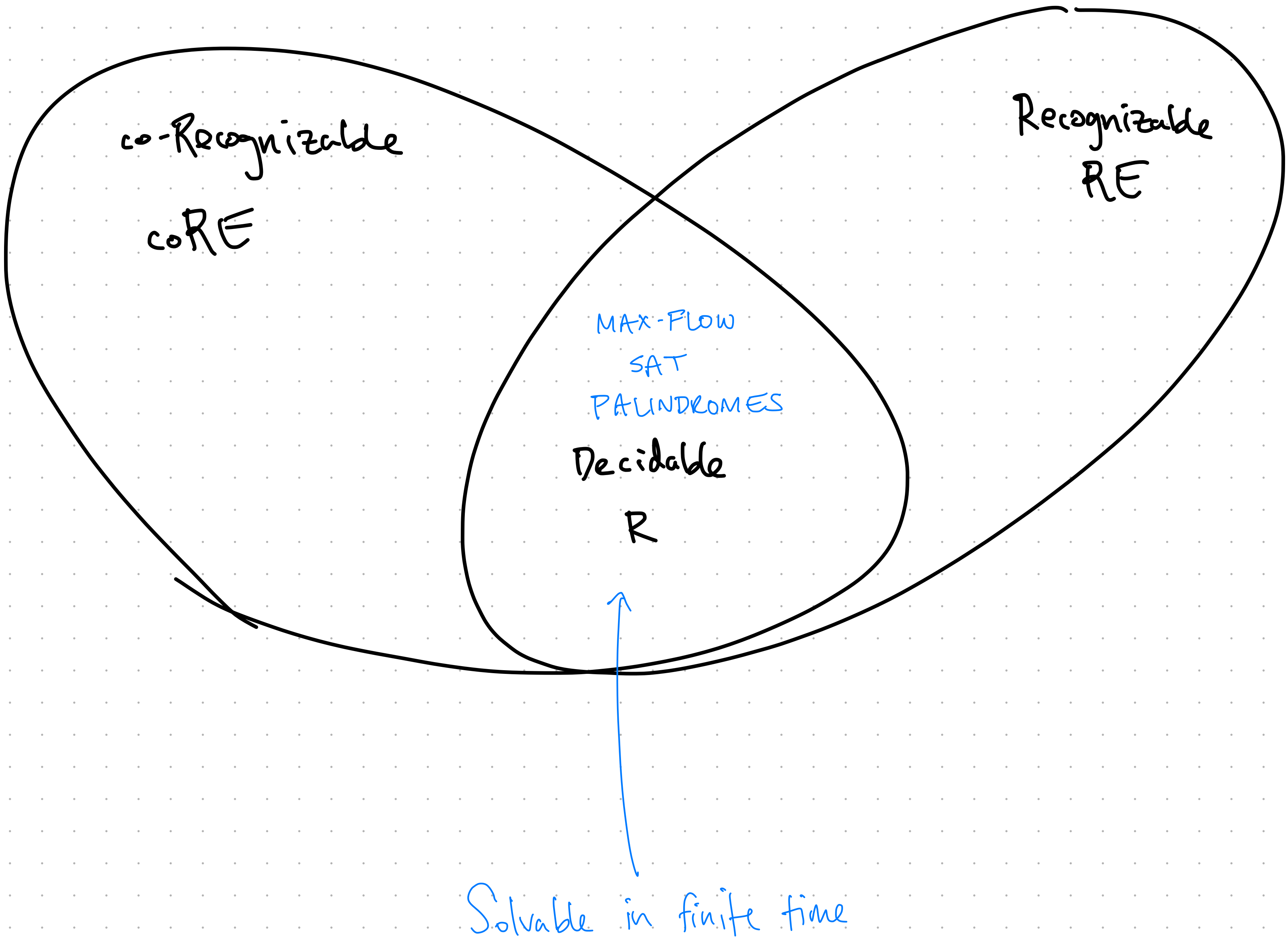
22 April 2024 Program Analysis

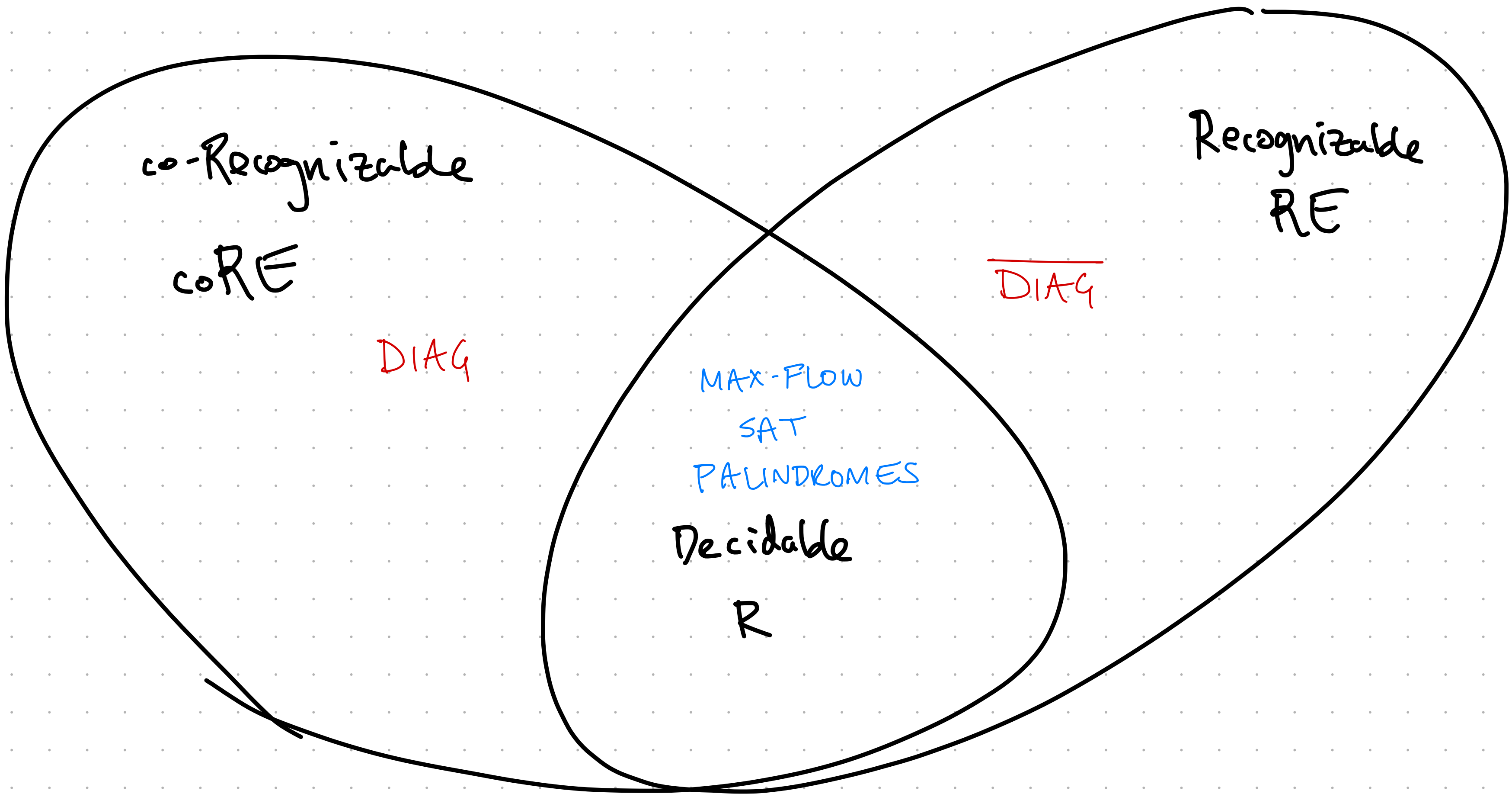
Plan

* Program Equivalence
  - ↳ EQ ∉ RE
  - ↳ EQ ∉ coRE

* Announcements

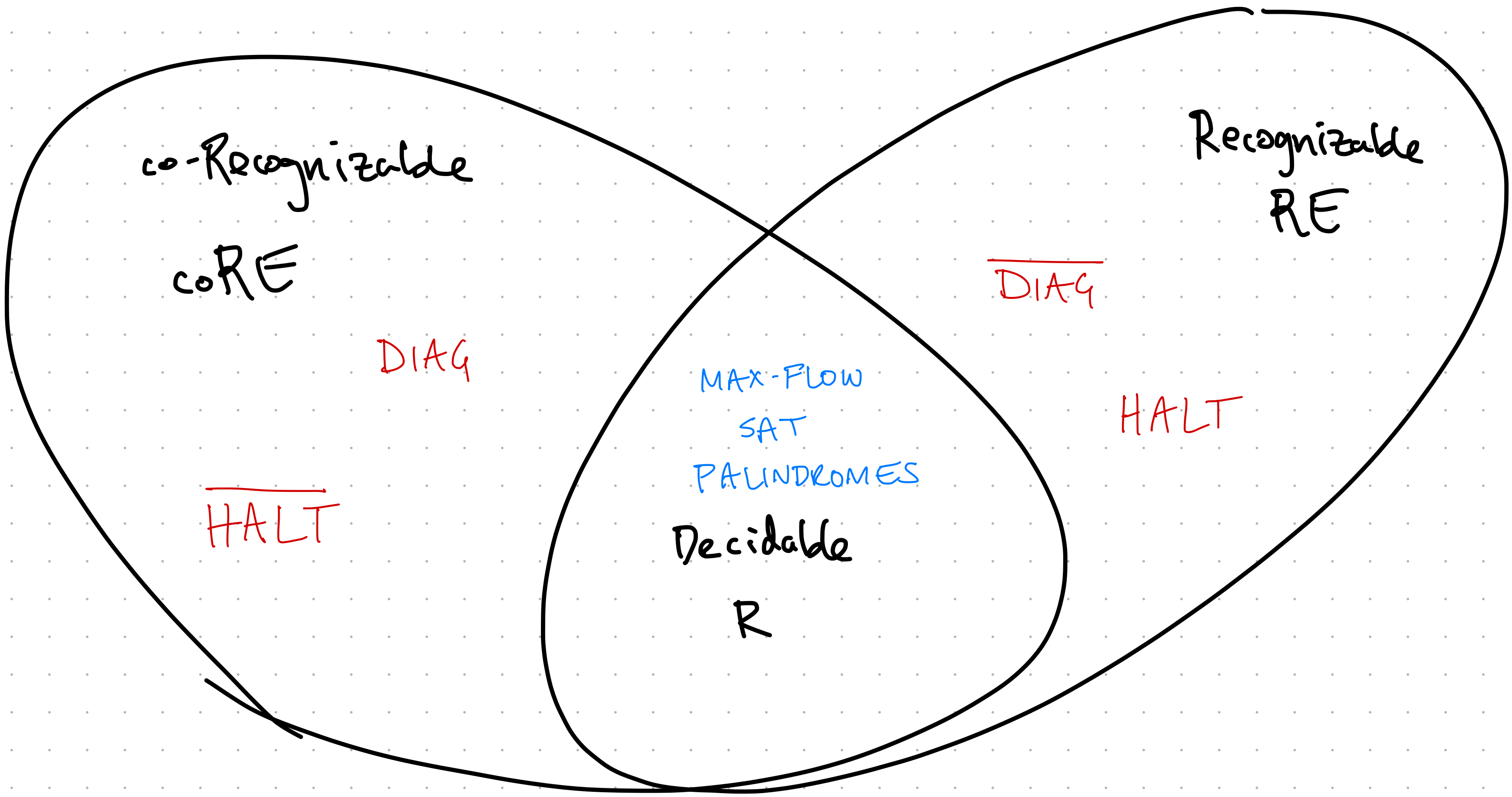
* Rice's Theorem


co-Recognizable
coRE
Recognizable
RE
MAX-FLOW
SAT
PALINDROMES
Decidable
R
Solvable in finite time



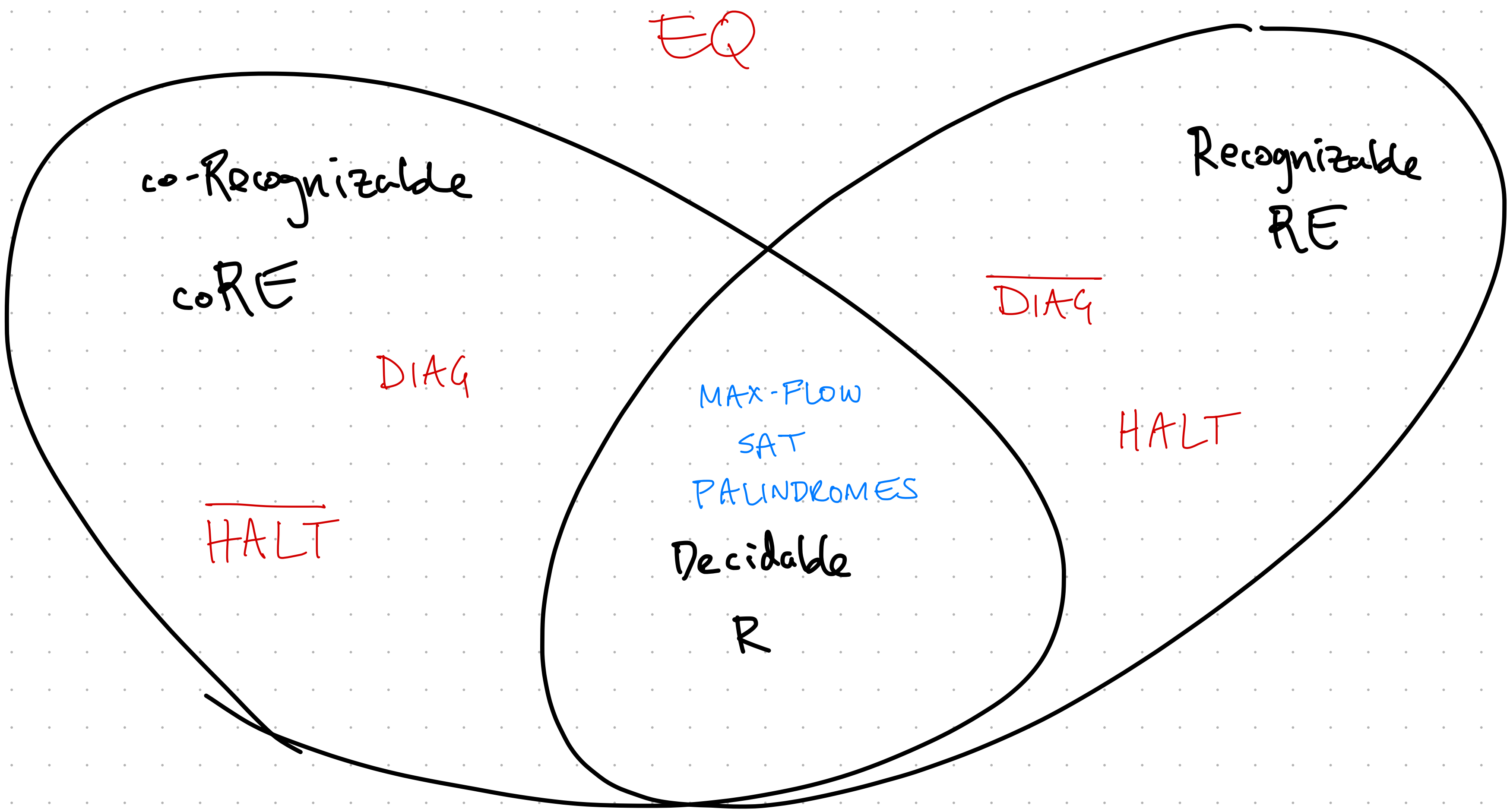
$$\text{DIAG} = \{ \langle M \rangle : M \text{ does not accept } \langle M \rangle \}$$

$$\overline{\text{DIAG}} = \{ \langle M \rangle : M \text{ accepts } \langle M \rangle \}$$



$$\text{HALT} = \{ \langle M \rangle \# \langle x \rangle : M \text{ halts on input } x \}$$

$$\overline{\text{HALT}} = \{ \langle M \rangle \# \langle x \rangle : M \text{ does not halt on input } x \}$$



$$EQ = \{ \langle M_1 \rangle \# \langle M_2 \rangle : \mathcal{L}(M_1) = \mathcal{L}(M_2) \}$$

Given two TMs, do they recognize the same language?

$$EQ = \{ \langle M_1 \rangle \# \langle M_2 \rangle : \mathcal{L}(M_1) = \mathcal{L}(M_2) \}$$

Given two TMs, do they recognize the same language?

<u>Theorem</u>. $EQ \notin RE \cup coRE$.

↳ Not Recognizable, nor coRecognizable!

$$EQ = \{ \langle M_1 \rangle \# \langle M_2 \rangle : \mathcal{L}(M_1) = \mathcal{L}(M_2) \}$$

Given two TMs, do they recognize the same language?

<u>Theorem</u>. $EQ \notin RE \cup coRE$.

↳ Not Recognizable, nor coRecognizable!

Proof Approach: Reduction from the Halting Problem.

Recall. HALT $\notin$ coRE

↳ Determining if M halts on input x is recognizable, but not decidable.

<u>Recall</u>. $\text{HALT} \notin \text{coRE}$

↳ Determining if M halts on input x is recognizable, but not decidable.

$$\text{HALT} \leq \text{EQ} \implies \text{EQ} \notin \text{coRE}.$$

Recall. HALT $\notin$ coRE

↳ Determining if M halts on input x is recognizable, but not decidable.

$$\text{HALT} \underbrace{\leq}_{\text{Computable Reduction } R} \text{EQ} \implies \text{EQ} \notin \text{coRE}.$$

$$\langle M \rangle \# \langle x \rangle \xrightarrow{\quad R \quad} \langle M_1 \rangle \# \langle M_2 \rangle$$

if M halts on input x then $\mathcal{L}(M_1) = \mathcal{L}(M_2)$

if M does not halt on input x then $\mathcal{L}(M_1) \neq \mathcal{L}(M_2)$

Reduction from HALT to EQ.

On input $\langle M \rangle \# \langle x \rangle$.

construct

$\underline{M_1}$

on input w

$\underline{M_2}$

on input w

Output $\langle M_1 \rangle \# \langle M_2 \rangle$

Reduction from HALT to EQ.

On input $\langle M \rangle \# \langle x \rangle$.

construct

$\underline{M_1}$

on input w

Accept

$\underline{M_2}$

on input w

Output $\langle M_1 \rangle \# \langle M_2 \rangle$

Reduction from HALT to EQ.

On input $\langle M \rangle \# \langle x \rangle$.

Construct

$\underline{M_1}$

on input w

Accept

$\underline{M_2}$

on input w

Execute M running on x

Accept

// Hard code M and x into finite state controller of M_2

Output $\langle M_1 \rangle \# \langle M_2 \rangle$

Reduction from HALT to EQ.

On input $\langle M \rangle \# \langle x \rangle$.

Construct

$\underline{M_1}$

on input w

Accept

$\mathcal{L}(M_1) = \Sigma^*$

$\underline{M_2}$

on input w

Execute M running on x

Accept

$\mathcal{L}(M_2) = ?$

Output $\langle M_1 \rangle \# \langle M_2 \rangle$

$\mathcal{L}(M_1) = \Sigma^*$

$\mathcal{L}(M_2) = ?$

$\underline{M_2}$

on input w

Execute M running on x

Accept

Suppose

$\langle M \rangle \# \langle x \rangle \in HALT$. $\Rightarrow$ M halts on input x

$\Rightarrow$ M_2 Accepts on input w, $\forall w \in \Sigma^*$

$\Rightarrow$ $\mathcal{L}(M_2) = \Sigma^*$

$\mathcal{L}(M_1) = \Sigma^*$

$\mathcal{L}(M_2) = ?$

$\boxed{\begin{array}{l} \underline{M_2} \\ \quad \text{on input } w \\ \quad \text{Execute } M \text{ running on } x \\ \quad \text{Accept} \end{array}}$

Suppose

$\langle M \rangle \# \langle x \rangle \in HALT.$ $\Rightarrow$ M halts on input x

$\Rightarrow$ M_2 Accepts on input w, $\forall w \in \Sigma^*$

$\Rightarrow$ $\mathcal{L}(M_2) = \Sigma^*$

Suppose

$\langle M \rangle \# \langle x \rangle \notin HALT.$ $\Rightarrow$ M does NOT halt on input x

$\Rightarrow$ M_2 does NOT halt on input w, $\forall w \in \Sigma^*$

$\Rightarrow$ $\mathcal{L}(M_2) = \phi$

$$\langle M \rangle \# \langle x \rangle \xrightarrow{R} \langle M_1 \rangle \# \langle M_2 \rangle$$

s.t.

$$\mathcal{L}(M_1) = \mathcal{L}(M_2) \iff M \text{ halts on input } x.$$

$$\Rightarrow \quad HALT \leq EQ. \qquad \Rightarrow \quad EQ \notin coRE$$

Recall: $\overline{HALT} \notin RE$.

So $\overline{HALT} \leq EQ \implies EQ \notin RE$

Computable Reduction: R

$$\langle M \rangle \# \langle x \rangle \xrightarrow{\quad R \quad} \langle M_1 \rangle \# \langle M_2 \rangle$$

if M does NOT halt on input x then $\mathcal{L}(M_1) = \mathcal{L}(M_2)$

if M halts on input x then $\mathcal{L}(M_1) \neq \mathcal{L}(M_2)$

Reduction from HALT to EQ.

On input $\langle M \rangle \# \langle x \rangle$.

construct

$\underline{M_1}$

on input w

Accept

$\underline{M_2}$

on input w

Execute M running on x

Reject

Output $\langle M_1 \rangle \# \langle M_2 \rangle$

Great idea!

But...

wrong.

$\langle M \rangle \# \langle x \rangle \in HALT \implies \mathcal{L}(M_2) = \emptyset$

$\langle M \rangle \# \langle x \rangle \notin HALT \implies \mathcal{L}(M_2) = \emptyset$

Reduction from HALT to EQ.

On input $\langle M \rangle \# \langle x \rangle$.

construct

$\underline{M_1}$

on input w

Reject.

$\mathcal{L}(M_1) = \emptyset$

$\underline{M_2}$

on input w

Execute M running on x

Accept

$\mathcal{L}(M_2) = ?$

Output $\langle M_1 \rangle \# \langle M_2 \rangle$

$\mathcal{L}(M_1) = \emptyset$

$\mathcal{L}(M_2) = ?$

$\underline{M_2}$

on input w

Execute M running on x

Accept

Suppose

$\langle M \rangle \# \langle x \rangle \in \overline{HALT}$. $\Longrightarrow$ M does <u>not</u> halt on x

$\Rightarrow \mathcal{L}(M_2) = \emptyset$

Suppose

$\langle M \rangle \# \langle x \rangle \notin \overline{HALT}$ $\Longrightarrow$ M halts on x

$\Rightarrow \mathcal{L}(M_2) = \Sigma^*$

$\mathcal{L}(M_1) = \mathcal{L}(M_2) \iff M$ does <u>not</u> halt on input x.

$$EQ = \{ \langle M_1 \rangle \# \langle M_2 \rangle : \mathcal{L}(M_1) = \mathcal{L}(M_2) \}$$

$$\boxed{EQ \notin RE \cup coRE}$$

Determining whether two programs have the same functionality is the hardest problem we've seen!

Announcements

* HW 8 out, due Thurs 11:59 pm.
  - ↳ Q1: No formal proof required
    Explain your TM design & why significant steps work.
    Analogy to well-documented code

The Check-GPT Problem.

* Write an inefficient algorithm A for 4820 homework
* Ask GPT to return an efficient algorithm A^* that solves the same problem as A.
* Return True iff A and A^* solve the same problem.

Theorem. Check-GPT is Undecidable!

Namely, there is no algorithm (current or future) that can reliably check the output of AI for correctness.

Languages about Turing Machines

* Examples

  DIAG, HALT, EQ, ...

* Many such languages are undecidable.

Do we have to show a new reduction for each one?

Languages about languages

* A TM Property is a language of TM descriptions

e.g. $P_{100} = \{ \langle M \rangle : Q(M) \text{ has } 100 \text{ states} \}$

Languages about languages

* A TM Property is a language of TM descriptions

e.g. $P_{100} = \{ \langle M \rangle : Q(M) \text{ has } 100 \text{ states} \}$

* A TM property is semantic if membership $\langle M \rangle \in P$ is determined by the language $\mathcal{L}(M)$

$$P_{empty} = \{ \langle M \rangle : \mathcal{L}(M) = \emptyset \}$$

$$P_{finite} = \{ \langle M \rangle : M \text{ accepts a finite collection of strings} \}$$

$$P_w = \{ \langle M \rangle : w \in \mathcal{L}(M) \}$$

Languages about languages

* A TM Property is a language of TM descriptions

e.g. $P_{100} = \{ \langle M \rangle : Q(M) \text{ has } 100 \text{ states} \}$

* A TM property is semantic if membership $\langle M \rangle \in P$ is determined by the language $\mathcal{L}(M)$

$\forall M_1, M_2$ s.t. $\mathcal{L}(M_1) = \mathcal{L}(M_2)$

M_1 satisfies $P \iff M_2$ satisfies P

Languages about languages

* A TM Property is a language of TM descriptions

e.g. $P_{100} = \{ \langle M \rangle : Q(M) \text{ has } 100 \text{ states} \}$

* A TM property is semantic if membership $\langle M \rangle \in P$ is determined by the language $\mathcal{L}(M)$

$\forall M_1, M_2$ s.t. $\mathcal{L}(M_1) = \mathcal{L}(M_2)$

M_1 satisfies $P \iff M_2$ satisfies P

* A TM Property is nontrivial if $P \notin \{ \emptyset, \Sigma^* \}$

Rice's Theorem

Every nontrivial, semantic TM property is Undecidable!

Proof Idea Reduce from Halting Problem!

Pf Suppose P is a nontrivial, semantic property.

* Consider the TM M_ϕ = "on input w, Reject".

Assume $\langle M_\phi \rangle \notin P$.

(Similar argument if $\langle M_\phi \rangle \in P$)

Pf Suppose P is a nontrivial, semantic property.

* Consider the TM $M_\emptyset$ = "on input w, Reject".

Assume $\langle M_\emptyset \rangle \notin P$.

* There exists some TM K s.t. $\langle K \rangle \in P$.

Pf Suppose P is a nontrivial, semantic property.

* Consider the TM M_ϕ = "on input w, Reject".

Assume $\langle M_\phi \rangle \notin P$.

* There exists some TM K s.t. $\langle K \rangle \in P$.

Reduction from $HALT \leq_{TM} P$.

Given $\langle M \rangle \# \langle x \rangle$ Construct

M_P

On input w.

Execute M running on x.

Run K on w.

if K accepts, Accept.

output $\rightarrow \langle M_P \rangle$

Pf Suppose P is a nontrivial, semantic property.

* Consider the TM M_ϕ = "on input w, Reject".

Assume $\langle M_\phi \rangle \notin P$.

* There exists some TM K s.t. $\langle K \rangle \in P$.

Reduction from HALT $\leq_{TM}$ P.

Given $\langle M \rangle \# \langle x \rangle$ Construct

M_P

On input w.

Execute M running on x.

Run K on w.

if K accepts, Accept.

output $\langle M_P \rangle$

semantic

$\mathcal{L}(M_P) = \mathcal{L}(M_\phi) = \phi \notin P$ if M does NOT halt on x

$\mathcal{L}(M_P) = \mathcal{L}(K) \in P$ if M halts on x.

Deciding any nontrivial, semantic property allows us to decide the halting problem!

Rice's Theorem

$\Downarrow$

Cannot generically verify correctness/security of arbitrary programs!