therefore$ $HP \leq_m AP$

Since HP is undecidable then so is AP.

Def $\Gamma, \Sigma \neq \emptyset$, $L_1 \subseteq \Sigma^*$, $L_2 \subseteq \Gamma^*$, a mapping / many-to-one reduction of L_1 to L_2 is a total computable function $f: \Sigma^* \to \Gamma^*$ where $\forall x \in \Sigma^*$

$$x \in L_1 \iff f(x) \in L_2$$

If such a function exists then $L_1 \leq_m L_2$.



f preserves the answer to L_1.

Ex In Ex A, The total computable function

$f : \{0,1\}^* \to \{0,1\}^*$ as described in Ex A.

$$f(v) = \begin{cases} \langle N, x \rangle & \text{if } v = \langle M, x \rangle \\ \varepsilon & \text{otherwise} \end{cases}$$

f is computable. Consider TM F

$F :=$ On input v

1. Checks if $v = \langle M, x \rangle$
2. If no, return ε
3. If yes, then return $\langle N, x \rangle$

$N :=$ On input...

Thm $\Sigma, \Gamma \neq \phi$, $L_1 \subseteq \Sigma^*$, $L_2 \subseteq \Gamma^*$.
$L_1 \leq_m L_2$ & L_2 is decidable, then
L_1 is also decidable.

Pf Suppose L_2 is decidable then $\exists$ TM M_2 s.t. $L(M_2) = L_2$ & M_2 halts on every input.

Since $L_1 \leq_m L_2$ then $\exists$ a mapping reduction f in f which is total computable i.e. $\exists$ a TM F which computes f.

Construct a TM M_1 which decides L_1

M_1 := On input x

1. Simulates F on x to compute $f(x)$
2. Simulate M_2 on $f(x)$

   If M_2 accepts then accept

   Otherwise reject

$L(M_1) = L_1$

$x \in L(M_1) \iff M_2$ accepts $f(x)$ i.e. $f(x) \in L(M_2)$

$\iff f(x) \in L_2$

$\iff x \in L_1$ f is a mapping reduction

Contrapositive $L_1 \leq_m L_2$ & L_1 is undecidable then so is L_2

Exercise T/F $L_1 \leq_m L_2$ then $\overline{L_1} \leq_m \overline{L_2}$.

Exercise Show correctness of $L_{HP} \leq_m L_{AP}$.

Post later.

Ex Show that the ALL is undecidable by $AP \leq_m ALL$.

$ALL(M)$: "Given TM M, is $L(M) = \Sigma^*$?"

1. Given I_{AP} (= $\langle M, x \rangle$) convert to I_{ALL} (= $\langle N \rangle$)

2. Answer to I_{AP} is y <=> answer to I_{ALL} is y

1. N := On input w
   1. Erase w from tape
   2. Load x onto tape // How? x is hardcoded in N's control unit
   3. Simulate M on x
   4. If M accepts x then accept.

y ↓ ↑ y

2. M accepts x => L(N) = Σ^*

   M does not accept x => N does not accept any input string. L(N) = ϕ

N ↓ ↑ N

$AP \leq_m ALL$ since AP is undecidable so is ALL.