15 April 2025 The Limits of Computation

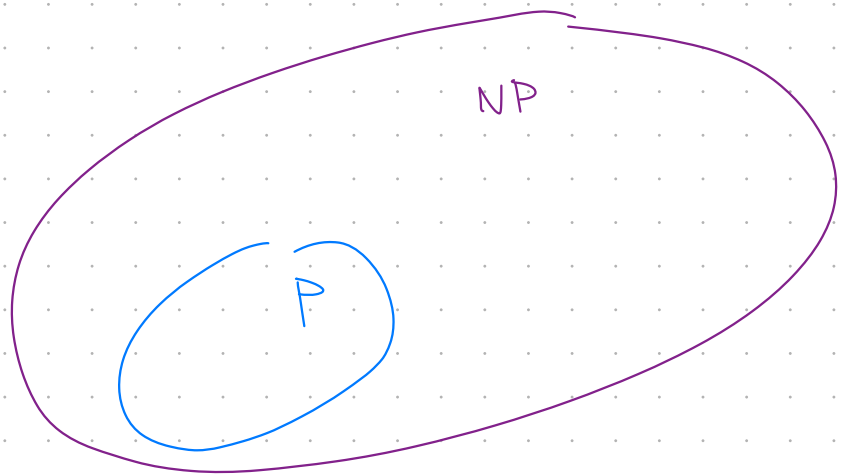
Plan

* Formal Languages & Programs
* Announcements
* Undecidability.

So Far

* Which problems can be solved efficiently ?

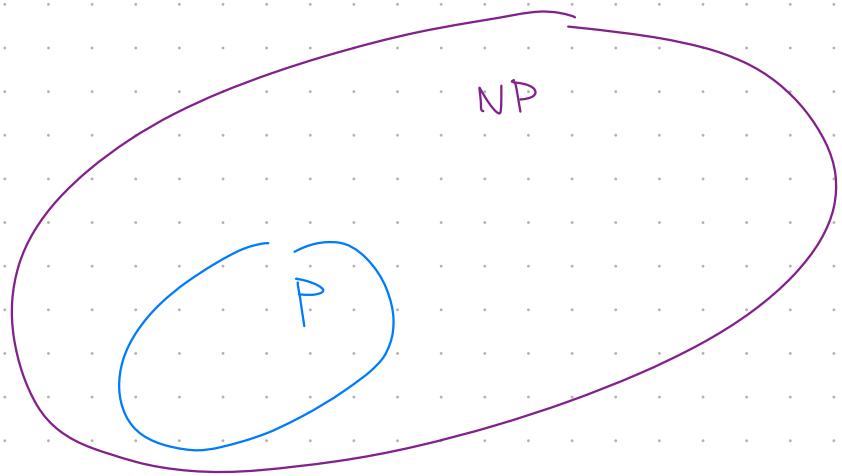
vs. solved inefficiently ?



So Far

* Which problems can be solved efficiently?

vs. Solved inefficiently?



Beyond NP...

Today

* Which problems can be solved?

AND

Which cannot?

How to reason about what CANNOT be solved?

Formalize

* Problems ≡ subsets of strings
* Programs ≡ strings
* Identify problem that is not solved by any program.

How to reason about what CANNOT be solved?

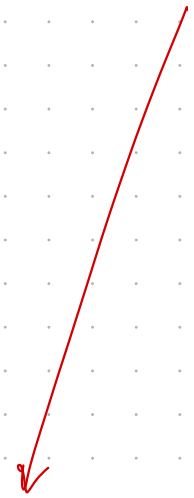
Formalize

* Problems ≡ subsets of strings
* Programs ≡ strings
* Identify problem that is not solved by any program.

Next week

* Formalize the notion of a program as a Turing Machine

Decision Problems ⟷ Formal Languages



A set of strings.

Decision Problems ⟷ Formal Languages

Ex. 3SAT

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

Decision Problems $\longleftrightarrow$ Formal Languages

Ex. 3SAT

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

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

Σ = input alphabet

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

Ex.

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

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

Σ = input alphabet

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

Ex.

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

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

Σ = input alphabet

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

Ex.

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

$\langle \cdot \rangle$ 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

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

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

Complement Languages

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

$$L_{ALL} = \Sigma^*$$

$$L_{EMPTY} = \emptyset$$

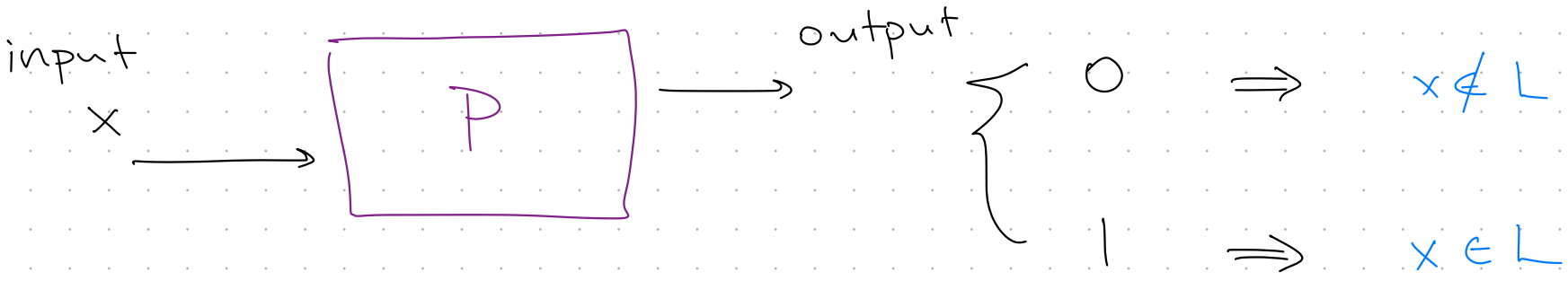
$$L_{ALL} = \overline{L_{EMPTY}}$$

Announcements

* Finish HW7

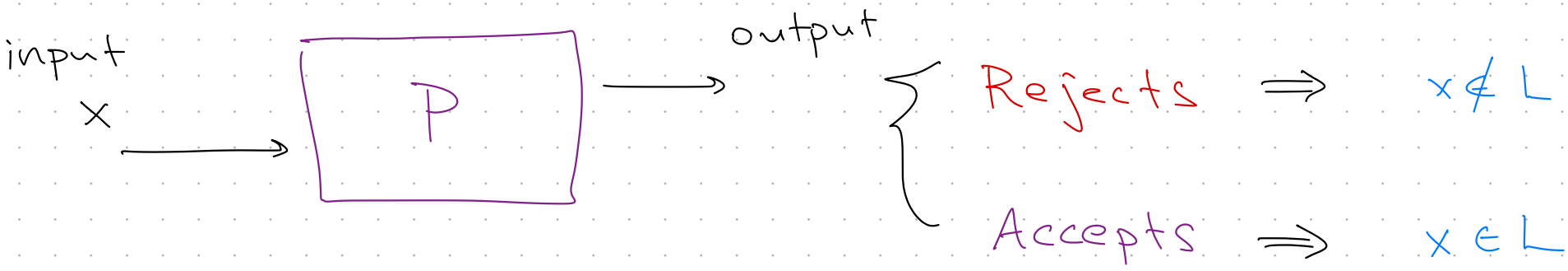
Programs

* Give instructions for solving a problem L



Programs

* Give instructions for solving a problem L



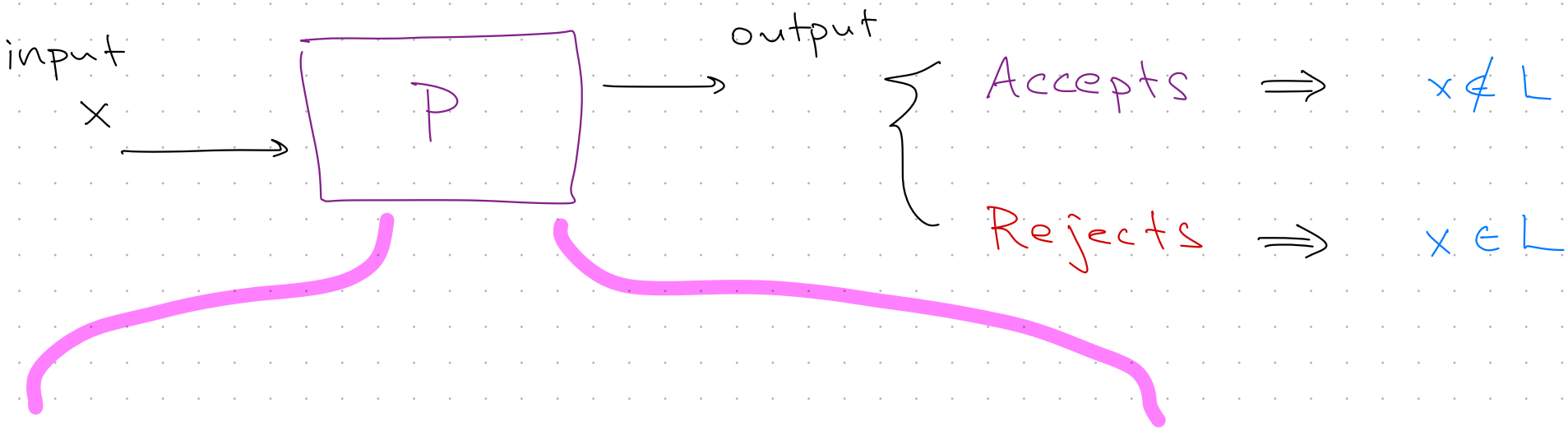
Language of a program.

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

P recognizes L if $\mathcal{L}(P) = L$.

Programs

* Give instructions for solving a problem L



Programs can be encoded as Strings

$$\langle P \rangle \in \Sigma^*$$

e.g. solve.py

Problems vs. Programs

Problems ≡ Languages ≡ Subsets of Strings

Programs ≡ Strings

Problems vs. Programs

Problems $\equiv$ Languages $\equiv$ Subsets of Strings

$$L \in \mathcal{P}(\Sigma^*)$$

← power set of Σ^*

Programs $\equiv$ Strings

$$\langle P \rangle \in \Sigma^*$$

← set of all strings

Problems vs. Programs

Problems $\equiv$ Languages $\equiv$ Subsets of Strings

$$L \in \mathcal{P}(\Sigma^*)$$

Programs $\equiv$ Strings

$$\langle P \rangle \in \Sigma^*$$

Fact. $|\mathcal{P}(\Sigma^*)| > |\Sigma^*|$

Problems vs. Programs

Problems $\equiv$ Languages $\equiv$ Subsets of Strings

$$L \in \mathcal{P}(\Sigma^*)$$

Programs $\equiv$ Strings

$$\langle P \rangle \in \Sigma^*$$

Fact. $|\mathcal{P}(\Sigma^*)| > |\Sigma^*|$

Corollary. There are problems that are not solved by ANY program!

Problems that can't be solved

Defn. A program P **decides** language L if

for all inputs $x \in \Sigma^*$

P accepts x if $x \in L$ &

P rejects x if $x \notin L$

A language L is **undecidable** if

no program P decides L.

Problems that can't be solved

Defn. A program P **decides** language L if

for all inputs $x \in \Sigma^*$

P accepts x if $x \in L$ &

P rejects x if $x \notin L$

$\mathcal{L}(P) = L$

A language L is **undecidable** if

no program P decides L.

Undecidable Problems

* Problems ABOUT Programs

If programs can be encoded as strings,

$\langle P \rangle$ can be the input to programs

Undecidable Problems

* Problems ABOUT Programs

If programs can be encoded as strings,

$\langle P \rangle$ can be the input to programs

SELF-REFERENCE IS ESSENTIAL COMPONENT OF UNDECIDABILITY

$$\text{DIAGONAL} = \{ \langle P \rangle : P \text{ does } \underline{\text{NOT}} \text{ accept } \langle P \rangle \}$$

↳ The set of encodings of programs
that do NOT accept their own encoding.

$$\text{DIAGONAL} = \{ \langle P \rangle : P \text{ does } \underline{\text{NOT}} \text{ accept } \langle P \rangle \}$$

↳ The set of encodings of programs that do NOT accept their own encoding.

Theorem DIAGONAL is Undecidable !

$$\text{DIAGONAL} = \{ \langle P \rangle : P \text{ does NOT accept } \langle P \rangle \}$$

↳ The set of encodings of programs that do NOT accept their own encoding.

Theorem DIAGONAL is Undecidable!

Pf. By contradiction, assume DIAGONAL is decidable.

$\Rightarrow \exists$ program D that decides DIAGONAL.

SELF-REFERENCE

Consider whether $\langle D \rangle \in$ DIAGONAL ?

$$\text{DIAGONAL} = \{ \langle P \rangle : P \text{ does } \underline{\text{NOT}} \text{ accept } \langle P \rangle \}$$

Suppose $\langle D \rangle \in \text{DIAGONAL}$.

$\Rightarrow$ D does NOT accept $\langle D \rangle$ (By defn. of DIAGONAL)

$$\text{DIAGONAL} = \{ \langle P \rangle : P \text{ does } \underline{\text{NOT}} \text{ accept } \langle P \rangle \}$$

Suppose $\langle D \rangle \in \text{DIAGONAL}$.

$\Rightarrow$ D does NOT accept $\langle D \rangle$ — By defn. of DIAGONAL

$\Rightarrow$ $\langle D \rangle$ is not in $\mathcal{L}(D)$ — By defn. of $\mathcal{L}(D)$

$$\text{DIAGONAL} = \{ \langle P \rangle : P \text{ does } \underline{\text{NOT}} \text{ accept } \langle P \rangle \}$$

Suppose $\langle D \rangle \in$ DIAGONAL.

$\Rightarrow$ D does NOT accept $\langle D \rangle$ — By defn. of DIAGONAL

$\Rightarrow \langle D \rangle$ is not in $\mathcal{L}(D)$ — By defn. of $\mathcal{L}(D)$

$\Rightarrow \langle D \rangle \notin$ DIAGONAL — By assumption $\mathcal{L}(D) =$ DIAGONAL

CONTRADICTION!

$$\text{DIAGONAL} = \{ \langle P \rangle : P \text{ does } \underline{\text{NOT}} \text{ accept } \langle P \rangle \}$$

Suppose $\langle D \rangle \in$ DIAGONAL.

$\Rightarrow$ D does NOT accept $\langle D \rangle$ — By defn. of DIAGONAL

$\Rightarrow$ $\langle D \rangle$ is not in $\mathcal{L}(D)$ — By defn. of $\mathcal{L}(D)$

$\Rightarrow$ $\langle D \rangle \notin$ DIAGONAL — By assumption $\mathcal{L}(D)$ = DIAGONAL

CONTRADICTION!

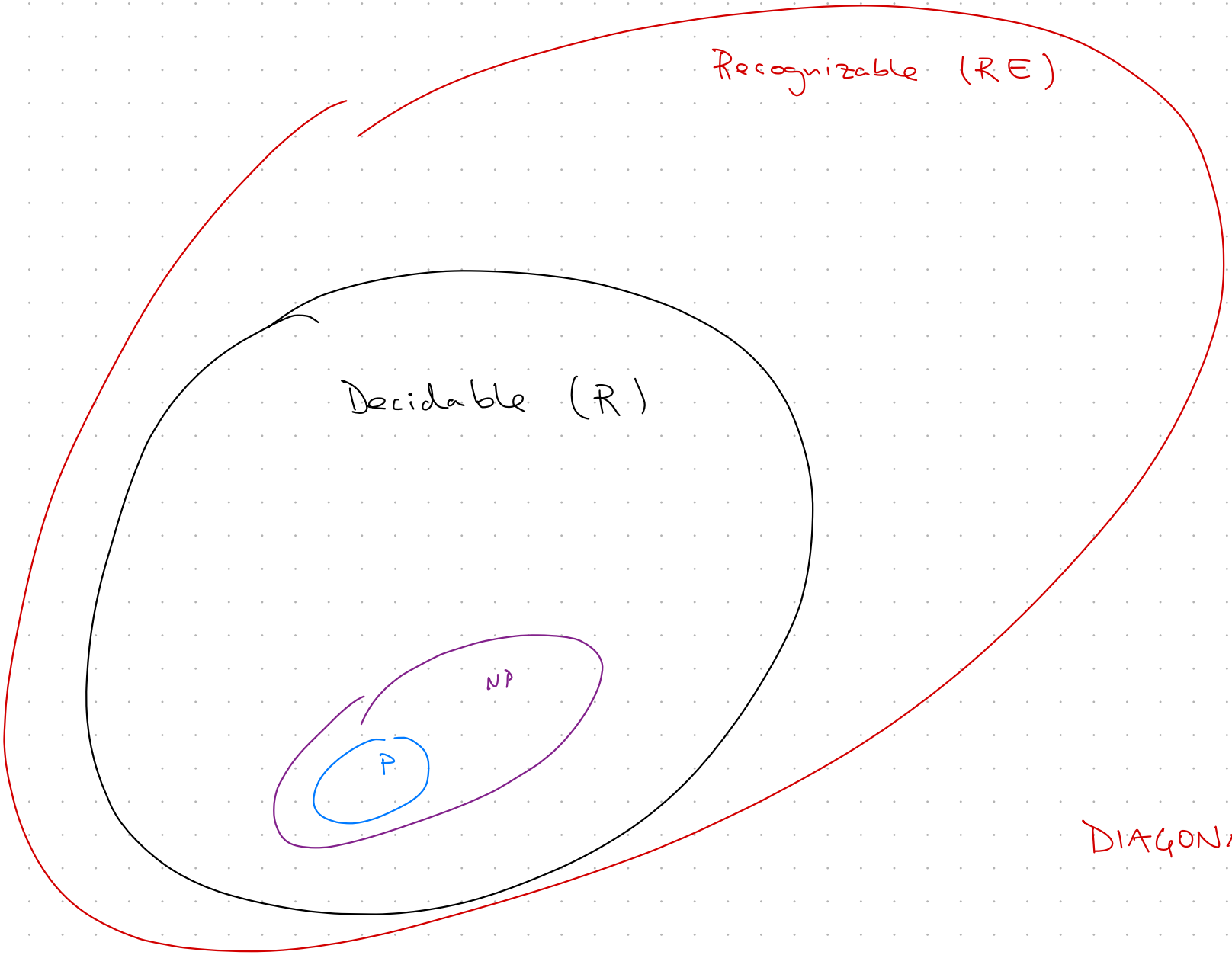
Suppose $\langle D \rangle \notin$ DIAGONAL.

$\Rightarrow$ D accepts $\langle D \rangle$ — By defn. of DIAGONAL

$\Rightarrow$ $\langle D \rangle$ is in $\mathcal{L}(D)$ — By defn of $\mathcal{L}(D)$

$\Rightarrow$ $\langle D \rangle \in$ DIAGONAL — By assumption $\mathcal{L}(D)$ = DIAGONAL

CONTRADICTION!



DIAGONAL

No program can solve DIAGONAL.

So what?

* DIAGONAL is quite contrived ...

So what?

* DIAGONAL is quite contrived ...

* Idea Reductions!

  ↳ show that other problems capture DIAGONAL

  ⟹ these problems are undecidable.

So what?

* DIAGONAL is quite contrived ...

* Idea Reductions!

  ↳ show that other problems capture DIAGONAL

  $\Rightarrow$ these problems are undecidable.

$$\text{HALT} = \{ \langle P, x \rangle : \text{Program } P \text{ halts on input } x \}$$

Theorem. HALT is undecidable.