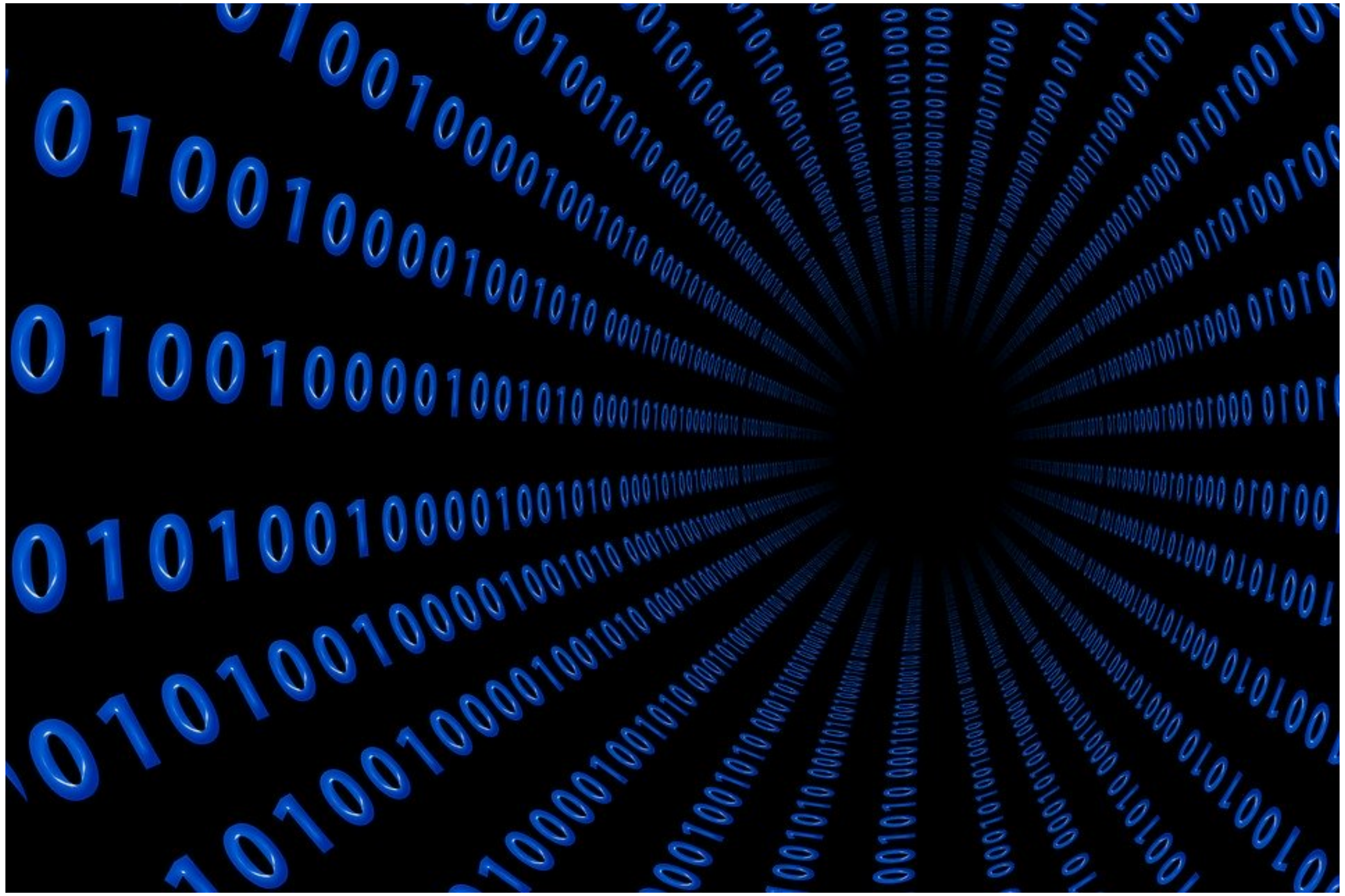
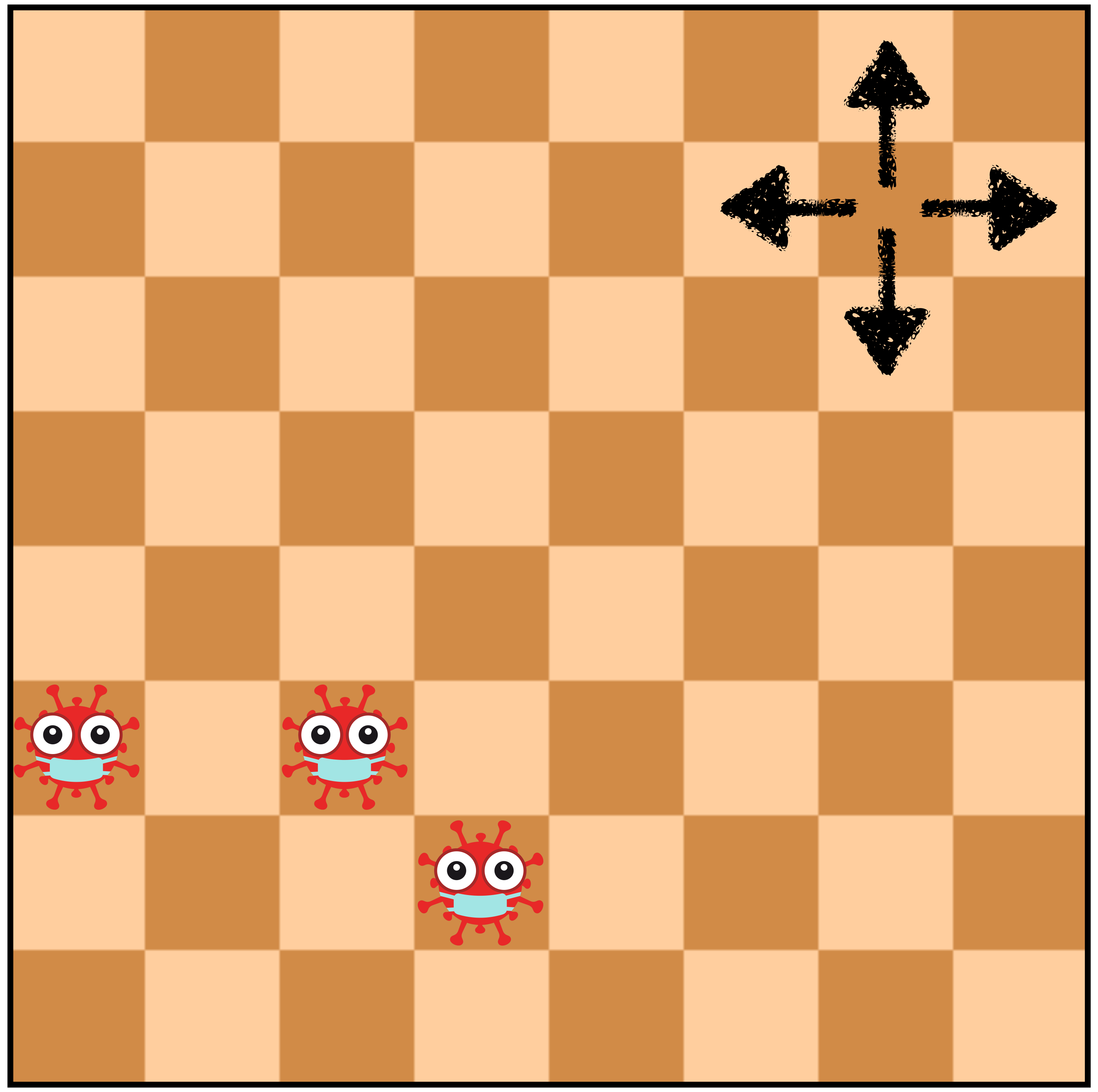
CS251

Great Ideas in *Theoretical* Computer Science



Strings, Encodings, Problems

Covid Puzzle



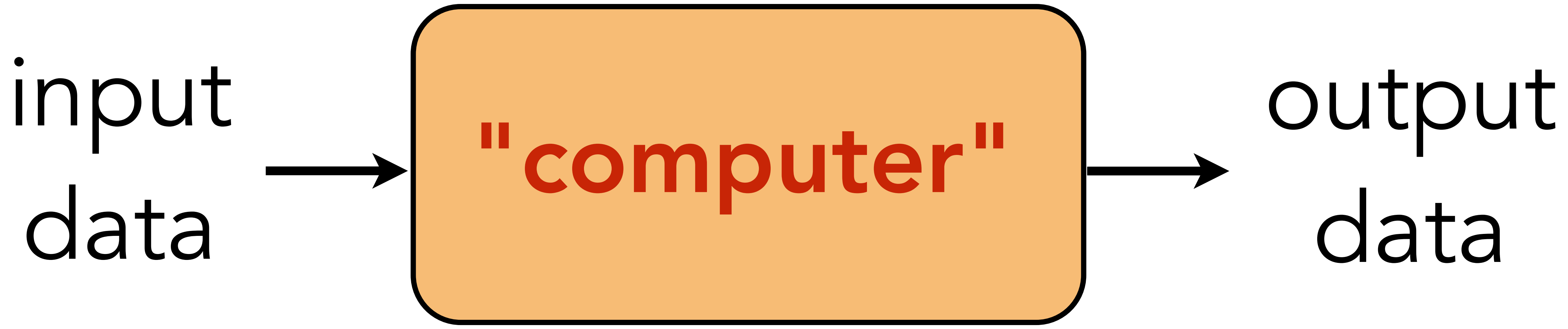
Neighbors in directions **N**, **S**, **W**, **E.**

Initially, some of the squares are **infected**.

If a square has 2 or more **infected** neighbors, it becomes **infected**.

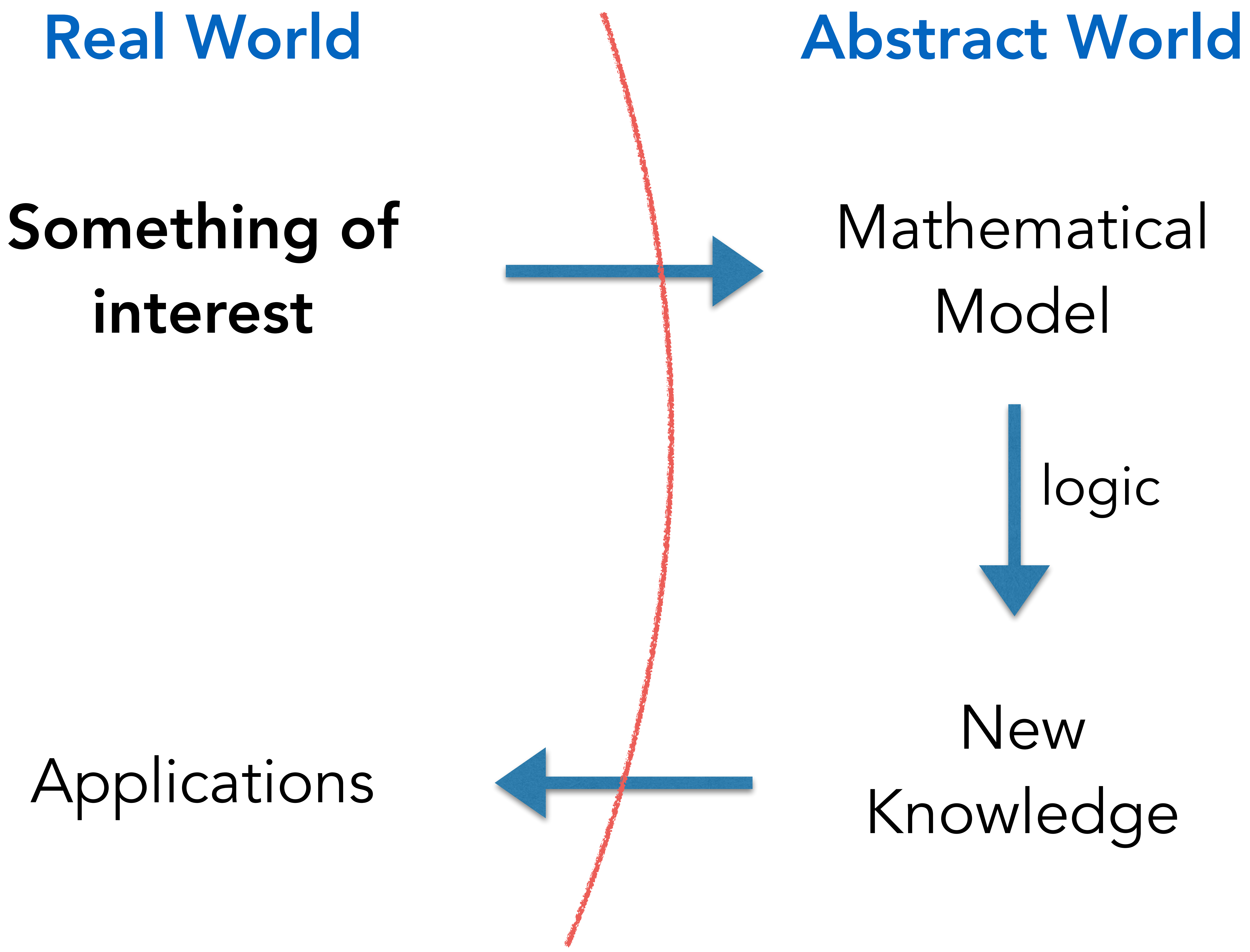
What is the min number of **infected** squares needed initially to infect the whole board?

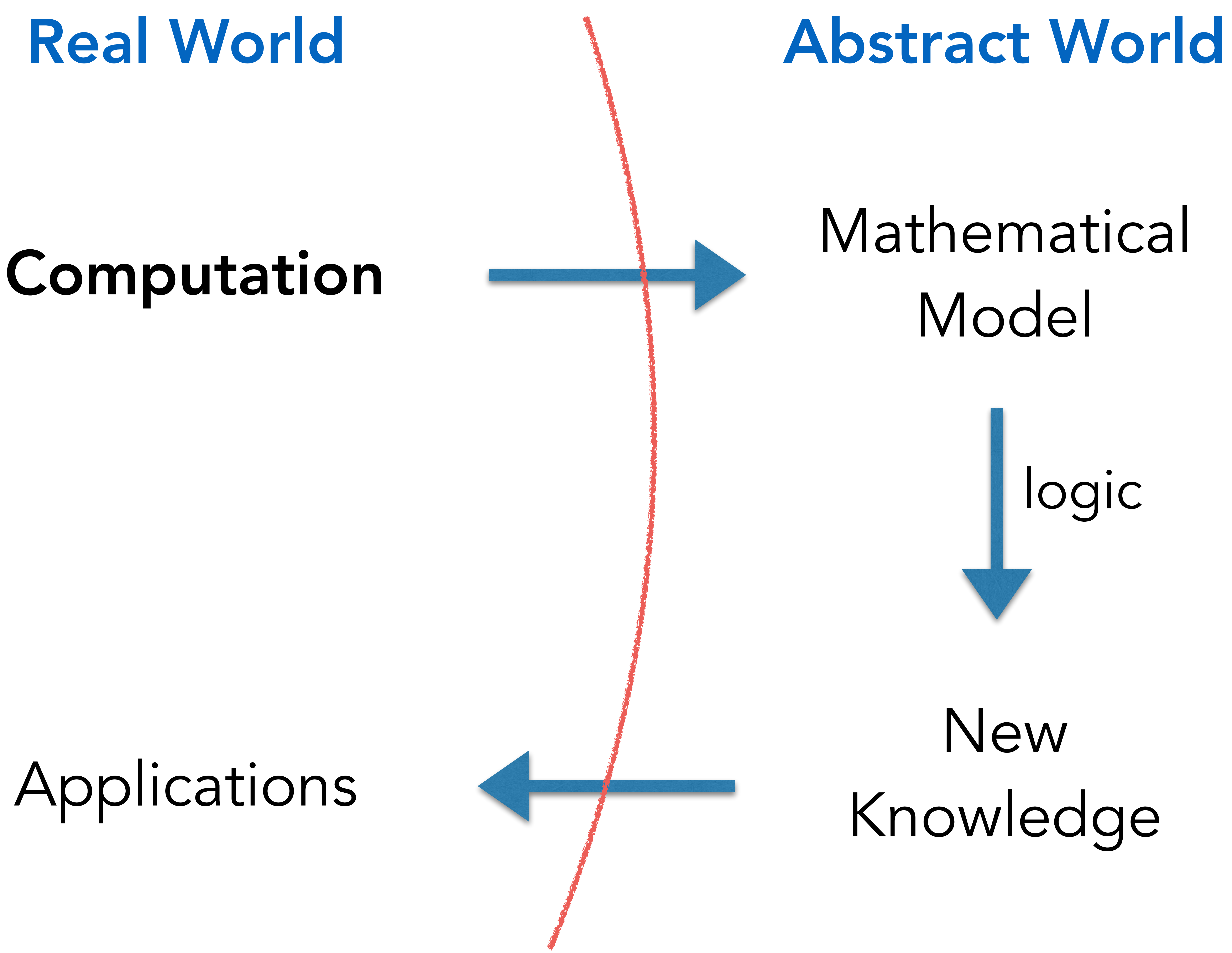
Next Few Chapters



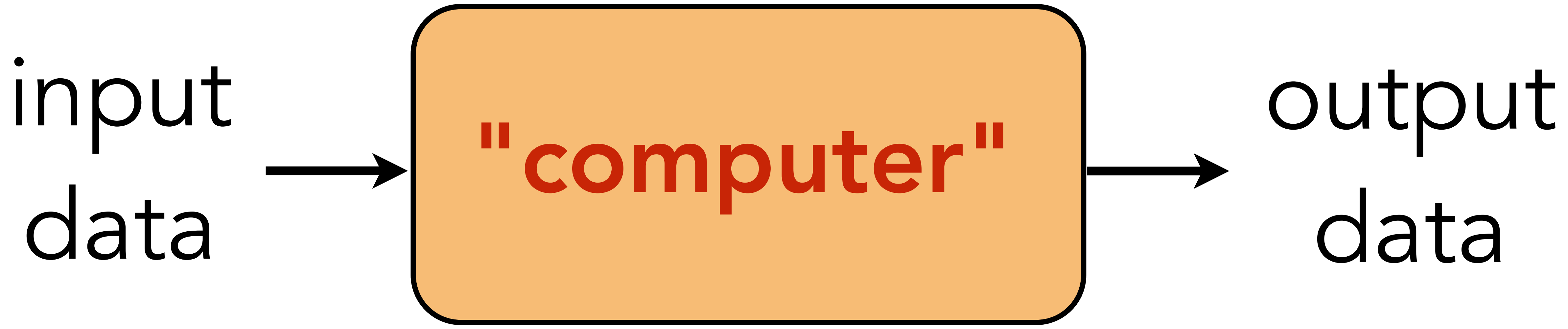
What is **computation**?

What basic properties does it have?

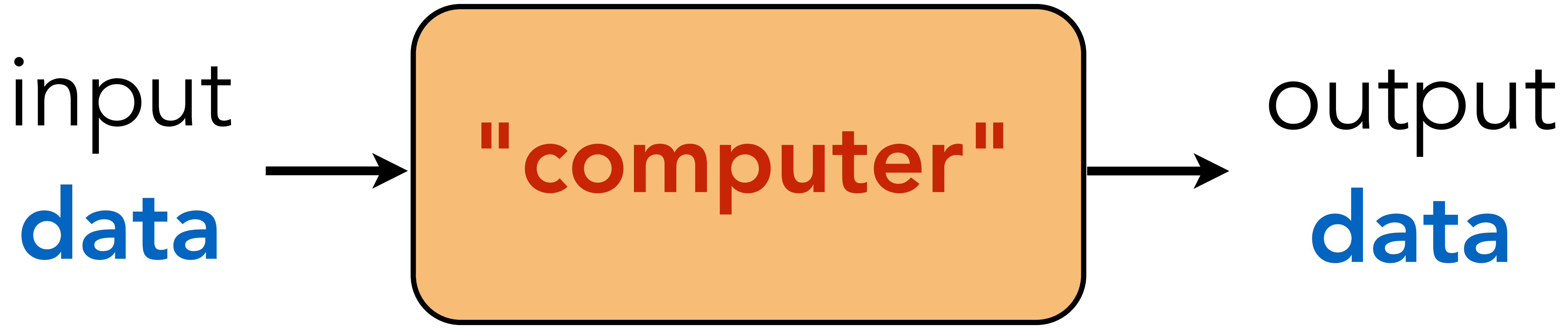

Real World
Abstract World
Something of interest
Mathematical Model
logic
New Knowledge
Applications


Real World
Abstract World
Computation
Mathematical Model
logic
New Knowledge
Applications

Now



Now



How do we mathematically represent **data**?

How we represent information

e.g. written communication:

"apple"

"car"

"happy"

"three" or "3"

English alphabet

$\Sigma = \{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\}$

Turkish alphabet

Σ = {a,b,c,ç,d,e,f,g,ḡ,h,ı,i,j,k,l,m,n,o,ö,p,r,s,ş,t,u,ü,v,y,z}

?

What if we had more symbols?

What if we had less symbols?

Binary alphabet

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

alphabet: a non-empty and finite set.
(usually denoted by Σ).

symbol/character: an element of an alphabet.

string/word: a finite sequence of symbols from Σ.

A string is denoted by $a_1a_2\ldots a_n$, where each $a_i \in \Sigma$.

(the definition sometimes includes infinite sequences)

Example: Some strings over $\Sigma = \{0,1\}$:

ϵ $\quad 0 \quad 1 \quad 01 \quad 101111010110111$

Example: Some strings over $\Sigma = \{a, b, c\}$:

$\epsilon \quad a \quad b \quad c \quad ca \quad caabcccab$

Length of a string s:

$$|s| = \text{the number of symbols in } s.$$

$\Sigma^* =$ set of all finite-length strings over Σ.

Examples:

$$\{0,1\}^* = \{\epsilon,0,1,00,01,10,11,000,001,0010,\ldots\}$$

$$\{a\}^* = \{\epsilon, a, aa, aaa, aaaa, \ldots\}$$

What is an encoding scheme?

$$\Sigma = \{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\}$$

Objects/concepts		String encoding
	$\longrightarrow$	apple
	$\longrightarrow$	car
	$\longrightarrow$	happy

?

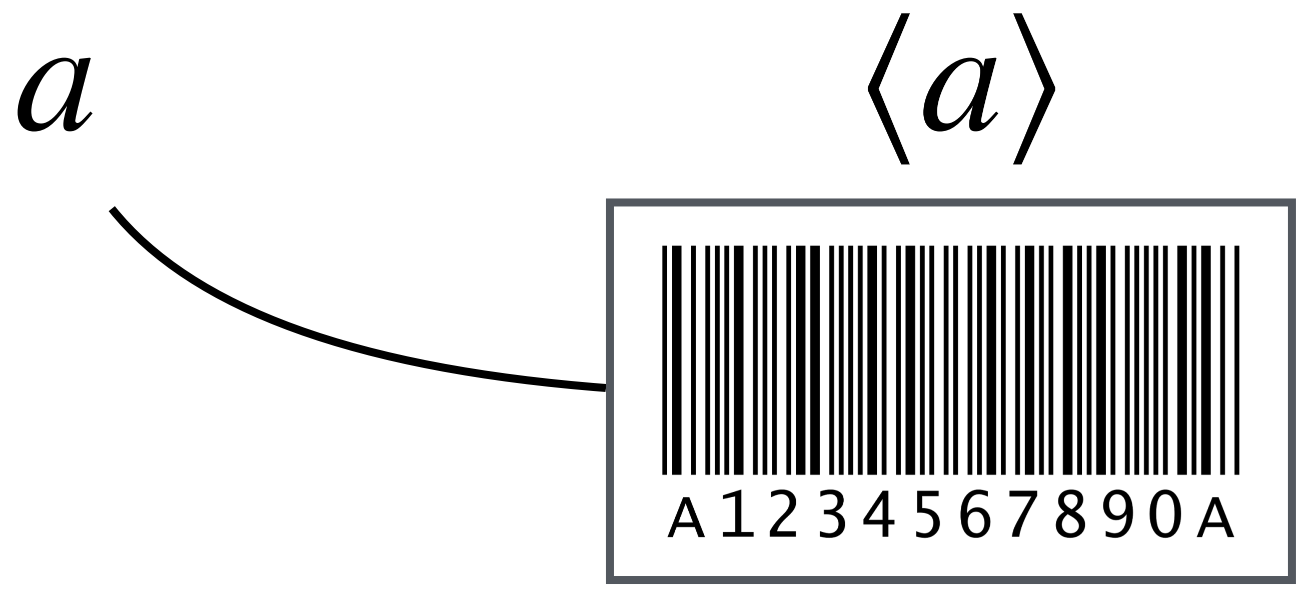
Does every object have a corresponding encoding?

Can two objects have the same encoding?

Does every string correspond to a valid encoding?

encoding: given a set A of objects, an encoding of elements of A is an **injective function** $\mathrm{Enc}\colon A \to \Sigma^*$.

For $a \in A$, $\langle a \rangle$ denotes $\mathrm{Enc}(a)$.



Warning: not all sets are encodable.

Examples $A = \mathbb{N}$

$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$\langle 36 \rangle =$ "36"

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

$\langle 36 \rangle =$ "100100"

$\Sigma = \{1\}$

$\langle 36 \rangle =$ "111111111111111111111111111111111111"

? Does Σ affect encodability?

Examples $A = \mathbb{Z}$

$\Sigma = \{-, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$\langle -36 \rangle =$ "-36"

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

$\langle -36 \rangle =$ "1100100"

$\Sigma = \{1\}$?

Examples $A = \mathbb{N} \times \mathbb{N}$

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \#\}$$

$$\langle (3, 36) \rangle = \langle 3, 36 \rangle = \text{``3\#36''}$$

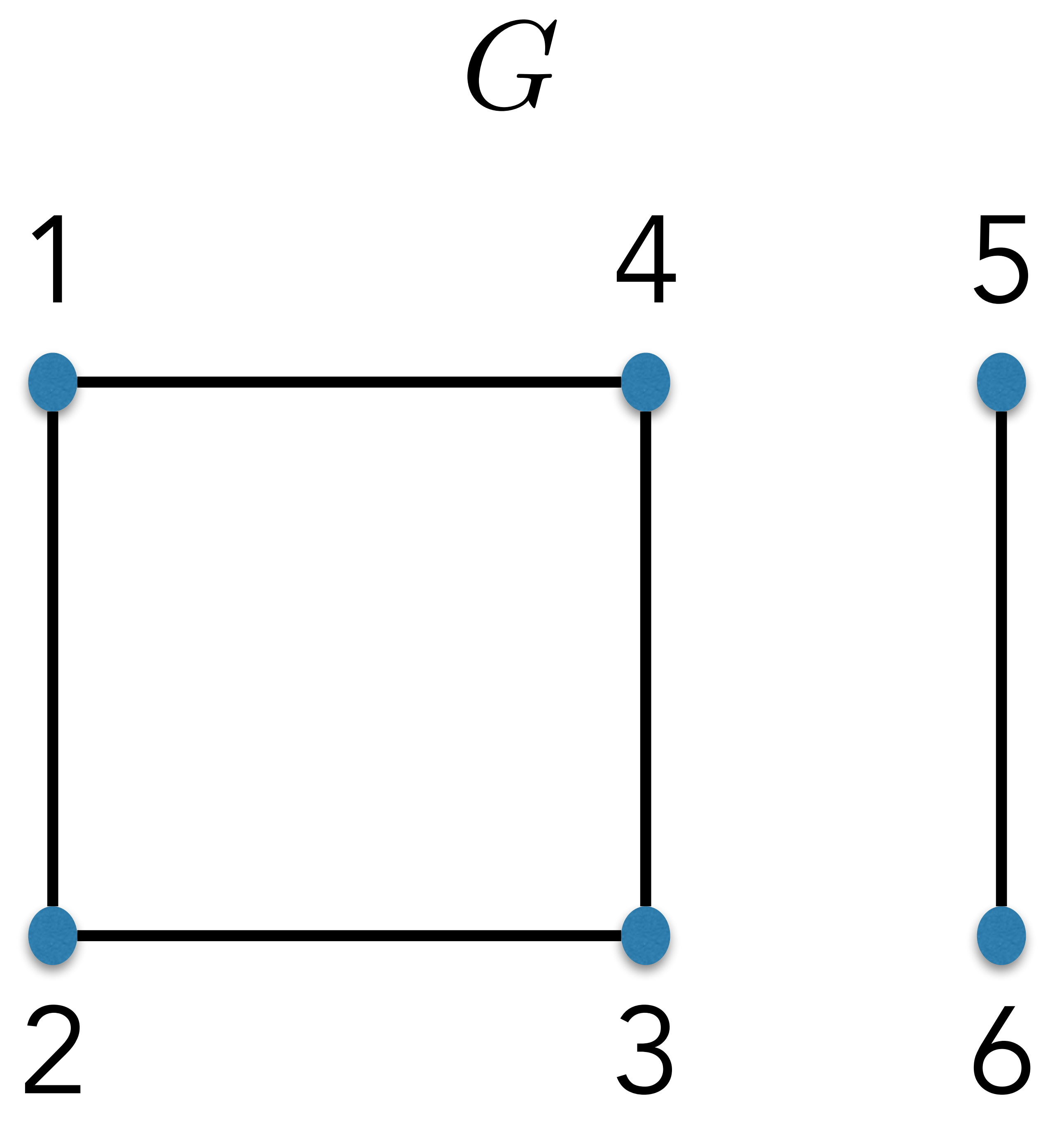
$$\Sigma = \{0, 1\}$$

Idea: encode all symbols above using 4 bits (why 4?)

$0 \rightarrow 0000$	$4 \rightarrow 0100$	$8 \rightarrow 1000$
$1 \rightarrow 0001$	$5 \rightarrow 0101$	$9 \rightarrow 1001$
$2 \rightarrow 0010$	$6 \rightarrow 0110$	$\# \rightarrow 1010$
$3 \rightarrow 0011$	$7 \rightarrow 0111$	

$$\langle 3, 36 \rangle = \text{``001110100011 0110''}$$

Examples $A =$ all undirected graphs



$\langle G \rangle =$ "V = {1, 2, 3, 4, 5, 6}
E = {{1,2}, {2,3}, {3,4}, {1,4}, {5,6}}"

Examples $A =$ all Python functions

```
def isPrime(N):
  if (N < 2):
    return False
  for factor in range(2, N):
    if (N % factor == 0):
      return False
  return True
```

$\langle \text{isPrime} \rangle =$

“def isPrime(N):\n if (N < 2):\n   return False\n for factor in range(2, N):\n   if (N % factor == 0):\n   return False\n return True”

Does $|\Sigma|$ matter?

Going from $|\Sigma| = k$ to $|\Sigma'| = 2$:

encode every symbol of Σ using t bits,

where $t = \lceil \log_2 k \rceil$.

A word of length n over Σ	→	A word of length tn over Σ'

Does $|\Sigma|$ matter?

$A = \mathbb{N}$ **Binary vs Unary**

	Binary	Unary
0	0	ϵ
1	1	1
2	10	11
3	11	111
4	100	1111
5	101	11111
6	110	111111
7	111	1111111
8	1000	11111111
9	1001	111111111
10	1010	1111111111
11	1011	11111111111
12	1100	111111111111

Does $|\Sigma|$ matter?

Binary vs Unary

n has length $\lfloor \log_2 n \rfloor + 1$ in **binary**

n has length $\lfloor \log_k n \rfloor + 1$ in **base** k

n has length n in **unary**

$$\log_k n = \frac{\log_2 n}{\log_2 k}$$

Unary is exponentially longer than other bases!

Which sets are encodable?

Encodability = Countability

(will see this later)

What about uncountable sets?

Approximate.

Summary So Far

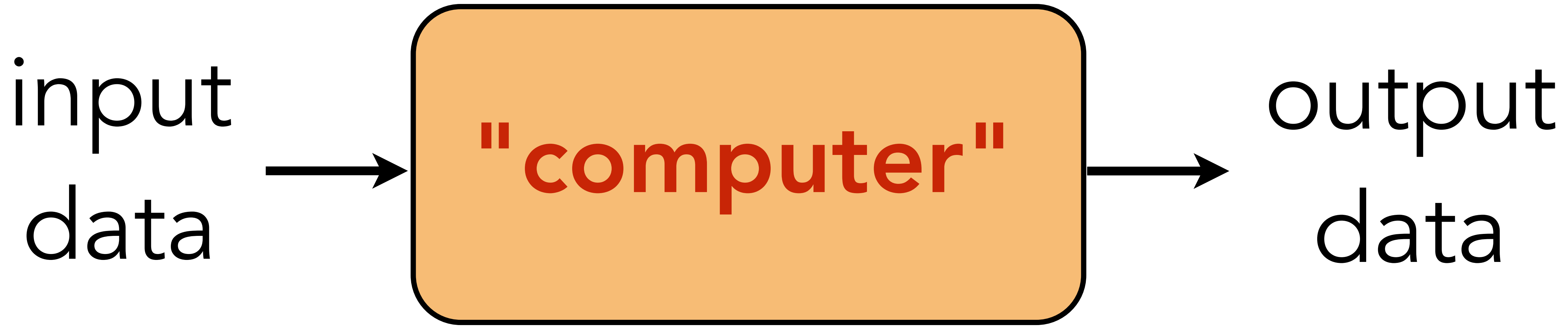
Alphabet Σ, String/word, Σ^*

Encoding of a set A: injective function $\text{Enc} : A \to \Sigma^*$.

Encodable = Countable

Alphabet doesn't matter much as long as $|\Sigma| > 1$.

Next Few Chapters

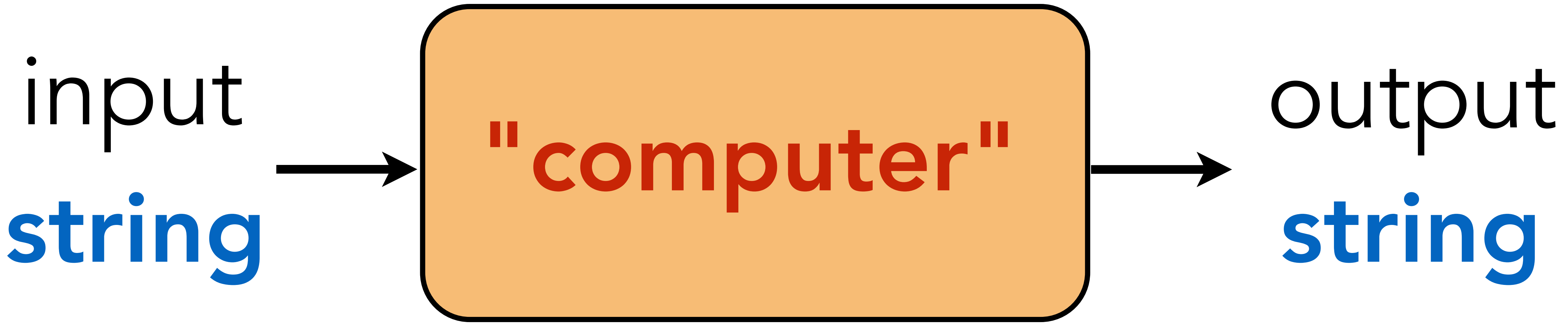


What is **computation**?

What is an **algorithm**?

How can we mathematically define them?

Next Few Chapters

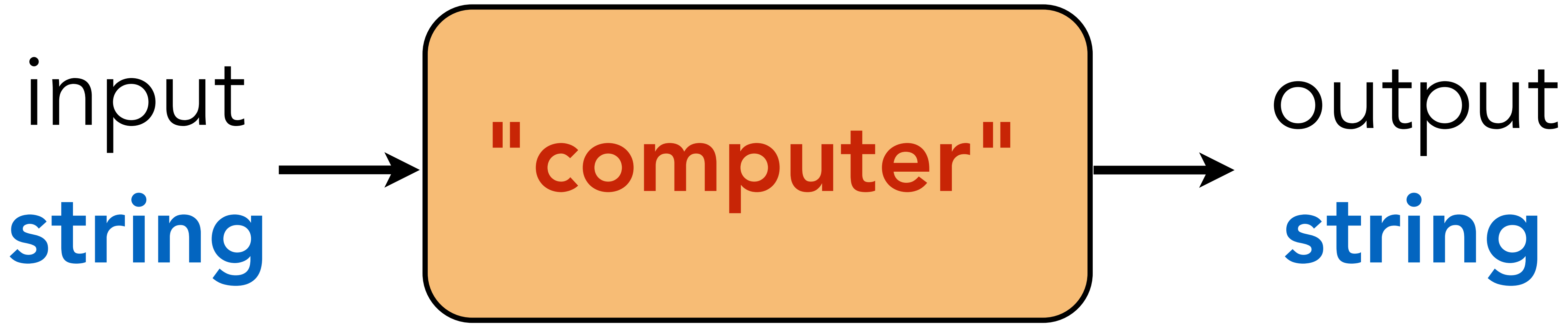


What is **computation**?

What is an **algorithm**?

How can we mathematically define them?

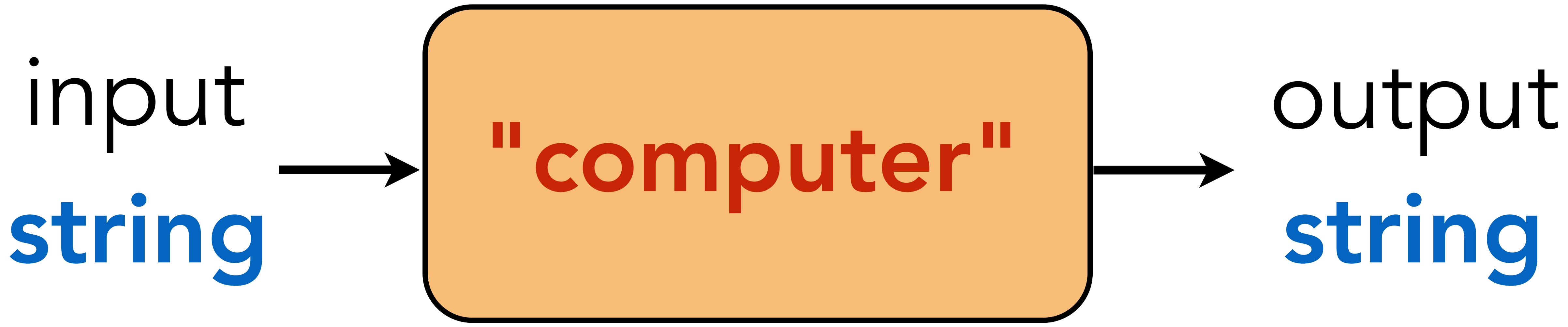
Let's lay the groundwork...



Reasonable assumptions to start with:

- Computer is *deterministic*
- Computation is a finite process
- Input can be any finite-length string
- For all inputs, there is an output
- Output is a finite-length string

? How can we characterize
the **input/output behavior** of a computer?



Is computer just a function $f : \Sigma^* \to \Sigma^*$ **???**

Function problem:
A function of the form $f : \Sigma^* \to \Sigma^*$.

A computer/algorithm **solves** function problem f
if its input/output behavior corresponds to f.

Function Problem Examples

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

Reverse function

$110100 \mapsto 001011$

Sort function

$110100 \mapsto 000111$

isPrime

$11111010 \mapsto 0$

$11111011 \mapsto 1$

Example description of a **function problem**:

Given a natural number N, output *True* if N is prime, and output *False* otherwise.

Input type: natural number

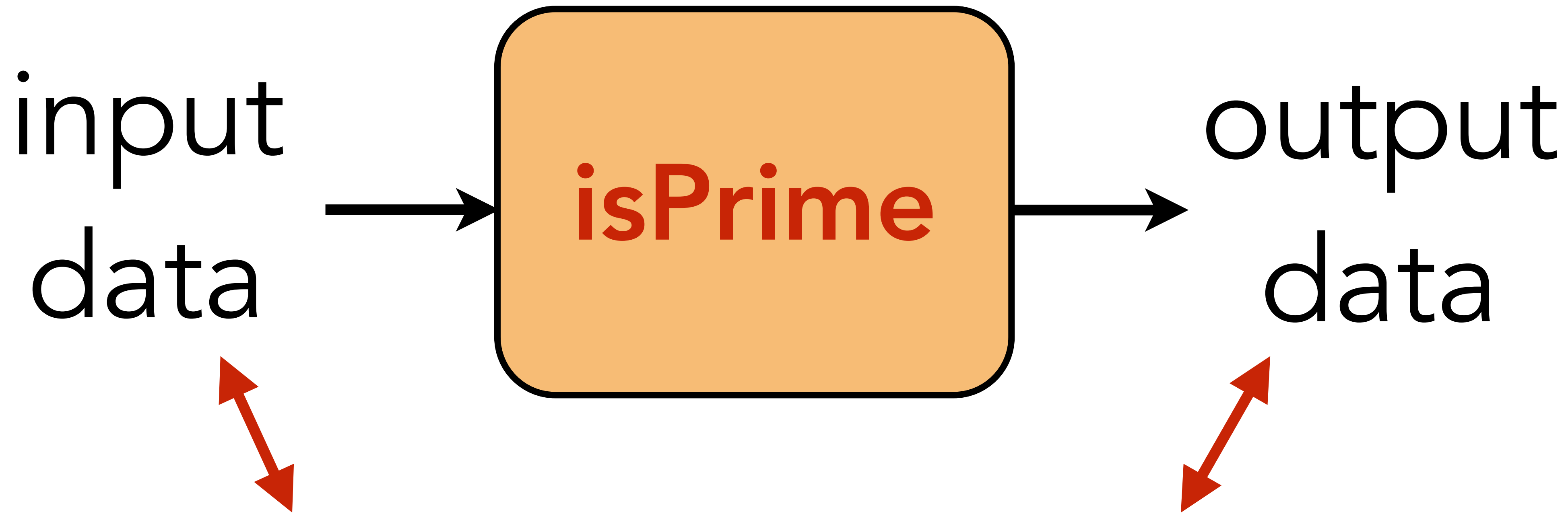
Output type: boolean

A **function problem** is a function

$$f : I \to S\ .$$

$I =$ set of possible input objects (called **instances**)

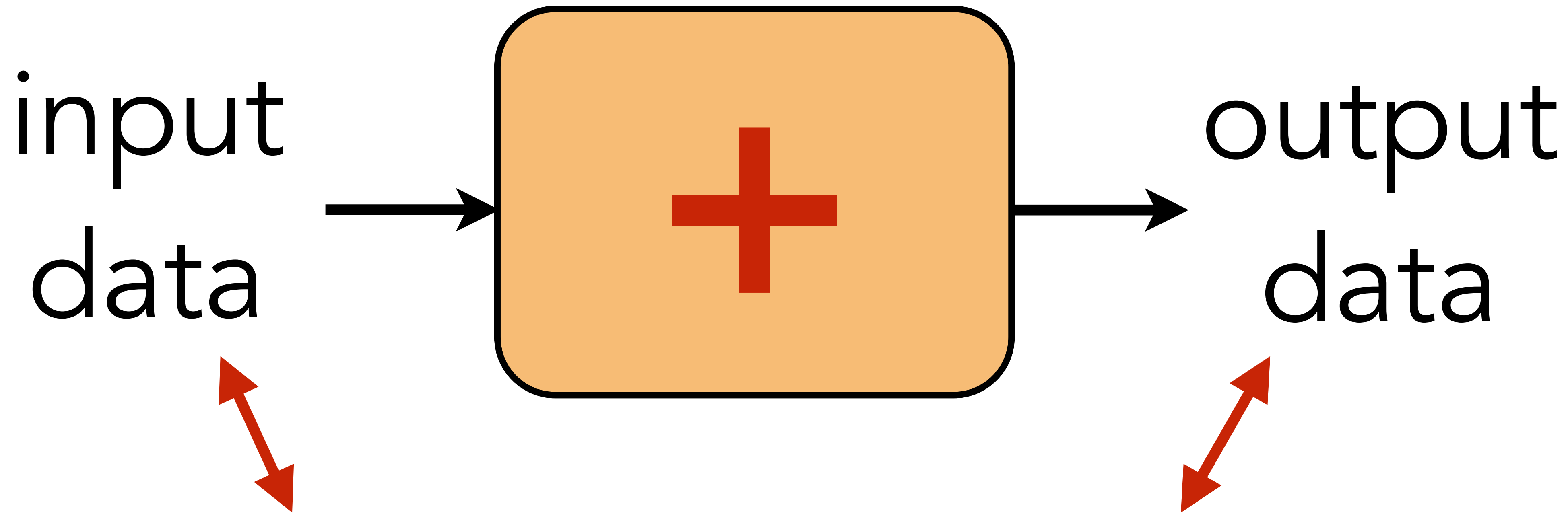
$S =$ set of possible output objects (called **solutions**)



$I = \mathbb{N}$

Instance	Solution
0	No
1	No
2	Yes
3	Yes
4	No
⋮	⋮
251	Yes
⋮	⋮

$S = \{\text{No}, \text{Yes}\}$



$I = \mathbb{N} \times \mathbb{N}$

Instance	Solution
0, 0	0
0, 1	1
1, 1	2
2, 2	4
2, 3	5
10, 1	11
100, 99	199
⋮	⋮

$S = \mathbb{N}$

Instance

["vanilla", "mind", "Anupam", "yogurt", "doesn't"]

Solution

["Anupam", "doesn't", "mind", "vanilla", "yogurt"]

A **function problem** is a function

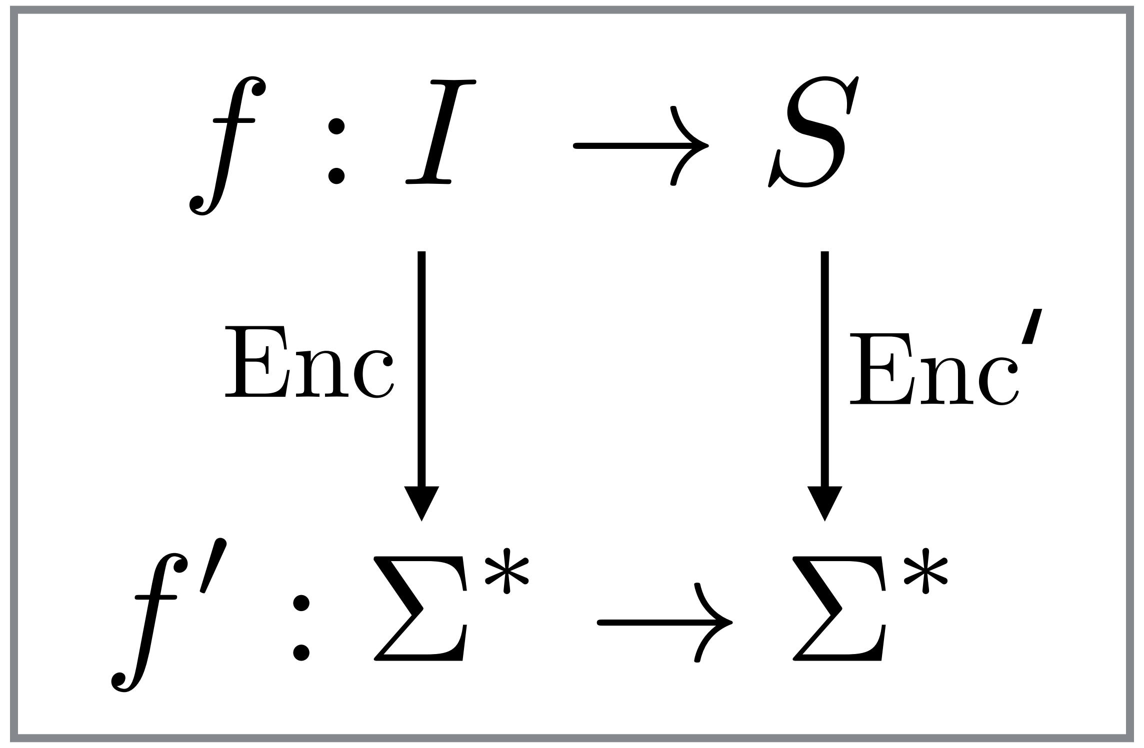
$$f : I \to S \,.$$

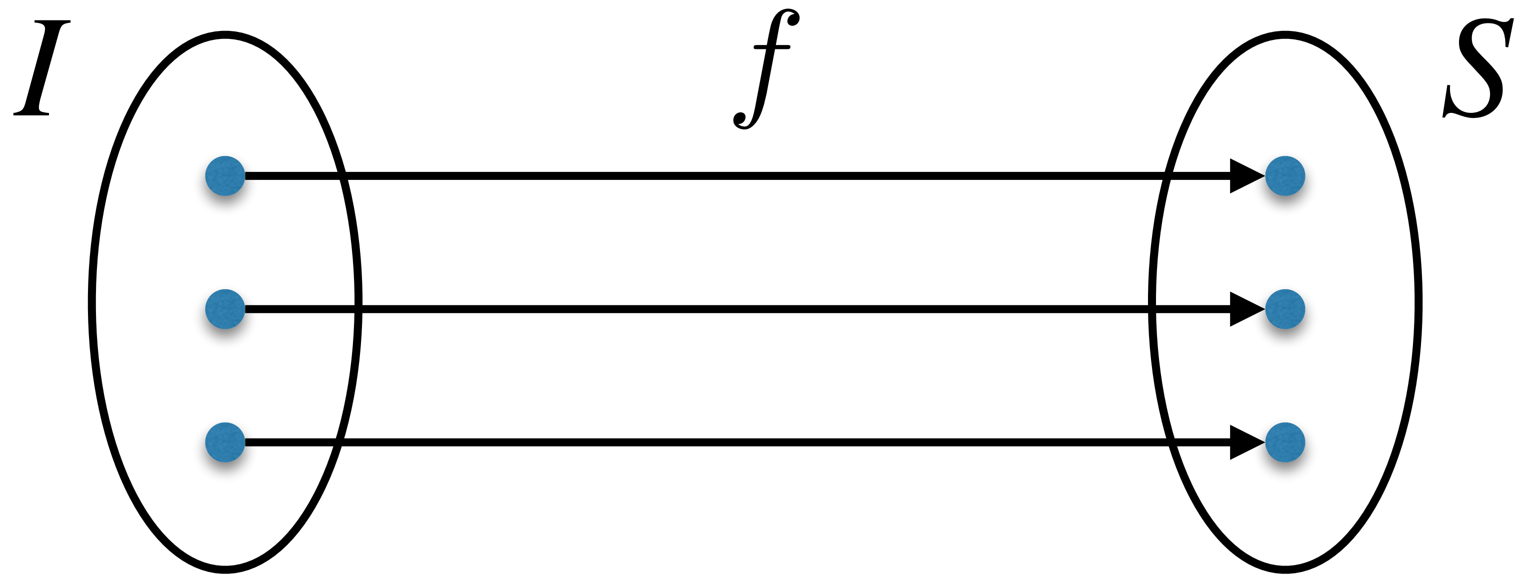
$I =$ set of possible input objects (called **instances**)

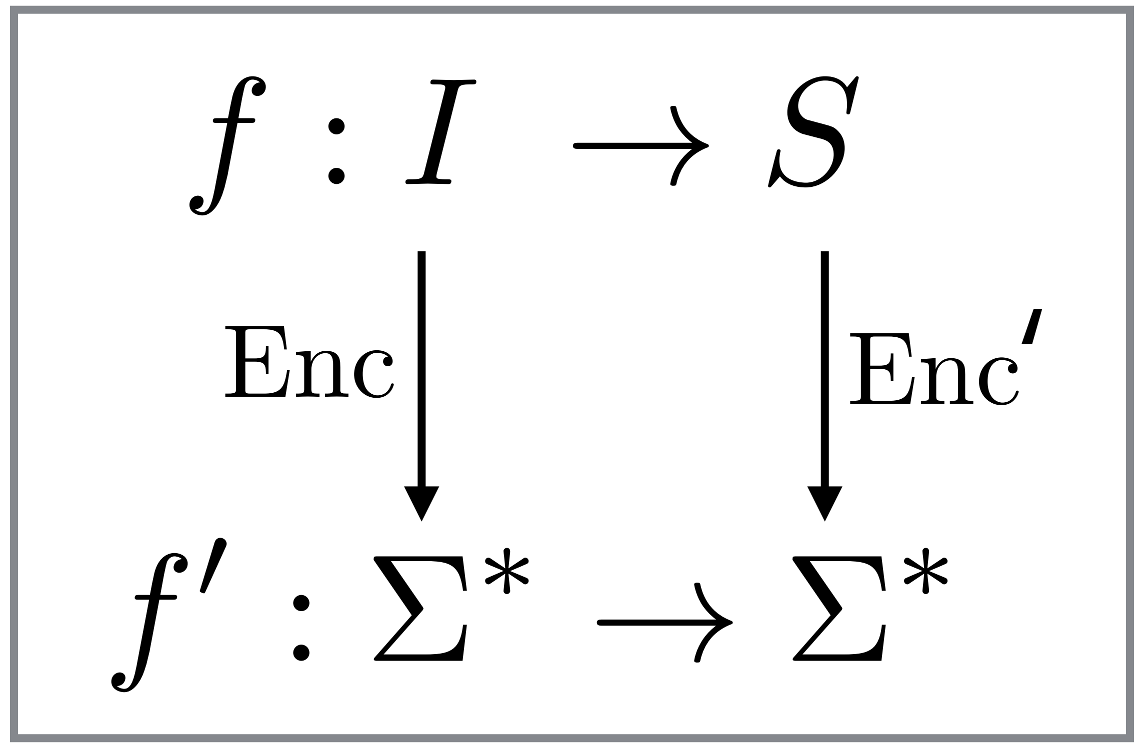
$S =$ set of possible output objects (called **solutions**)

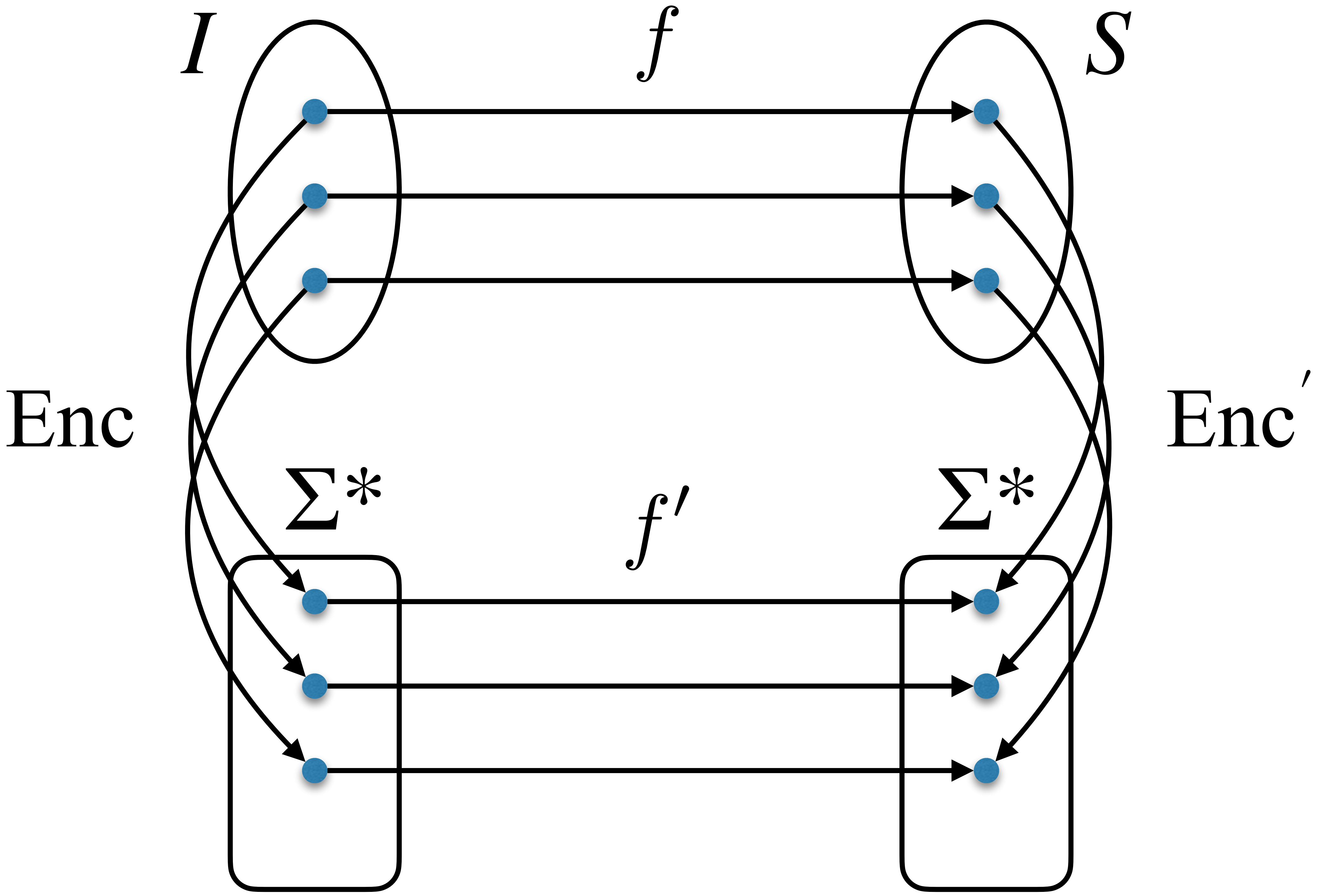
In TCS, we don't deal with arbitrary types,
we deal with **strings** (encodings).

$$\begin{array}{ccc} f : I & \to & S \\ \text{Enc}\downarrow & & \downarrow\text{Enc}' \\ f' : \Sigma^* & \to & \Sigma^* \end{array}$$


f : I → S
Enc
Enc′
f′ : Σ* → Σ*


I
f
S





Technicality: What if $w \in \Sigma^*$ does not correspond to an encoding of an instance?

In TCS, there is only one type of data:

string

A convenient restriction:

Problems with 2 possible solutions.

Example: isPrime

Decision problem:

A function of the form $f : \Sigma^* \rightarrow \{0,1\}$.

$\{\text{False, True}\}$

$\{\text{Reject, Accept}\}$

$\{\text{No, Yes}\}$

Why?

1. Simpler objects
2. Without loss of generality

1. Simpler objects

Language:

Any set L of finite-length strings over an alphabet Σ.

i.e. any set $L \subseteq \Sigma^*$.

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

$L = \varnothing$

$L = \Sigma^*$

$L = \{0,\ 1,\ 00,\ 11\}$

$L = \{\epsilon,\ 01,\ 0011,\ 000111,\ \ldots\}$

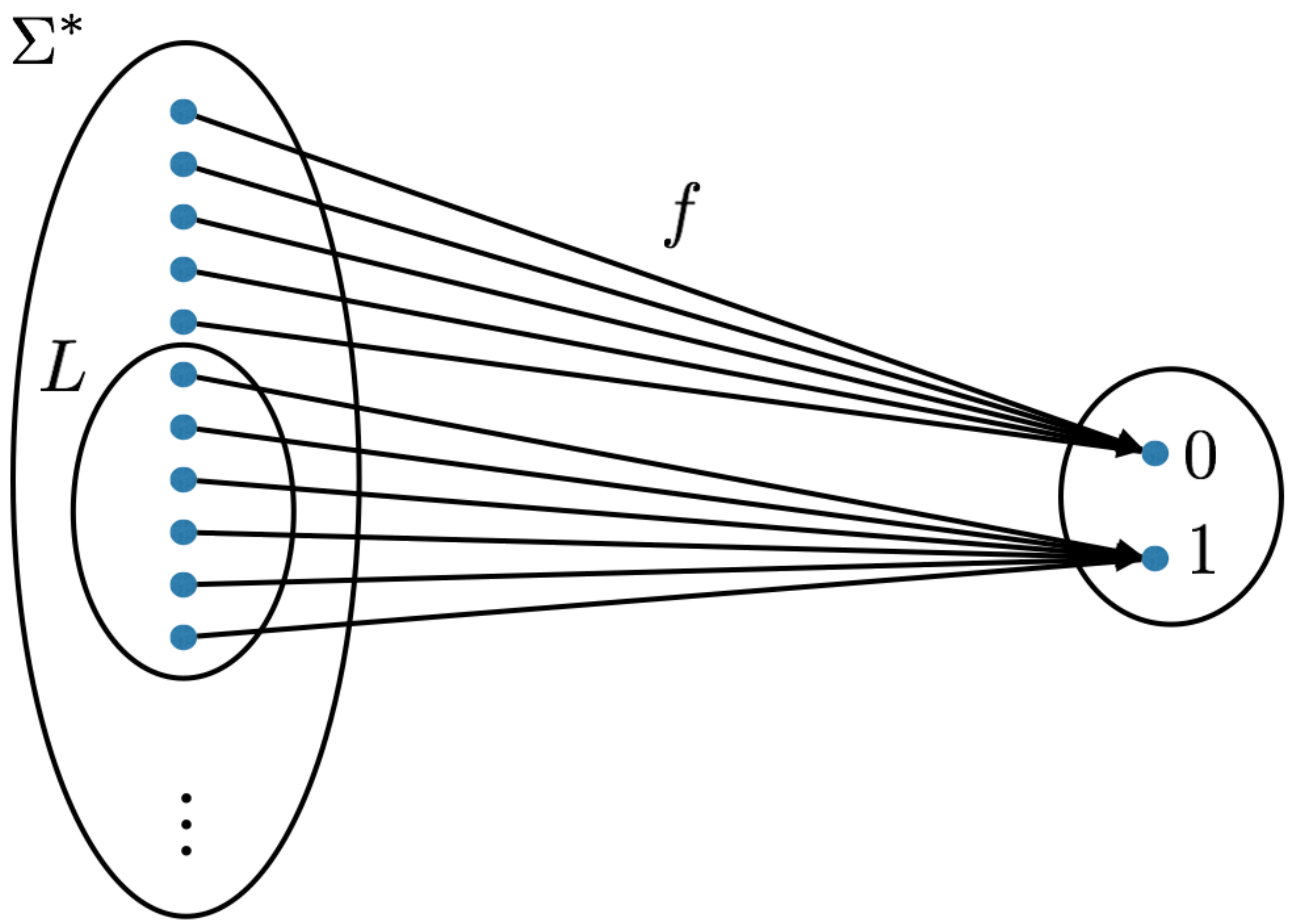
1. Simpler objects

There is a one-to-one correspondence between **decision problems** and **languages**.

$$f : \Sigma^* \rightarrow \{0,1\}$$

Instance	Solution
ϵ	1
0	1
1	1
00	1
01	0
10	0
11	1
000	1
001	0
$\vdots$	$\vdots$

$$L = \{\epsilon, 0, 1, 00, 11, 000, \ \ldots\}$$



2. Without loss of generality

function problem ≈ corresponding decision problem

Integer factorization problem:

Input: natural number N,

Output: prime factorization of N.

Decision version:

Input: natural numbers N and k,

Output: True iff N has a factor between 2 and k?

2. Without loss of generality

function problem $\approx$ corresponding decision problem

Integer factorization problem:

Input: natural number N,

Output: prime factorization of N.

Smallest factor problem:

Input: natural number N,

Output: smallest (prime) factor of N.

Decision version:

Input: natural numbers N and k,

Output: True iff N has a factor between 2 and k?

- Are all languages **computable/solvable**?
- How can you prove a language is not **solvable**?
- How do we measure the **complexity** of algorithms solving languages?
- How do we classify languages according to the resources needed to solve them?
- $P \stackrel{?}{=} NP$