More o undecidability

Halting Problem (P, I) → reduction → Generalized Halting Problem (any input I for which P halts)

⇒ undecidable

EQUIVALENCE: Given programs P and P', do they do the same thing?

⇒ Undecidable

Proof:

Let P be a program for which we want to decide the halting problem.

Let P' be non-terminating program

Run P and P' through the algorithm that decides equivalence.

↳ YES : P does not halt

↳ NO : P halts

Contradiction. Since we already know halting problem is undecidable.

SPECIFICATION: Does a program do what it is supposed to?

Also, undecidable.

Rice's theorem

Functional property:

→ Property describing ~~a solution~~ input/output behavior

→ Some programs have this property, others don't

	Functional property?
Is a computer program?	X
Always returns 1	✓
Solves vertex cover	✓
Is a computer virus	✓
Requires O(n) time	X

Rice's THM: Any functional property is undecidable.

Proof:

Consider: Does program P have f.p. xy?

Want to solve ~~pr~~ halting problem for P'

P' → X (runs P')
(clears memory)
(exhibit f.p. y)

Assume there's an algo. S(P, y) that decides a f.p. y for program P.

S(X, y) → YES: X has y → P' halts
→ NO: P' doesnot halt

CONTRADICTION. Since halting problem is undecidable.

Property	Decidable?
Always returns 1	X
Solves V.C.	X
Adds 2 nos.	X
Doesn't do anything	X
Converts mp3 file	X
Reads a file	X
Terminates in k timesteps	✓
Is a virus	X
Writes to a file	X

Malware detection

A malware is something that exhibits undesired behavior, and causes disruption

Let's assume we found a formal criteria for malware detection.

⇒ Undecidable

⇒ any program that attempts to solve it cannot be perfect.

~~Halting Problem v/s~~ Incompleteness theorem v/s Undecidability.

- In any formal system, we have a set of axioms (finite & consistent)
- These are some statements, that are true and cannot be proved.
- A system of axioms cannot show it's consistency. (2nd incompleteness thm)

Undecidable v/s Semi-decidable

Does P stop?

P ⟶ YES (can obtain in finite time)
⟶ NO (cannot obtain in finite time)

⇒ We can decide sometimes (Semi decidable)

Semi-decidable problems are recursively enumerable

recursively enumerable:

Program × Input (all combinations)

⇒ That came to halt

```
max_len = 0
while (true):
    max_len = max_len + 1
    for each program P of length ≤ max_len:
        for each input I of length ≤ max_len:
            if P halts on I in max_len steps:
                print P, I
```

→ Eventually finds all combinations of P, I that halts

→ Infeasible in practice