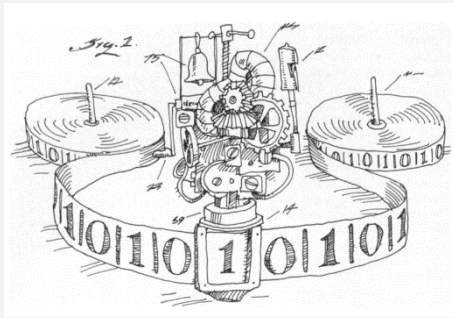
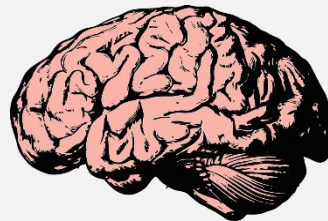
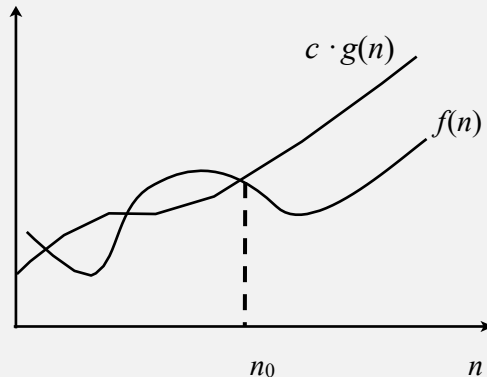


COMP 3602

Intro to Theory of Computation

Lecture 12



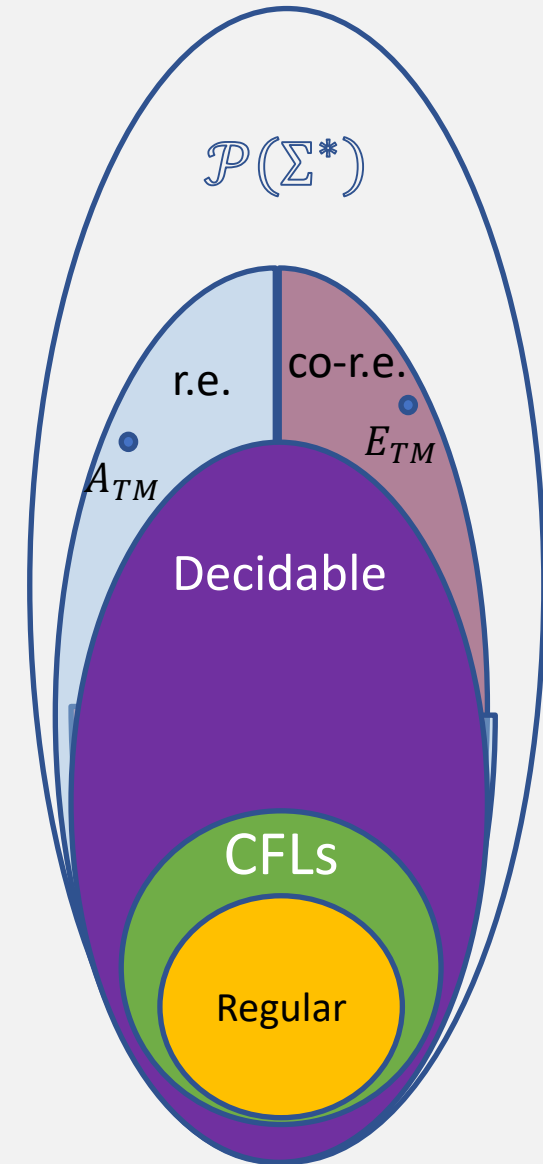
Logistics

- Classes back to 50 minutes starting next week?
 - Except Fridays still till 4-5:15
- First oral exam: as soon as we finish the next topic
 - Which days of the week and times are OK? Mon Mar 2nd?
 - Practice problems posted (incl. today's lecture)
- Presentation proposals? To say which topic you picked. Mar 2nd?
 - About 30 (50?) minutes intro to a topic in complexity (or computability)
 - In the reading group on in class?
 - Like Jordan's intro to Kolmogorov complexity, but shorter
 - Quantum computing, proof complexity, fine-grained complexity, circuit complexity, communication complexity, pseudorandomness, etc...

Questions/comments?

Reduction examples: E_{TM}

- $E_{TM} = \{\langle M \rangle \mid \mathcal{L}(M) = \emptyset\}$
 - Set of machines M that do not accept anything
- *Claim:* E_{TM} is co-Turing-recognizable and undecidable.
$$\overline{E_{TM}} = \{x \mid x \neq \langle M \rangle\} \cup \{\langle M \rangle \mid M \text{ accepts some string } w\}$$
- E_{TM} is co-Turing-recognizable
 - Show that $\overline{E_{TM}}$ is Turing-recognizable
- E_{TM} is undecidable.
 - Show that $\overline{A_{TM}} \leq_m E_{TM}$, or, equivalently, $A_{TM} \leq_m \overline{E_{TM}}$
 - Remember that **there cannot be a reduction** $A_{TM} \leq_m E_{TM}$!



- $E_{TM} = \{\langle M \rangle \mid M \text{ does not accept anything, that is, } \mathcal{L}(M) = \emptyset\}$
- $\overline{E_{TM}} = \{x \mid x \neq \langle M \rangle\} \cup \{\langle M \rangle \mid M \text{ accepts some string } w\}$

- E_{TM} is co-Turing-recognizable

- Equivalently, $\overline{E_{TM}}$ is Turing-recognizable, and this is what we will show

Idea: $M_{\overline{E_{TM}}}$ will work in stages, on each stage trying a few more strings, and doing one more step on the previously considered ones.

$M_{\overline{E_{TM}}} =$ “ On input x

If $x \neq \langle M \rangle$, *accept*

For $i = 0$ to ∞

 Simulate M on all strings of length at most i for i steps each

 If M enters accept state on any of these strings, *accept*.

Decidable to check if x is a valid encoding of a TM

- Note: could also take strings in, say, string order (so that $w_0 = \lambda, w_1 = 0, w_2 = 1, w_3 = 00, \dots$) and simulate M on inputs $w_0 \dots w_i$ for i steps each at each stage.

- As long as each stage is guaranteed to finish in finite time,

- If there is w which M accepts, say in t steps, then $M_{\overline{E_{TM}}}$ accepts $\langle M \rangle$ during stage $\max(|w|, t)$
- If $\mathcal{L}(M) = \emptyset$, $M_{\overline{E_{TM}}}$ will run forever and never accept.

- $E_{TM} = \{\langle M \rangle \mid M \text{ does not accept anything, that is, } \mathcal{L}(M) = \emptyset\}$
- $\overline{E_{TM}} = \{x \mid x \neq \langle M \rangle\} \cup \{\langle M \rangle \mid M \text{ accepts some string } w\}$

• E_{TM} is undecidable.

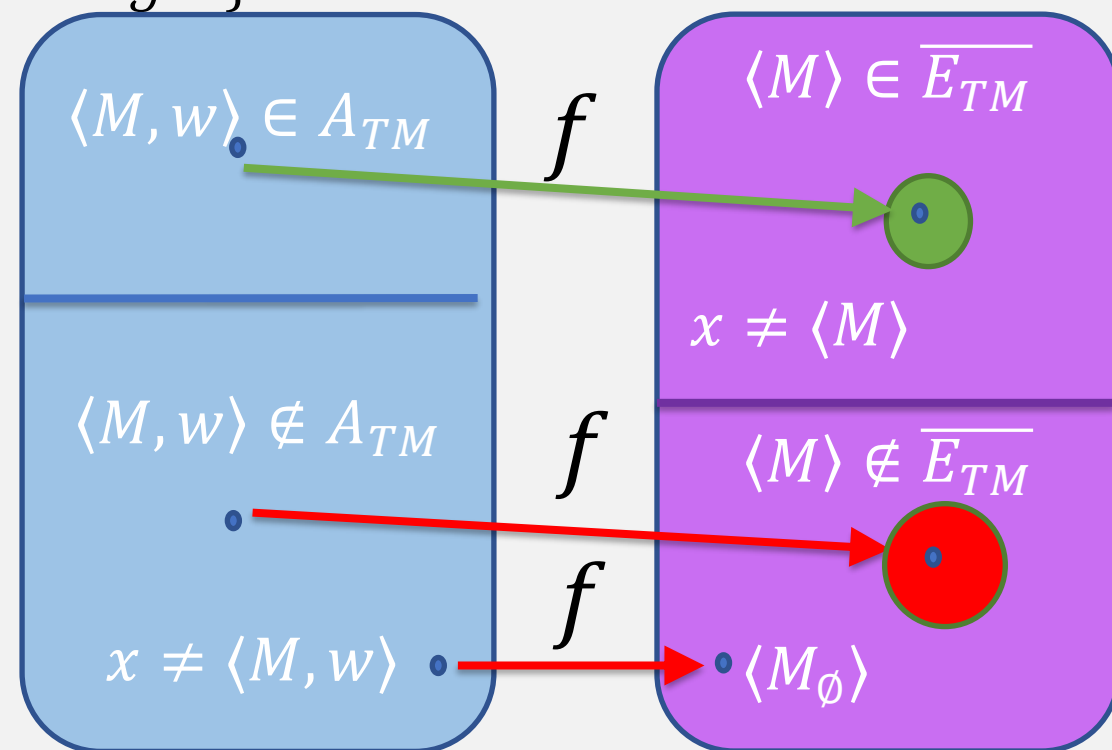
- Let's show that $A_{TM} \leq_m \overline{E_{TM}}$
- Construct a computable function f

$$f(x) = \begin{cases} \langle M' \rangle & \text{if } x = \langle M, w \rangle \\ \langle M_\emptyset \rangle & \text{otherwise} \end{cases}$$

- Where M' is:



M' = “ On input x
 Simulate M on input w
 If M ever enters accept state, *accept*.
 If M ever enters reject state, *reject*. ”



$M_\emptyset$: $\Sigma = \{0,1\}, \Gamma = \{0,1,_ \}$

$Q = \{q_0, q_a, q_r\}$

$\delta: (q_0, s) \rightarrow (q_r, s, R) \text{ for all } s \in \Gamma$

- $E_{TM} = \{\langle M \rangle \mid M \text{ does not accept anything, that is, } \mathcal{L}(M) = \emptyset\}$
- $\overline{E_{TM}} = \{x \mid x \neq \langle M \rangle\} \cup \{\langle M \rangle \mid M \text{ accepts some string } w\}$

• E_{TM} is undecidable.

– Let's show that $\text{Halt}_B \leq_m \overline{E_{TM}}$

– Construct a computable function f

$$f(x) = \begin{cases} \langle M' \rangle & \text{if } x = \langle M \rangle \\ \langle M_\emptyset \rangle & \text{otherwise} \end{cases}$$

– Where M' is:

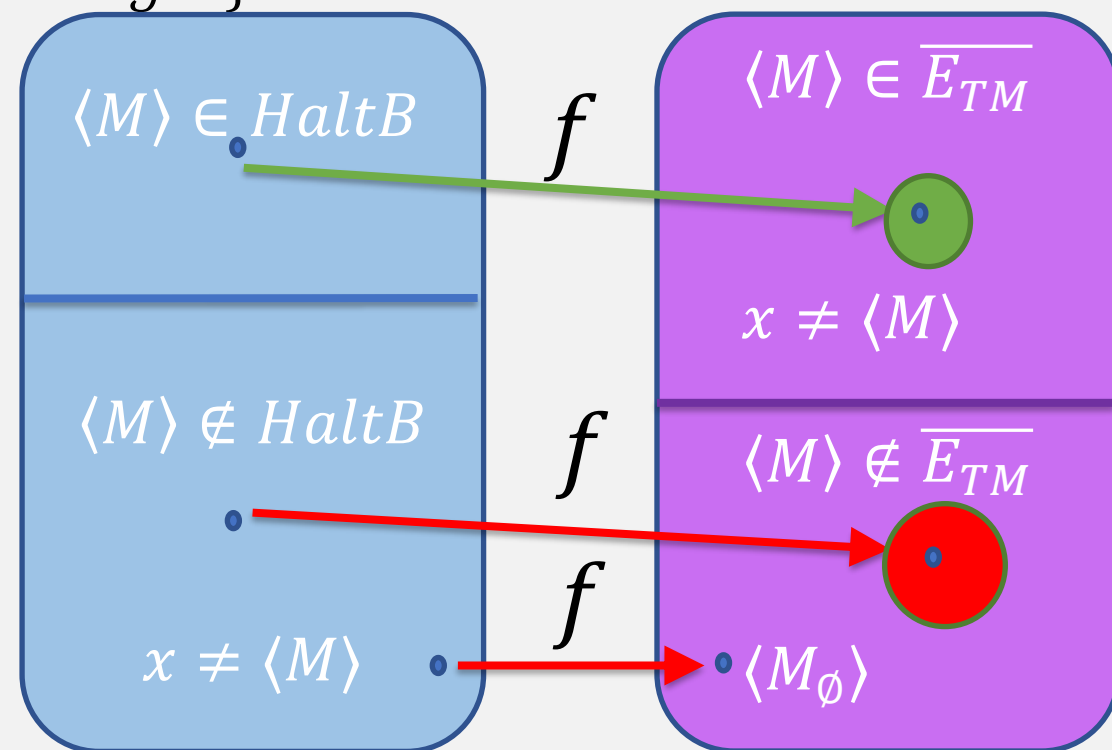


M' = “ On input x

Simulate M on input λ

If M ever enters accept state, *accept*.

If M ever enters reject state, *accept*. ”



$M_\emptyset: \Sigma = \{0,1\}, \Gamma = \{0,1,_ \}$

$Q = \{q_0, q_a, q_r\}$

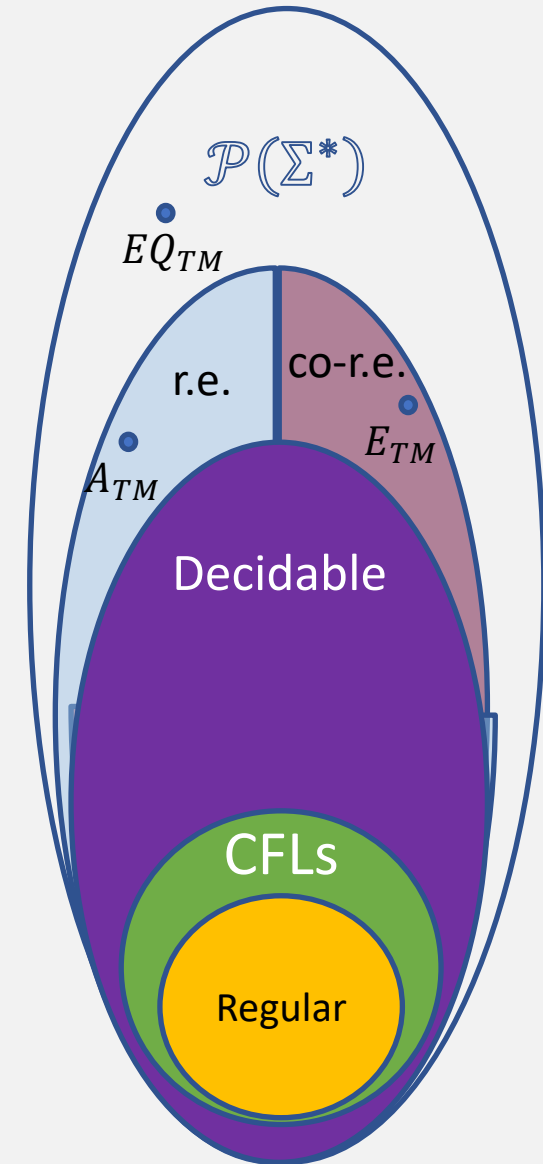
$\delta: (q_0, s) \rightarrow (q_r, s, R) \text{ for all } s \in \Gamma$

Reduction examples: EQ_{TM}

- $EQ_{TM} = \{\langle M_1, M_2 \rangle \mid \mathcal{L}(M_1) = \mathcal{L}(M_2)\}$
 - Set of pairs of machines accepting the same language.

Claim: EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable

- EQ_{TM} is not co-Turing-recognizable
 - Show that $A_{TM} \leq_m EQ_{TM}$
- EQ_{TM} is not Turing-recognizable.
 - Show that $\overline{A_{TM}} \leq_m EQ_{TM}$, or, equivalently, $A_{TM} \leq_m \overline{EQ_{TM}}$



- $EQ_{TM} = \{\langle M_1, M_2 \rangle \mid \mathcal{L}(M_1) = \mathcal{L}(M_2)\}$
- $\overline{EQ_{TM}} = \{x \mid x \neq \langle M_1, M_2 \rangle\} \cup \{\langle M_1, M_2 \rangle \mid \mathcal{L}(M_1) \neq \mathcal{L}(M_2)\}$

- EQ_{TM} is not Turing-recognizable

- Let's show that $A_{TM} \leq_m \overline{EQ_{TM}}$
- Construct a computable function f

$$f(x) = \begin{cases} \langle M', M_\emptyset \rangle & \text{if } x = \langle M, w \rangle \\ \langle M_\emptyset, M_\emptyset \rangle & \text{otherwise} \end{cases}$$

- Where M' is:

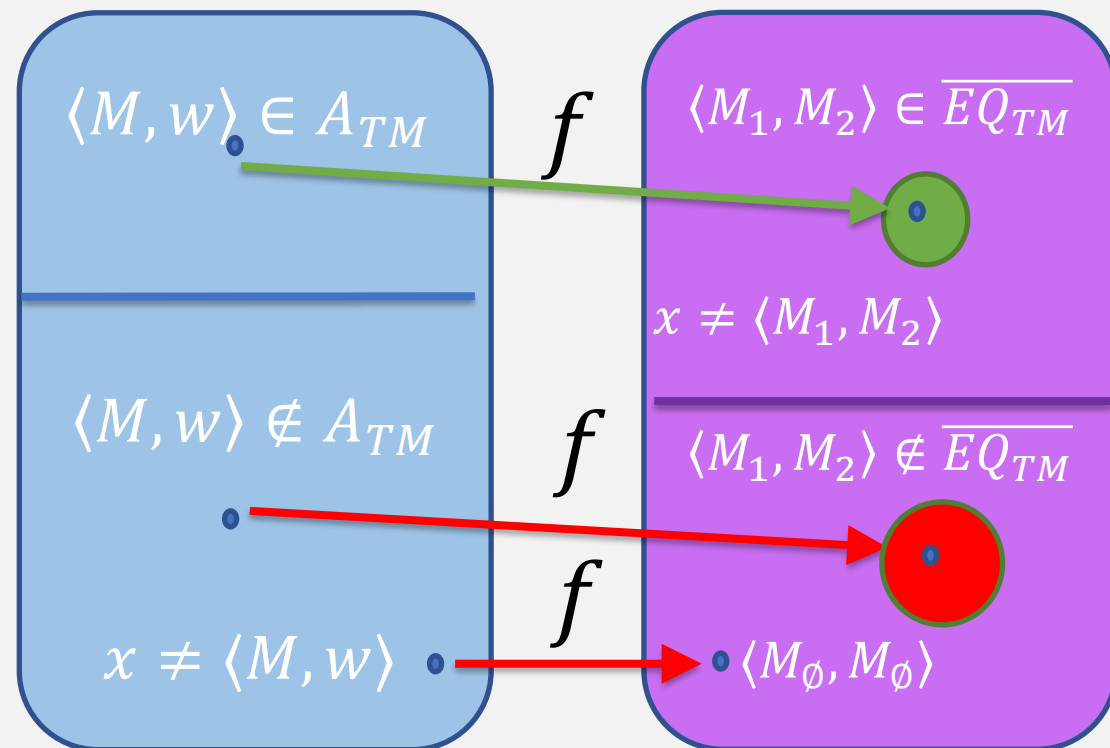


M' = “ On input x

Simulate M on input w

If M ever enters accept state, *accept*.

If M ever enters reject state, *reject*. ”



$M_\emptyset$: $\Sigma = \{0,1\}, \Gamma = \{0,1, _Q = \{q_0, q_a, q_r\}$
 $\delta: (q_0, s) \rightarrow (q_r, s, R)$ for all $s \in \Gamma$

- $EQ_{TM} = \{\langle M_1, M_2 \rangle \mid \mathcal{L}(M_1) = \mathcal{L}(M_2)\}$
- $\overline{EQ_{TM}} = \{x \mid x \neq \langle M_1, M_2 \rangle\} \cup \{\langle M_1, M_2 \rangle \mid \mathcal{L}(M_1) \neq \mathcal{L}(M_2)\}$

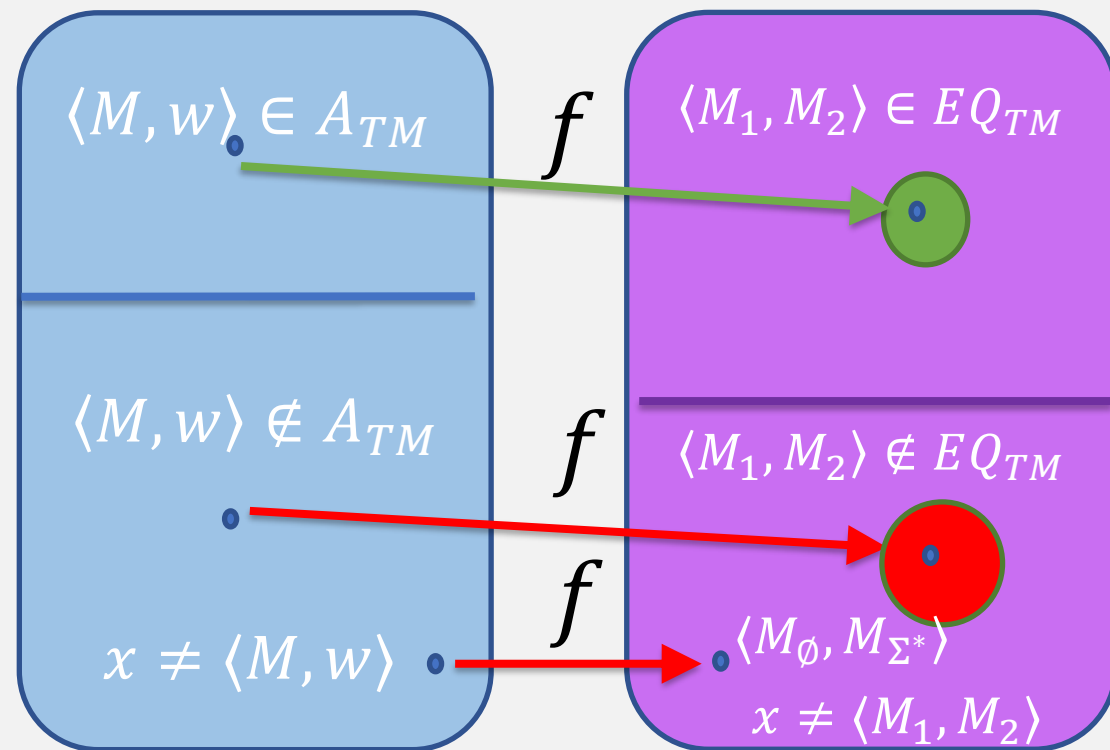
- EQ_{TM} is not **co-Turing-recognizable**

- Let's show that $A_{TM} \leq_m EQ_{TM}$
 - Construct a computable function f
- $$f(x) = \begin{cases} \langle M', M_{\Sigma^*} \rangle & \text{if } x = \langle M, w \rangle \\ \langle M_{\emptyset}, M_{\Sigma^*} \rangle & \text{otherwise} \end{cases}$$

- Where M' is:



M' = “ On input x
 Simulate M on input w
 If M ever enters accept state, **accept**.
 If M ever enters reject state, **reject**. ”



$M_{\emptyset}$: $\Sigma = \{0,1\}, \Gamma = \{0,1, _Q = \{q_0, q_a, q_r\}$
 $\delta: (q_0, s) \rightarrow (q_r, s, R)$ for all $s \in \Gamma$

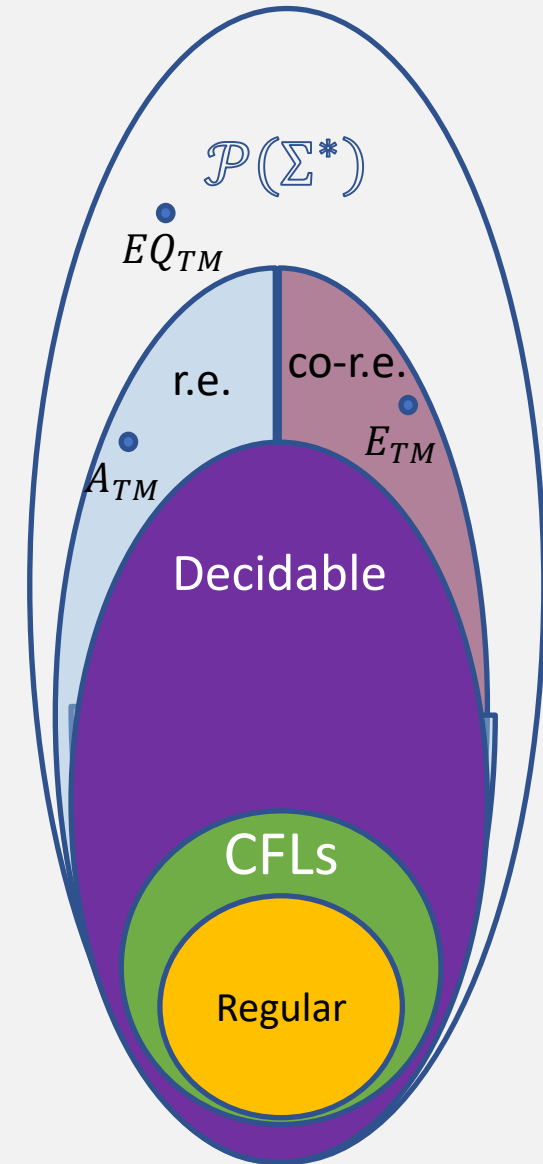
M_{Σ^*} : $\Sigma = \{0,1\}, \Gamma = \{0,1, _Q = \{q_0, q_a, q_r\}$
 $\delta: (q_0, s) \rightarrow (q_a, s, R)$ for all $s \in \Gamma$

Reduction examples: EQ_{TM}

- $EQ_{TM} = \{\langle M_1, M_2 \rangle \mid \mathcal{L}(M_1) = \mathcal{L}(M_2)\}$
 - Pairs of machines accepting the same language.

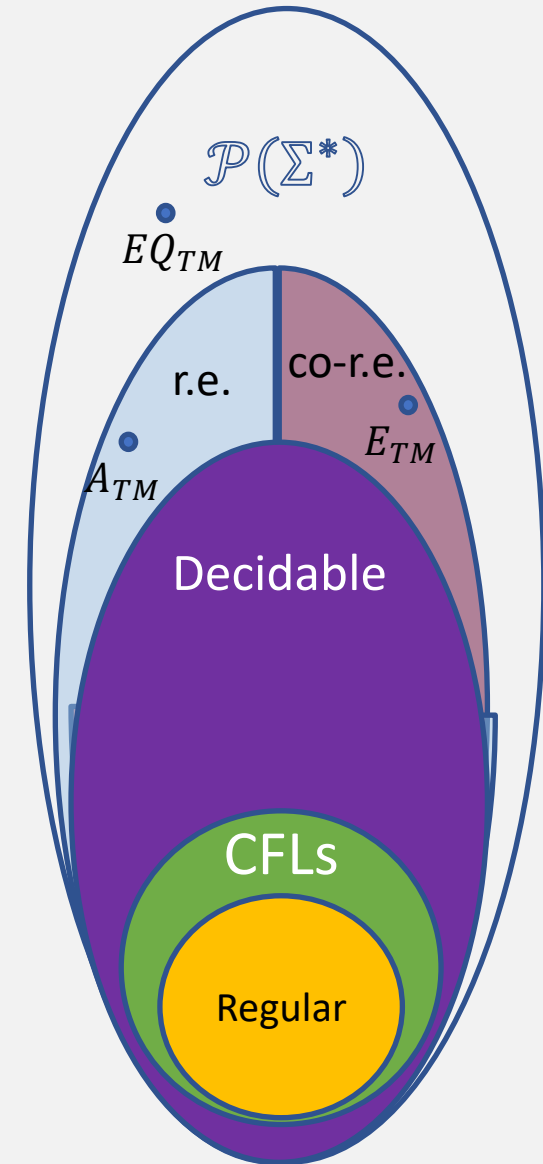
Claim: EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable

- EQ_{TM} is not co-Turing-recognizable
 - Show that $A_{TM} \leq_m EQ_{TM}$
- EQ_{TM} is not Turing-recognizable.
 - Show that $\overline{A_{TM}} \leq_m EQ_{TM}$, or, equivalently, $A_{TM} \leq_m \overline{EQ_{TM}}$



Is it possible to guess from the language description where in the picture it is likely to be?

- Yes!
 - The quantifiers used in the description of a language tell us if it is likely to be r.e., co-r.e., etc
 - And they let us define a whole hierarchy of language classes generalizing r.e. and co-r.e., called *arithmetic hierarchy*





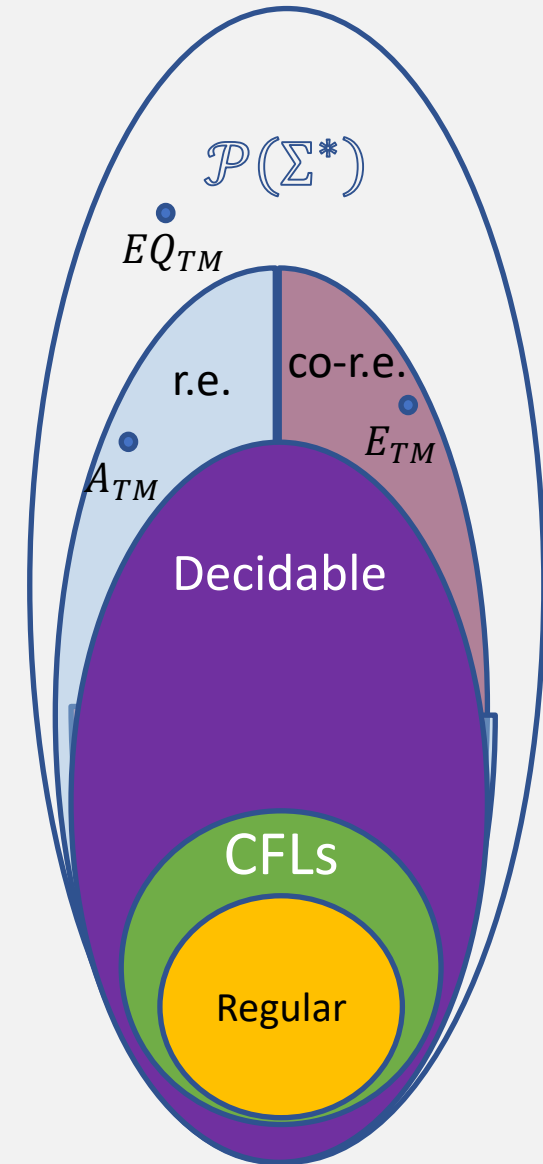
Defining languages in terms of computations

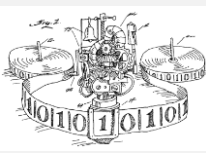
- Recall: a *computation* of a TM is a sequence of *configurations* $C = (C_0, C_1, C_2, \dots)$
- If the last configuration has q_a/q_r , call it accepting/rejecting computation.
 - If a computation is either accepting or rejecting, it is halting.
- A computation is finite (can be encoded as a finite string) iff it is halting (accepting or rejecting).
 - If a TM M accepts w , this can be stated as “ $\exists C$ such that C is an accepting computation of M on w ”
 - If a TM M does not accept w , then “ $\forall C$, C is not an accepting computation of M on w ”.
- Let predicate $Check_A(M, w, y)$ be true iff y encodes an accepting computation of M on w .
 - $Check_R(M, w, y), Check_H(M, w, y)$ for checking for a rejecting/halting computation respectively
 - Now, $A_{TM} = \{ \langle M, w \rangle \mid \exists y Check_A(M, w, y) \}$
 - Call a predicate $R(x_1 \dots x_k)$ decidable iff $\{ (x_1, \dots, x_k) \in R \}$ is decidable.
- *Claim:* $Check_A(M, w, y), Check_R(M, w, y), Check_H(M, w, y)$ are decidable predicates.



Examples

- $A_{TM} = \{ \langle M, w \rangle \mid \exists y \text{ Check}_A(M, w, y) \}$
- $\text{HaltB} = \{ \langle M \rangle \mid M \text{ halts on } \lambda \}$
 - $\{ \langle M \rangle \mid \exists y \text{ Check}_H(M, \lambda, y) \}$
- $E_{TM} = \{ \langle M \rangle \mid \mathcal{L}(M) = \emptyset \}$
 - $\{ \langle M \rangle \mid \forall w \forall y \neg \text{Check}_A(M, w, y) \}$
 - $\overline{E_{TM}} = \{ x \mid x \neq \langle M \rangle \} \cup \{ \langle M \rangle \mid M \text{ accepts some string } w \}$
 - $\{ x \mid x \neq \langle M \rangle \} \cup \{ \langle M \rangle \mid \exists (w, y) \text{Check}_A(M, w, y) \}$
 - Recall our $M_{\overline{E_{TM}}}$ iterating over pairs (w, y) !
- $EQ_{TM} = \{ \langle M_1, M_2 \rangle \mid \mathcal{L}(M_1) = \mathcal{L}(M_2) \}$
 - $\{ \langle M_1, M_2 \rangle \mid \forall w (\exists y_1 \text{Check}_A(M_1, w, y_1)) \leftrightarrow (\exists y_2 \text{Check}_A(M_2, w, y_2)) \}$
 - $$= \{ \langle M_1, M_2 \rangle \mid \forall w \left(\begin{array}{c} \forall z_1 \forall z_2 \neg \text{Check}_A(M_1, w, z_1) \wedge \neg \text{Check}_A(M_2, w, z_2) \\ \vee \\ \exists y_1 \exists y_2 \text{Check}_A(M_1, w, y_1) \wedge \text{Check}_A(M_2, w, y_2) \end{array} \right) \}$$
- $ALL_{TM} = \{ \langle M \rangle \mid M \text{ accepts everything} \}$





Quantifiers and language classes

- *Claim:* A language is of the form $L = \{x \mid \exists y R(x, y)\}$ where R is a decidable predicate iff L is Turing-recognizable.
 - $\Rightarrow$ Design a TM M_L which checks $R(x, y)$ on each y in some order
 - If $R(x, y)$ is true for some y , accept. Otherwise M_L runs forever.
 - $\Leftarrow$ Suppose L is Turing-recognizable. Then there exists some machine M_L recognizing L . Then $L = \{x \mid \exists y \text{Check}_A(M_L, x, y)\}$
- *Corollary:* A language is of the form $L = \{x \mid \forall y R(x, y)\}$ where R is a decidable predicate iff L is co-Turing-recognizable.
 - Note that here we are only showing “easiness”: these claims say nothing about undecidability
- We are quantifying over potentially *infinitely* many finite things.
 - If can quantify over *finitely* many finite things, that is a decidable language.
 - For example, $A_{TM}^b = \{\langle M, w, t \rangle \mid M \text{ accepts } w \text{ in } t \text{ steps}\}$ is decidable.



Arithmetic hierarchy

- Idea: a level for each # of quantifier alternations
 - Bottom (0^{th}) level: decidable
 - 1^{st} level: $\exists R, \forall R$ where R is decidable.
 - 2^{nd} level: $\exists \forall R, \forall \exists R$ where R is decidable.
 - Equivalently, $\exists \varphi, \forall \varphi$ where φ from 1^{st} level with opposite quantifier
 - ... i^{th} level: $\exists \varphi, \forall \varphi$ where φ is from level $i-1$
 - With opposite quantifiers
- Each subsequent layer contains all the previous ones.
 - It is possible to prove that each contains languages not appearing below.
- Some languages can be written both as $\exists x_1 \varphi_1(x_1)$ and $\forall x_2 \varphi_2(x_2)$ where $\varphi_1 = \forall \dots$, $\varphi_2 = \exists \dots$ from the previous level.
 - Eg: $L = \{\langle M \rangle \mid M \text{ accepts } 0 \text{ and loops on } 1\} =$
 $= \{\langle M \rangle \mid \exists y_1 \text{Check}_A(M, 0, y_1) \wedge \forall y_2 \neg \text{Check}_H(M, 1, y_2)\}$
 is in both $\forall \exists$ and $\exists \forall$ (though not in $\exists$, and not in $\forall$)

