Not all problems can be solved by computers.

- There are uncountably many problems.
- Computers can solve only countably many problems.

An alphabet is a finite set of symbols.

$\{0, 1\}$, $\{0, 1, 2, \ldots, 9, +, -\}$

$\{a, b, \ldots, z, A, B, \ldots, Z\}$

ASCII

Σ, Γ, Δ

A <u>string</u> over Σ is a <u>finite</u> ordered sequence of symbols from Σ.

$\{0, 1\}$

010110

000111

Σ^* = the set of all strings over Σ

Proposition : Σ^* is countable.

Proof : $$\Sigma^* = \bigcup_{\ell \geq 0} \Sigma^\ell$$

$$|\Sigma^\ell| = |\Sigma|^\ell \rightarrow \text{finite}$$

$\rightarrow$ countable.

Σ^* is a countable union of countable sets.

Proposition : $\wp(\Sigma^*)$ is uncountable.

Language: Any subset of Σ^*.

ASCII alphabet

English language is the set of valid English sentences.

Language of integers over $\{0, 1, \ldots, 9, +, -\}$

$$\{0, 1, -1, 2, -2, 3, -3, \ldots\}$$

Language of primes $\{2, 3, 5, 7, 11, 13, \ldots\}$

Problem:

Given a language L over Σ,
and a string $\alpha \in \Sigma^*$,
decide whether $\alpha \in L$.

English: Checking correctness.

Language of integers: Check whether α is syntactically valid

$+235$ $\quad$ $2+35$

Language of primes: Primality-testing problem

How to represent a language?

L finite $\quad L = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$

L infinite $\rightarrow$ exhaustive enumeration not possible

<u>Grammars</u> / English-language / Mathematical

English grammar

$\{a \in \mathbb{N} \mid a \text{ is a prime}\}$

C language — grammar

A grammar / description is a string over some representation alphabet Γ.

Finite

Grammars / descriptions $\in \Gamma^*$

Only countably many languages can have finite descriptions.

L is specified by such a finite description.

Problems that computers can solve

$\subseteq$ Languages that have finite descriptions

An unsolvable problem

HP (The Halting Problem)

Input: A C program P
An input I for P.

Output: H if P halts on input I
L if P does not halt on input I

It is impossible to write a C program that solves (all instances) of the HP.

A diagonalization proof

P — a C program $\in \text{ASCII}^*$

I — a string $\in \text{ASCII}^*$

Each char in ASCII is a sequence of eight bits.

Both P and I are binary strings

$\{0, 1\}^* \longrightarrow \mathbb{N}$

$\alpha \longmapsto (1\alpha)_2$

Bijection $\varepsilon \mapsto 1$

001

$(1001)_2 = 9$

Every C program is a natural number.
Every input is a natural number.

A natural number can be viewed as a C program or an input.

An invalid program can be identified as a program:

```
main( )
{
    exit(0);
}
```

$\exists$ a C program A that for all instances of P and I, outputs H or L (as the case is).

Then I can generate a program B that runs P on P.

B introduces a contradiction.

P \ I	1	2	3	4	5	⋯	B	⋯
1	*H*	*H*	*L*	*H*	*L*			
2	*L*	*L*	*H*	*L*	*L*			
3	*H*	*H*	*H*	*H*	*H*			
4	*L*	*H*	*H*	*H*	*H*			
5	*H*	*H*	*L*	*L*	*L*			
⋮								
B	*L*	*H*	*L*	*L*	*H*	⋯	*H*	⋯
⋮								
B	*L*	*H*	*L*	*L*	*H*	⋯	*L*	⋯

B on input P:

B runs A on P, P.

If A outputs H,
B enters an infinite loop.

If A outputs L,
B exits.

B halts on B <u>and</u>
B loops on B.