17 April 2024

Computability & Undecidability

Plan

* Languages
* Announcements
* What can Turing Machines compute?

So Far

* Turing Machines: a formal model of "algorithms"

Church-Turing Thesis:

Everything that can be computed by a physically-realizable model of computation can be computed by Turing Machines.

So... what can Turing Machines compute?

Decision Problems $\longleftrightarrow$ Formal Languages

A set of strings.

Decision Problems $\longleftrightarrow$ Formal Languages

Ex. $\underbrace{\text{3SAT}}$

Decision Prob. Given a 3CNF φ, is there a satisfying assignment $\vec{\alpha}$ s.t. $\varphi(\vec{\alpha}) = T$?

Decision Problems $\longleftrightarrow$ Formal Languages

Ex. $\underbrace{\text{3SAT}}$

Decision Prob. Given a 3CNF φ, is there a satisfying assignment $\vec{\alpha}$ s.t. $\varphi(\vec{\alpha}) = T$.

Language. The set of all satisfiable 3CNF formulas φ.

A **Language** $L \subseteq \Sigma^*$ is a subset of strings

Σ = input alphabet

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

A Language $\underbrace{L \subseteq \Sigma^*}$ is a subset of strings

Σ = input alphabet

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

Ex.

$$\text{PALINDROME} = \{ s_{\in \Sigma^*} : s \text{ is a palindrome} \}$$

A <u>Language</u> $\underbrace{L \subseteq \Sigma^*}$ is a subset of strings

Σ = input alphabet

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

<u>Ex.</u>

$$3SAT = \{ \langle \varphi \rangle : \exists \vec{x} \text{ s.t. } \varphi(\vec{x}) = T \}$$

φ is a 3CNF $\wedge$

A **Language** $L \subseteq \Sigma^*$ is a subset of strings

Σ = input alphabet

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

Ex.

$$3SAT = \{ \langle \varphi \rangle : \underbrace{\varphi \text{ is a 3CNF} \wedge \exists \vec{x} \text{ s.t. } \varphi(\vec{x}) = T}\}$$

$\langle \cdot \rangle$ denotes string encoding of math object

Can assume objects of interest are encoded as strings $w \in \Sigma^*$.

Example Languages

$$3SAT = \{ \langle \varphi \rangle : \varphi \text{ is a 3CNF} \wedge \exists \vec{x} \text{ s.t. } \varphi(\vec{x}) = T \}$$

$$PALINDROME = \{ s : s \text{ is a palindrome} \}$$

Example Languages

$$3SAT = \left\{ \langle \varphi \rangle : \begin{array}{l} \varphi \text{ is a } 3CNF \wedge \\ \exists \vec{x} \text{ s.t. } \varphi(\vec{x}) = T \end{array} \right\}$$

$$PALINDROME = \{ s : s \text{ is a palindrome} \}$$

Complement Languages

$$\overline{L} = \Sigma^* \setminus L$$

$$\overline{PALINDROME} = \Sigma^* \setminus PALINDROME = \{ s : s \text{ is } \underline{\text{NOT}} \text{ a palindrome} \}$$

Example Languages

$$3SAT = \left\{ \langle \varphi \rangle : \begin{array}{l} \varphi \text{ is a 3CNF } \wedge \\ \exists \vec{x} \text{ s.t. } \varphi(\vec{x}) = T \end{array} \right\}$$

$$\text{PALINDROME} = \{ s : s \text{ is a palindrome} \}$$

$$\text{EMPTY-LANG} = \emptyset = \{ \ \}$$

$$\text{ALL} = \Sigma^* = \{ w : w \in \Sigma^* \}$$

$$\overline{\text{ALL}} = \text{EMPTY}$$

$$\overline{\text{EMPTY}} = \text{ALL}$$

Example Languages

$$3SAT = \left\{ \langle \varphi \rangle : \begin{array}{l} \varphi \text{ is a 3CNF} \wedge \\ \exists \vec{x} \text{ s.t. } \varphi(\vec{x}) = T \end{array} \right\}$$

$$PALINDROME = \{ s : s \text{ is a palindrome} \}$$

GOLDBACH COUNTEREXAMPLES =

$$\left\{ 0^n : \begin{array}{l} n > 2 \text{ is even} \wedge \\ \forall p,q \text{ prime} . \ n \neq p+q \end{array} \right\}$$

Goldbach's Conjecture $\Longrightarrow$ GOLDBACH COUNTEREXAMPLES $= \emptyset$

Announcements

* Prelim 2 Graded
  - ↳ Returned this evening
* HW8 Released TODAY
  - ↳ Due Thurs, 25 Apr. 11:59p.

Turing Machines <u>recognize</u> languages.

Turing Machines <u>recognize</u> languages.

A language L is <u>Recognizable (RE)</u> if there exists a TM M s.t.

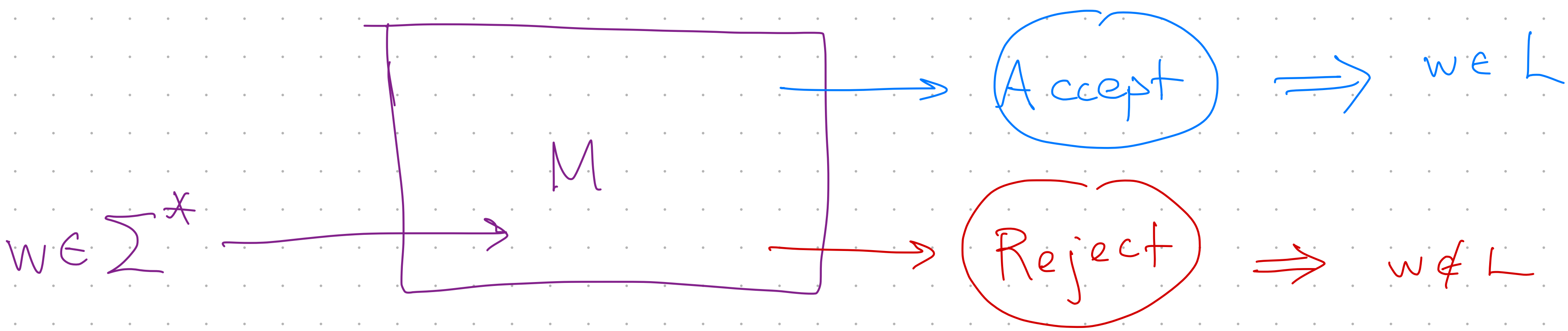
M accepts w iff $w \in L$

$w \in \Sigma^*$

Turing Machines <u>recognize</u> languages.

A language L is <u>Recognizable (RE)</u> if there exists a TM M s.t.

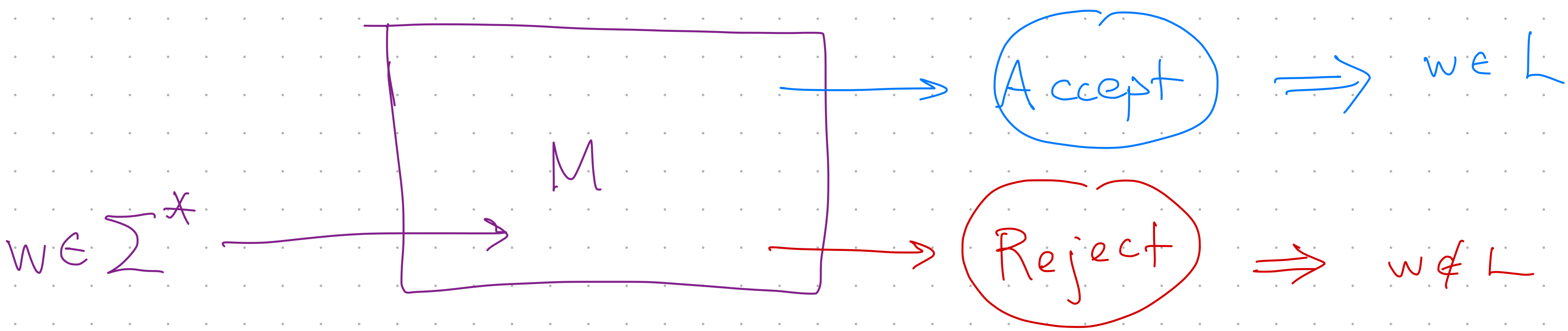
M accepts w iff $w \in L$



Turing Machines <u>recognize</u> languages.

A language L is Recognizable (RE) if there exists a TM M s.t.

M accepts w iff $w \in L$

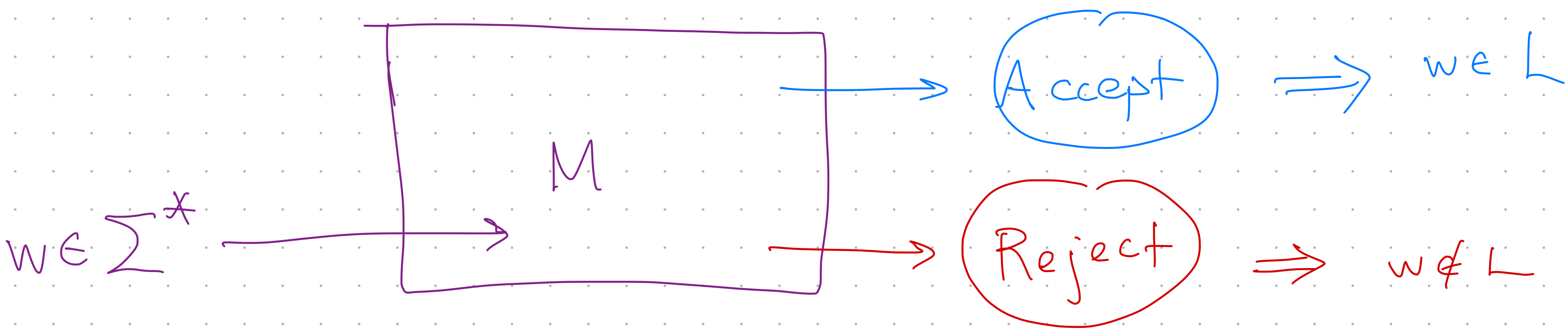


$$\mathcal{L}(M) = \{ w \in \Sigma^* : M \text{ accepts } w \}$$

Turing Machines <u>recognize</u> languages.

A language L is <u>Recognizable (RE)</u> if there exists a TM M s.t.

$$\boxed{\mathcal{L}(M) = L.}$$



$$\mathcal{L}(M) = \{ w \in \Sigma^* : M \text{ accepts } w \}$$

A language L is coRecognizable (coRE) if there exists a TM M s.t.

M rejects w iff $w \notin L$

Equivalently

$L \in \text{coRE}$ iff $\overline{L} \in \text{RE}$

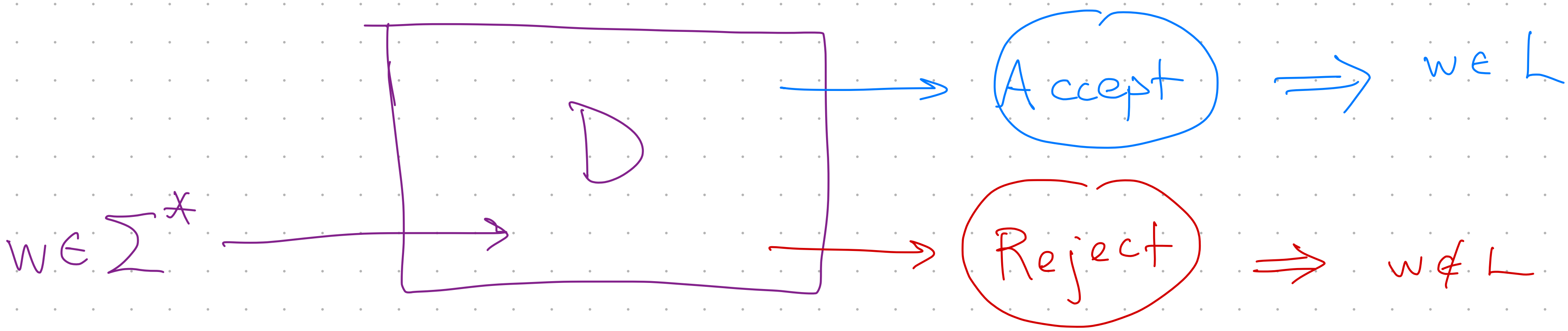
(change the Accept and Reject states)

Some languages are <u>Decidable</u> (R).

A Turing Machine D is called a <u>Decider</u> if it always halts.

Some languages are **Decidable** (R).

A Turing Machine D is called a **Decider** if it always halts.



$\forall w \in \Sigma^*$, Running D on input w ends in either Accept or Reject in **finite** steps.

Some languages are <u>Decidable</u> (R).

A language L is <u>decidable</u> if there exists a decider D s.t. $\mathcal{L}(D) = L$.

Some languages are <u>Decidable</u> (R).

A language L is <u>decidable</u> if there exists a decider D s.t. $\mathcal{L}(D) = L$.

$\Sigma^* \setminus L$.

<u>Theorem</u>. L is decidable if and only if L and $\overline{L}$ are recognizable.

$$L \in R \iff L \in RE \cap coRE$$

Theorem. L is decidable if and only if L and $\overline{L}$ are recognizable.

Pf. ($\Longrightarrow$) L is decidable $\Rightarrow$ $\exists$ decider D s.t. $\mathcal{L}(D) = L$.

Observe: A decider is a recognizer!

$L \in RE$? D accepts w iff $w \in L$

$\Rightarrow L \in RE$

$L \in coRE$? D rejects w iff $w \notin L$

$\Rightarrow L \in coRE$

✓

Theorem. L is decidable if and only if L and $\overline{L}$ are recognizable.

Pf. $(\Leftarrow)$

* $L \in RE \Rightarrow \exists$ machine M s.t. M acc. $w \iff w \in L$
* $\overline{L} \in RE \Rightarrow \exists$ machine N s.t. N rej $w \iff w \notin L$

We construct a multi-tape decider D using M and N

Decider D.

On input w —

* copy w to second tape
* In parallel
  - ↳ Run M on w on tape 1
  - ↳ Run N on w on tape 2
* if either machine Accepts / Rejects, do the same.

Decider D.

On input w —

* copy w to second tape
* In parallel
  - ↳ Run M on w on tape 1
  - ↳ Run N on w on tape 2
* if either machine Accepts / Rejects, do the same.

Suppose $w \in L$ $\Longrightarrow$ M accepts w

$\Longrightarrow$ N does not reject w.

$\Big\} \Longrightarrow$ D accepts w

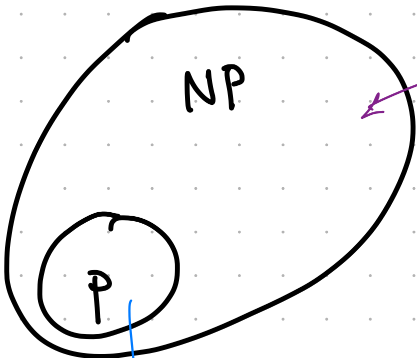
Decider D.

On input w —
* copy w to second tape
* In parallel
  ↳ Run M on w on tape 1
  ↳ Run N on w on tape 2
* if either machine Accepts / Rejects, do the same.

Suppose $w \in L$ $\Longrightarrow$ M accepts w
$\Longrightarrow$ N does not reject w. $\Bigg\}$ $\Longrightarrow$ D accepts w

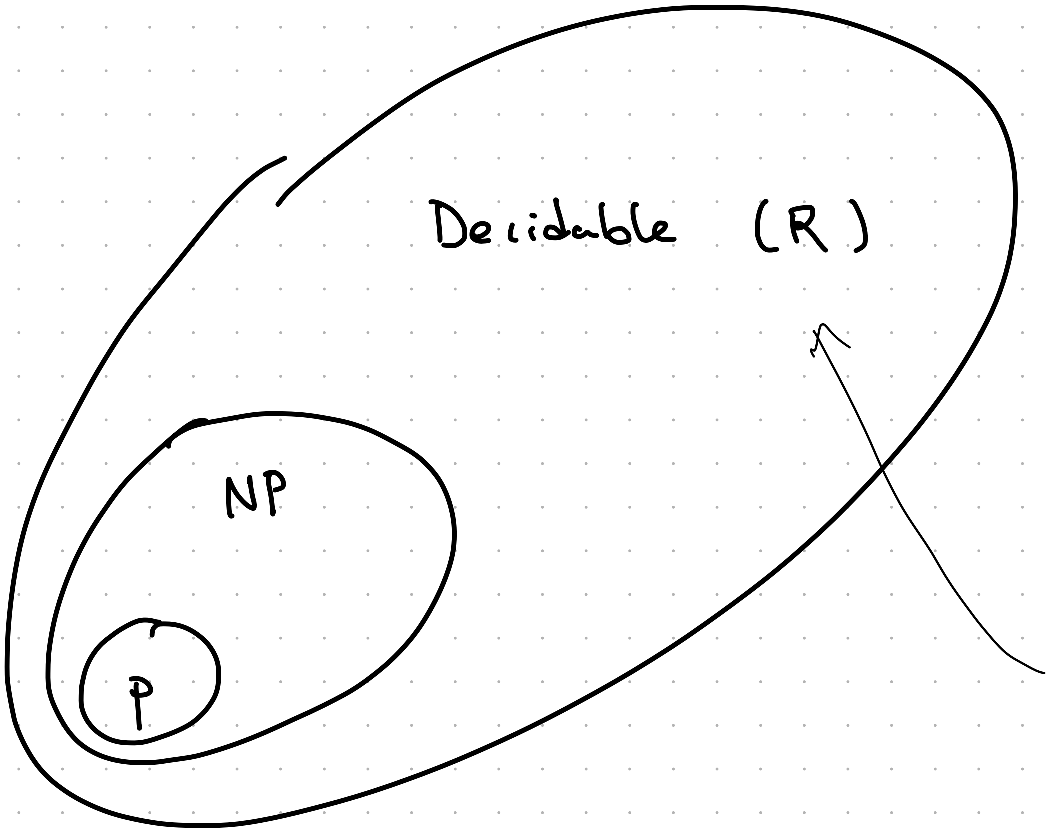
Suppose $w \notin L$ $\Longrightarrow$ N rejects w
M does not accept w $\Bigg\}$ $\Longrightarrow$ D rejects w

Where are we?

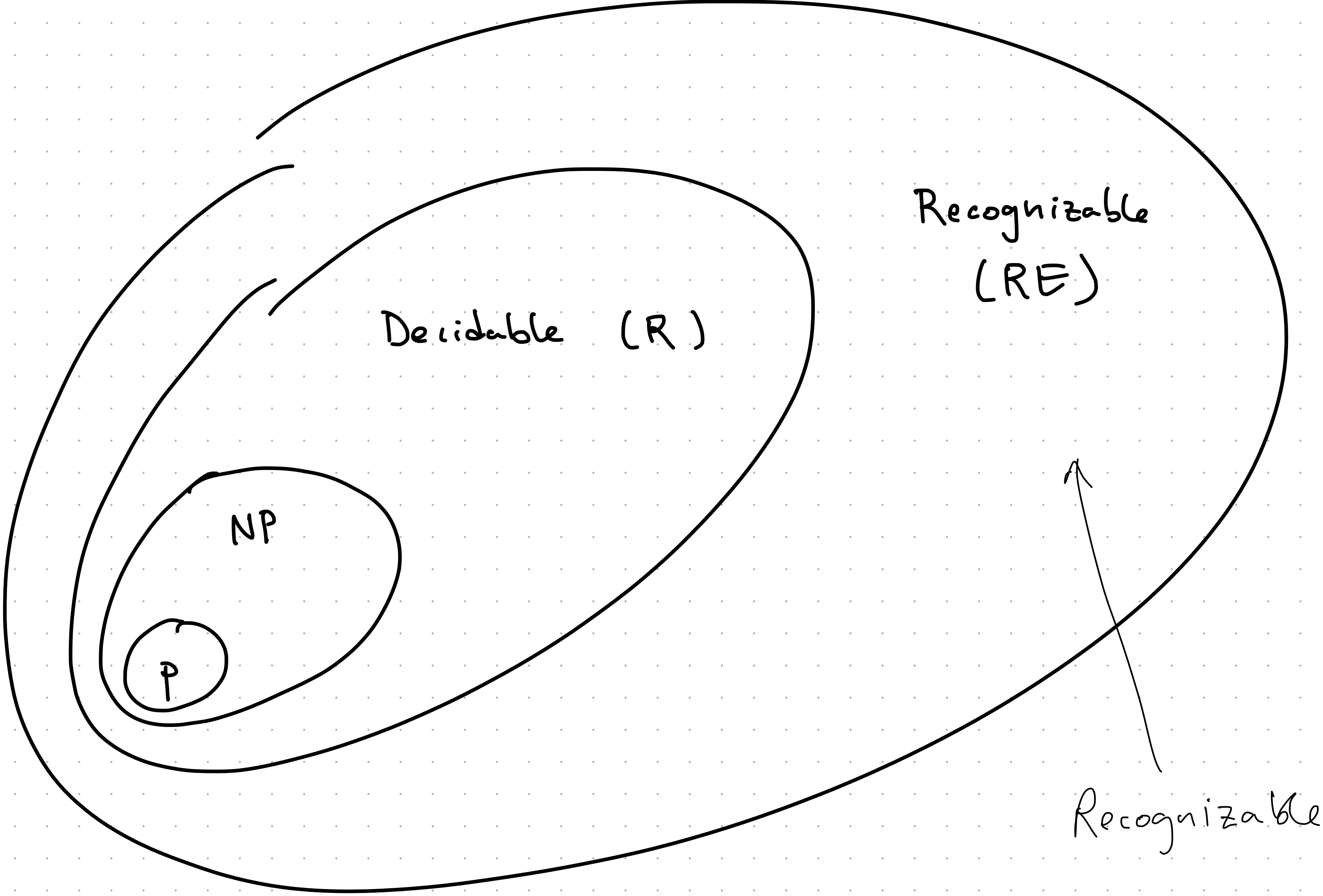


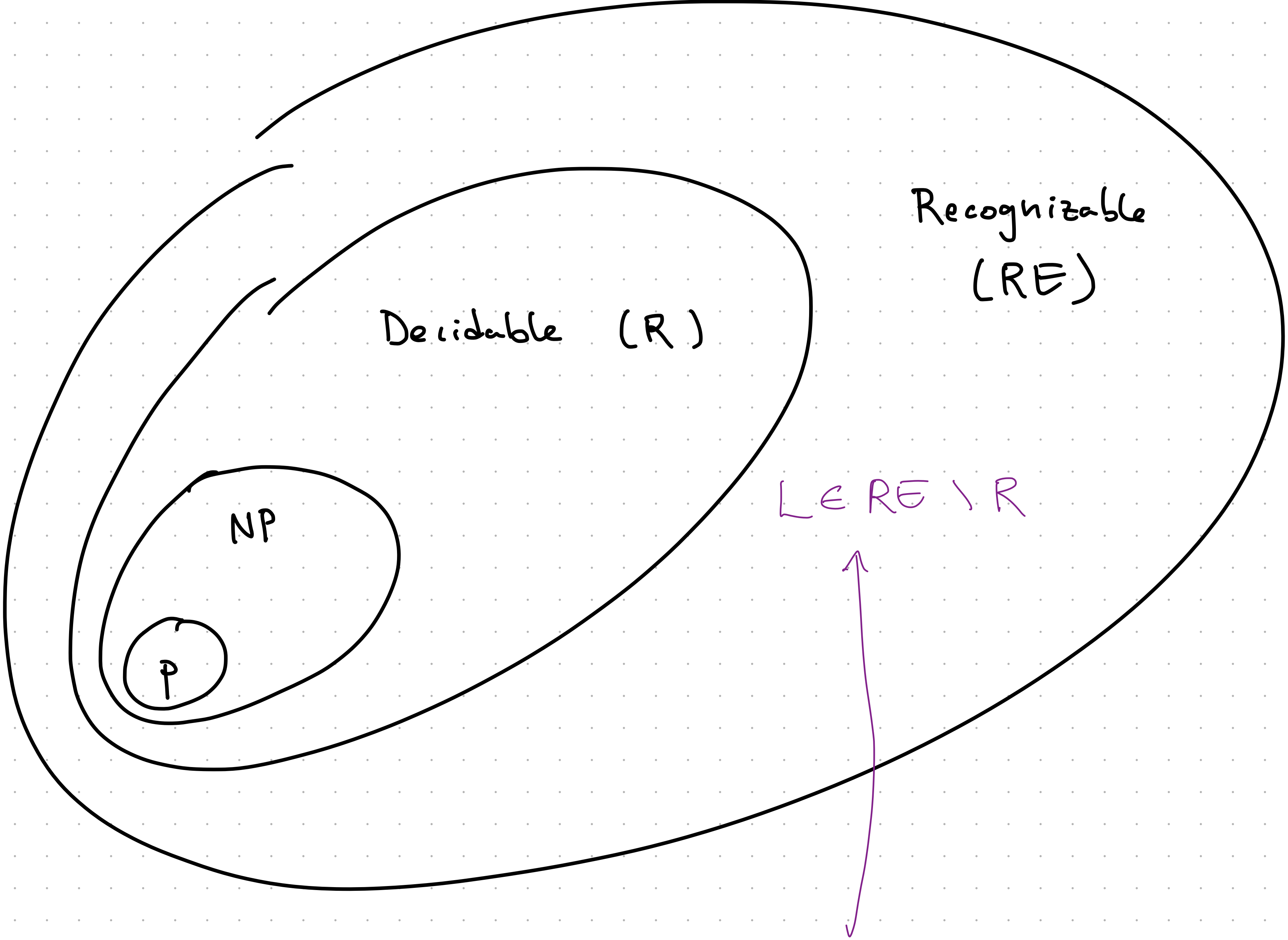
Decidable in exponential time (try all witnesses and verify)

Decidable in polynomial time

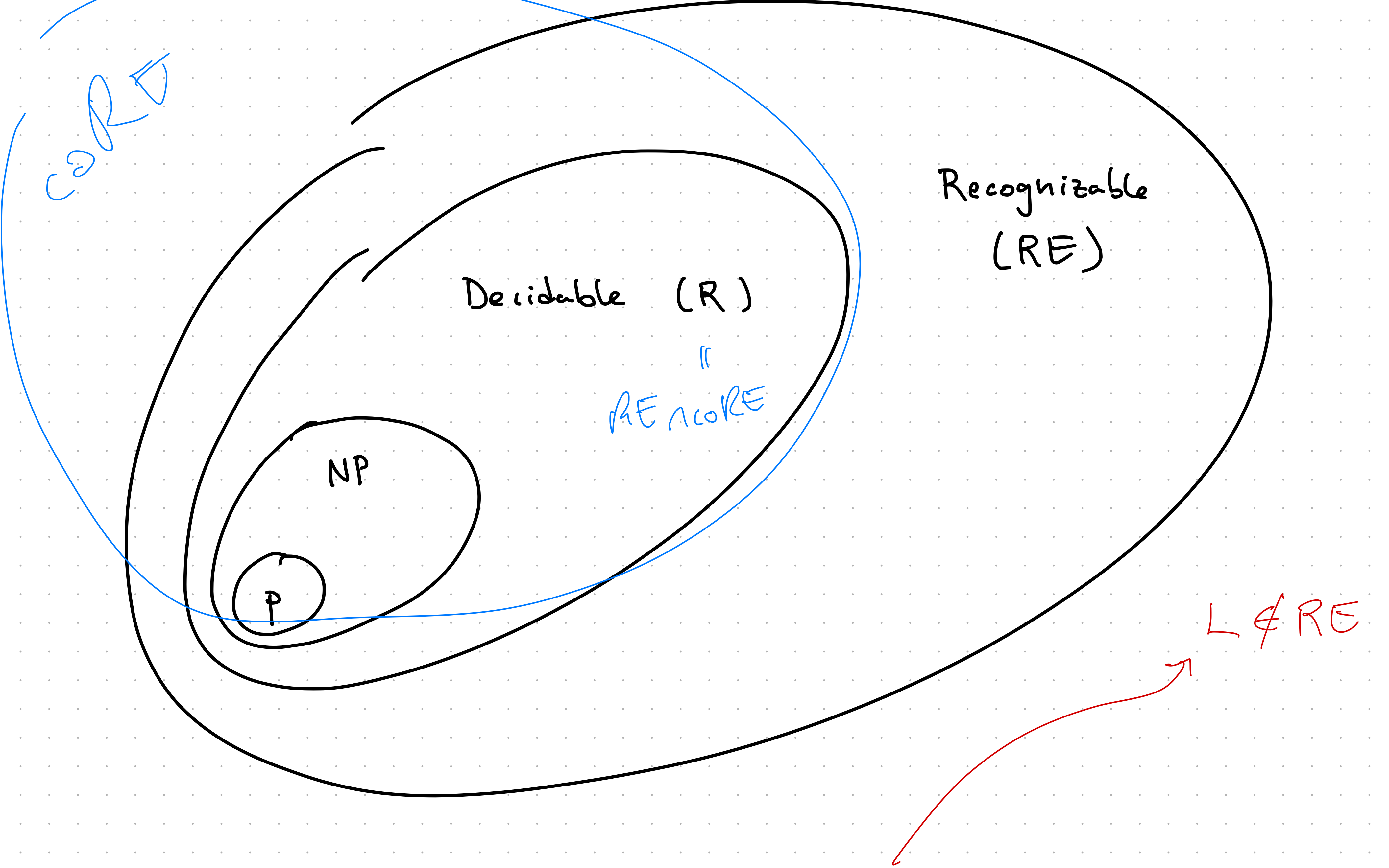


Decidable in finite time.


Recognizable
(RE)
Decidable (R)
NP
P
Recognizable



<u>Claim</u>. There exist <u>Undecidable</u> languages!



Claim. There exist **Unrecognizable** languages!!!

Key to Undecidability. Self-Reference.

* Languages about Turing Machines!

$$L_{property} = \{ \langle M \rangle : \underbrace{M \text{ has some property}}_{\text{property of interest}} \}$$

TM descriptions $\subseteq \Sigma^*$

property of interest

The Diagonal Language

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

TMs that do not accept
their own description

<u>Theorem</u>. DIAG is undecidable!

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

Pf. By contradiction. Suppose $\exists$ D that decides DIAG.

Consider the behavior of D running on $\langle D \rangle$.

$\mathrm{DIAG} = \{ \langle M \rangle : M$ does <u>NOT</u> accept $\langle M \rangle \}$

Pf. By contradiction. Suppose $\exists$ D that decides DIAG.

Consider the behavior of D running on $\langle D \rangle$.

Case 1

Suppose $\langle D \rangle \in \mathcal{L}(D) = \mathrm{DIAG}$.

$\Rightarrow \langle D \rangle \in \mathrm{DIAG} \Rightarrow$ D does <u>NOT</u> accept $\langle D \rangle$

$\Rightarrow \langle D \rangle \notin \mathcal{L}(D)$.

contradiction!

$$\mathrm{DIAG} = \{ \langle M \rangle : M \text{ does } \underline{\text{NOT}} \text{ accept } \langle M \rangle \}$$

Pf. By contradiction. Suppose $\exists$ D that decides DIAG.

Consider the behavior of D running on $\langle D \rangle$.

Case 1

Suppose $\langle D \rangle \in \mathcal{L}(D) = \mathrm{DIAG}$.

$\Rightarrow \langle D \rangle \in \mathrm{DIAG} \Rightarrow$ D does $\underline{\text{NOT}}$ accept $\langle D \rangle$

$\Rightarrow \langle D \rangle \notin \mathcal{L}(D)$.

contradiction!

Case 2

Suppose $\langle D \rangle \notin \mathcal{L}(D) = \mathrm{DIAG}$.

$\Rightarrow \langle D \rangle \notin \mathrm{DIAG} \Rightarrow$ D accepts $\langle D \rangle$

$\Rightarrow \langle D \rangle \in \mathcal{L}(D)$.

Contradiction!