Self Reference, Diagonalization and Uncomputability.

Induction & Recursion
Analysis of algorithms

This statement is false

Russell's Paradox:

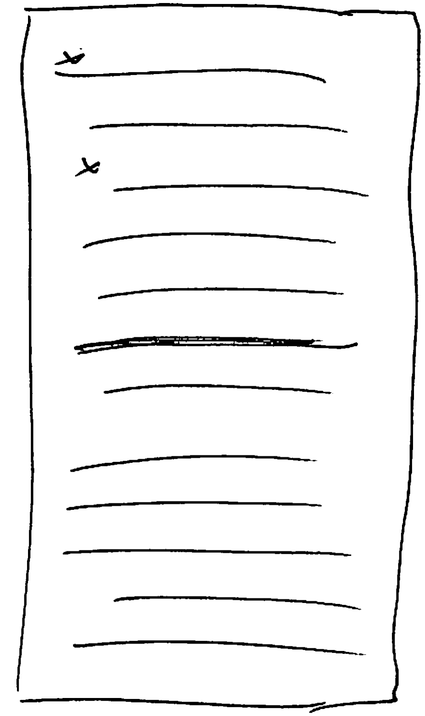
S = all men in village, who don't shave themselves.
(are shaved by Barber,)

Barber $\in S$?

Axioms

Proofs

Peano Axioms (Natural Numbers):



1. $0 \in \mathbb{N}$.
2. $x \in \mathbb{N} \Rightarrow x+1 \in \mathbb{N}$.
3. $x+1 = y+1 \Rightarrow x = y$.
4. $\nexists x : x+1 = 0$
5. Induction Axiom:

$\left\{ \begin{array}{l} S \subseteq \mathbb{N} \\ 0 \in S \quad \text{and if} \quad x \in S \Rightarrow x+1 \in S \end{array} \right.$

then $S = \mathbb{N}$.

$$\left. \begin{array}{l} A \Rightarrow B \\ A \end{array} \right\} \quad \text{infer} \quad B$$

$$\left. \begin{array}{l} \text{All men are mortal} \\ \text{Socrates is a man} \end{array} \right\} \Rightarrow \text{Socrates is mortal.}$$

$$\forall n \geq 3 \ \nexists \ x, y, z \in \mathbb{N} \ \ x, y, z \neq 0 : x^n + y^n = z^n$$

Gödel: Cannot write down a set of axioms:

Prove that all true statements follow from axioms via proofs.

Consistency: Cannot prove consistency

If you prove consistent then it is inconsistent.

Halting Problem:

P, I   does P halt on input I.

Assume ∃ exists.

$$\text{TestHalt}(P, I) = \begin{cases} \text{"yes"} & \text{if } P \text{ halts on input } I \\ \text{"no"} & \text{if } P \text{ does not halt.} \end{cases}$$

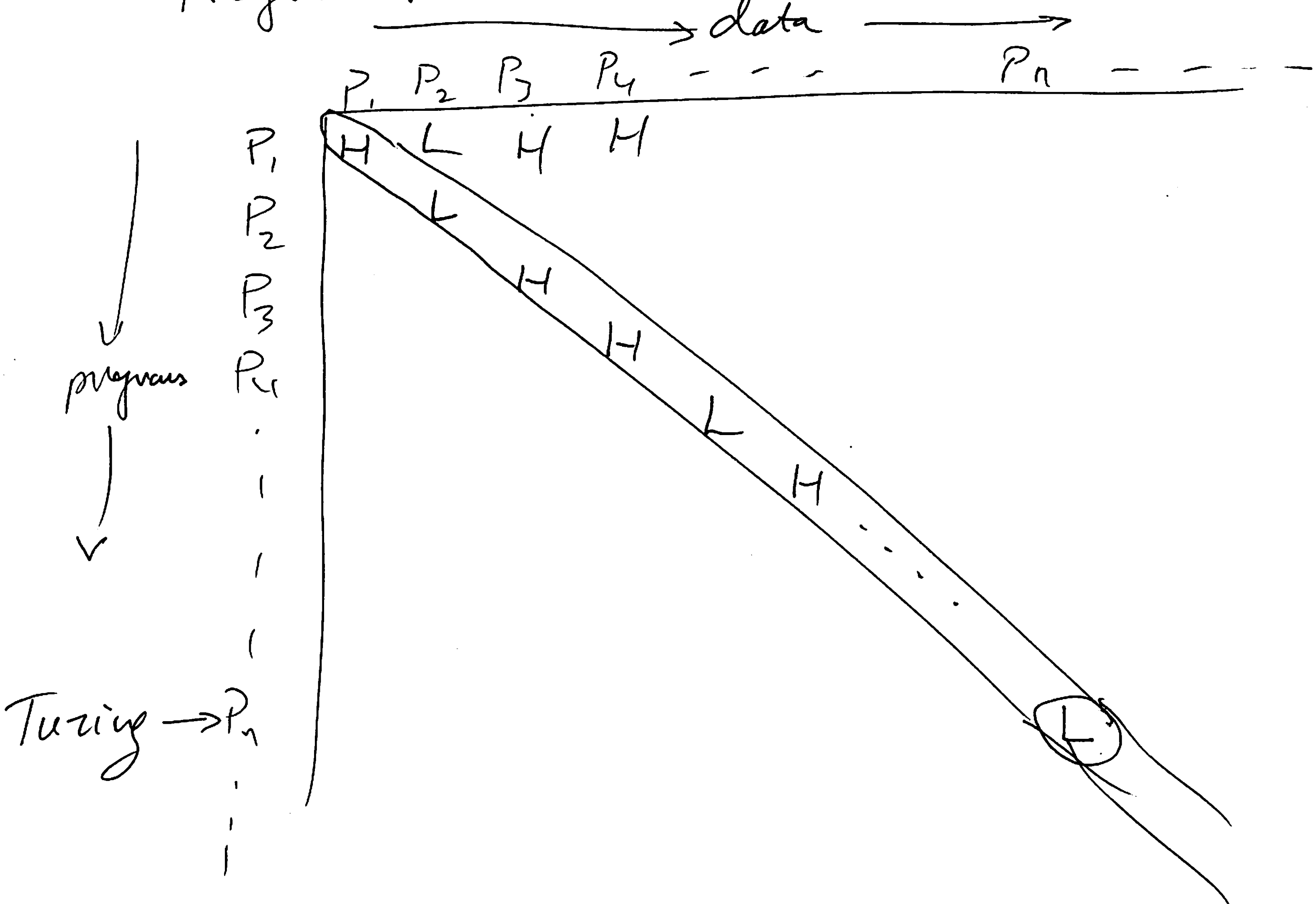
↑ Does not exist!!

```
Turing(P)
    If TestHalt(P, P) = "yes" then loop forever.
    else halt.
```

Turing(Turing) halts?

Contradiction.

Program:



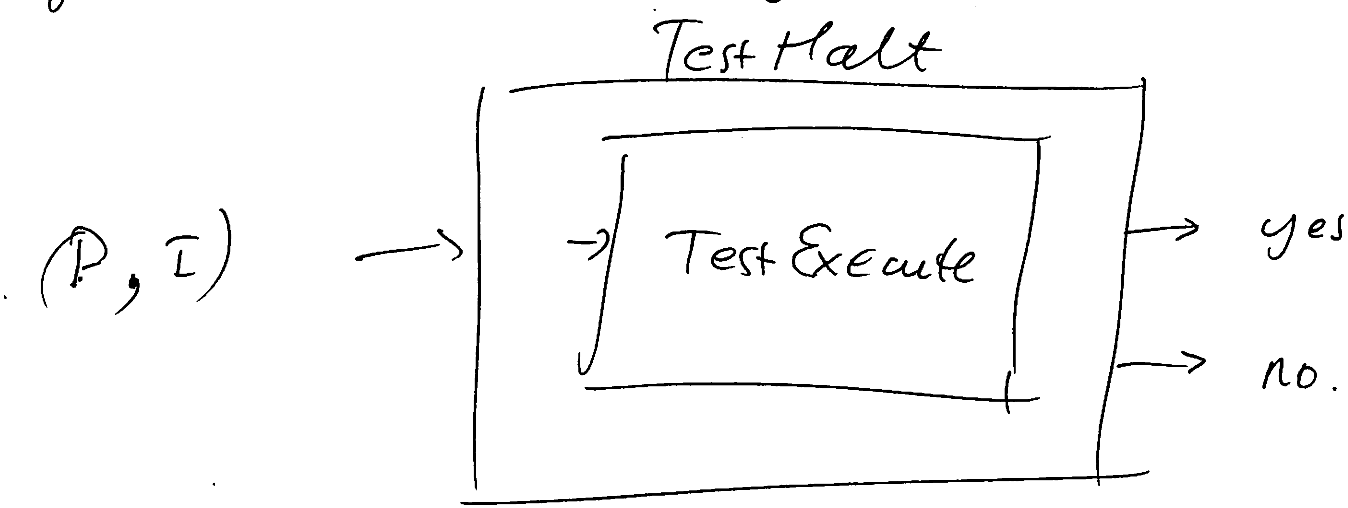
Turing(P) does the opposite of P(P).

∴ Turing does not occur on the list of all programs.

⇒ Turing does not exist

Does program P ever execute a certain line of code?

Show how to use a program Test Execute as a subroutine in a program to solve the Halting problem.



Assume for contradiction that Test Execute exists.
Show how to use it to implement Test Halt.
Contradiction. ∴ Test Execute does not exist.

Virus

~~Program~~.

(Print "Program")

(Quine "compute") = (compute "compute")

(Quine "Quine") = (Quine "Quine").

Induction vs Recursion } CS 170
Analysis of Algorithms

* Modular arithmetic } AI
Probability } Security ...

* Model: Stable Marriage
Error correcting codes
Load Balancing.
Hashing
⋮

* Problem solving