While we wait... (Class starts 6:10)

Meet the people sitting next to you if you don't already know them!

CSC 236 Lecture 1: Welcome to CS Theory

Harry Sha

May 10, 2023

Today

Communication

I will make all announcements through Ed (which will send you an email notification).

You can find all course material on the course website:

`https://www.cs.toronto.edu/~shaharry/csc236/`

Syllabus

All the course policies can be found on the syllabus, so make sure you read it carefully!

Prerequisites

CSC165 (or equivalent)

- Sets
- Graphs
- Proofs
- Mathematical Logic
- Asymptotics

Here are some course notes by David Liu and Toniann Pitassi if you'd like to review something!

Grading Breakdown

1.) TA check-ins: 10% ($5 \times 2\%$)
2.) Midterm: 40%
3.) Final: 50%
4.) Ed Contributor Prize 2%

HW + Check Ins (10%)

- There will be 5 HWs throughout the semester and one check in per HW.
- You will get full credit for check ins if you manage to convince your TAs that you made an honest attempt at trying to solve the HW problems.
- See the guide to hw, and guide to check ins for more information and tips for the HW and the check ins!

Exams (40% + 50%)

- At least 20% of the exam with be based on the HW problems.
- The midterm will June 28, 6-9PM at EX100.
- The final will be sometime August 17 - 25.

Support

Sign up for Ed! Please ask and answer questions!

Come to office hours!

The TAs and I are here for you, so please don't hesitate to reach out. The best way to get in touch is through Ed.

Course overview

Part 1: Mathematical tools.

Part 2: Algorithm correctness and runtime.

Part 3: Finite automata.

Welcome!

Questions we care about in Theory

Is it possible to . . . .

solve problem X
send secure messages
guarantee privacy
convey information efficiently and robustly
have "fair" machine learning algorithms
reach collective decisions to maximize "social welfare"
play a game optimally

⋮

Questions we care about in Theory

Is it possible to . . . .

solve problem X — Computability theory

send secure messages — cryptography

guarantee privacy — differential privacy

convey information efficiently and robustly — coding theory

have "fair" machine learning algorithms — algorithmic fairness

reach collective decisions to maximize "social welfare" — social choice theory

play a game optimally — game theory

⋮

Questions we care about in Theory

If so, how? Also, how much/many

$$\left\{ \begin{array}{l} \text{time} \\ \text{space} \\ \text{randomness} \\ \text{processors} \\ \text{quantum bits} \\ \text{circuit gates} \\ \vdots \end{array} \right\}$$

do we need?

Questions we care about in Theory

If so, how? Also, how much/many

algorithm design

time

space

randomness

processors

quantum bits

circuit gates

⋮

parallel and distributed computing

quantum computation

circuit complexity

do we need?

Complexity theory

The sell

For programmers:

- Rigorously analyze your programs.
- Model real world problems as well studied problems in theory and apply known algorithms.

The sell

For programmers:

- Rigorously analyze your programs.
- Model real world problems as well studied problems in theory and apply known algorithms.

In general:

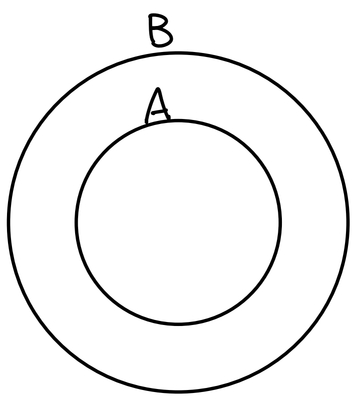
- There are no compiler issues in Theory Land: problems are distilled to the core puzzle.
- CS Theory is a fascinating and new field! There are lots of unknowns, and breakthroughs happen very often.

Are there problems computers can't solve?

Set theory review

"subset" (\subseteq)

$A \subseteq B, \quad A \subseteq B$

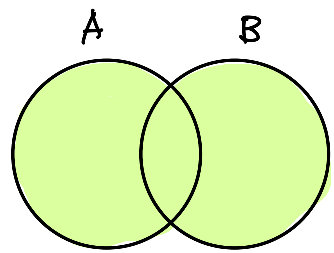


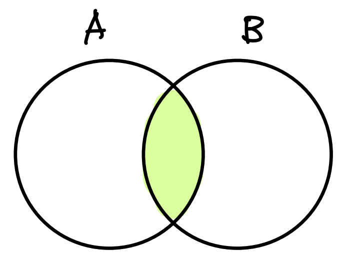
$\forall x. (x \in A \Rightarrow x \in B)$

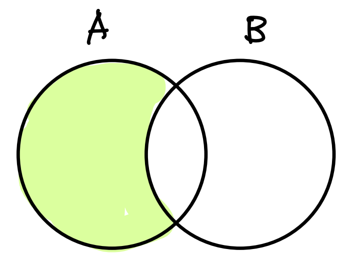
"union" (\cup) $A \cup B$

"intersection" (\cap) $A \cap B$

"difference" (\setminus) $A \setminus B$







"emptyset"

$\{\} = \emptyset$ (\emptyset)

$\wp(A) = \{B : B \subseteq A\}$

"powerset of A" (\wp)

i.e. $\wp(A)$ is the set containing all the subsets of A

$$\wp(\{1,2\}) = \{\{1\}, \{2\}, \{\}, \{1,2\}\}$$

Set theory review

$$A \times B = \{(a,b) : a \in A,\ b \in B\}$$

↑ " A times B " (\times)

$$A^n = \underbrace{A \times A \times \cdots \times A}_{n} = \{(a_1, a_2, \ldots, a_n) : \forall i \in [n],\ a_i \in A\}.$$

" A to the n "

Functions

$$\underset{\text{"domain"}}{\overset{\text{sets}}{f : A \to B}}$$

"f is a function from A to B"

"domain" "codomain"

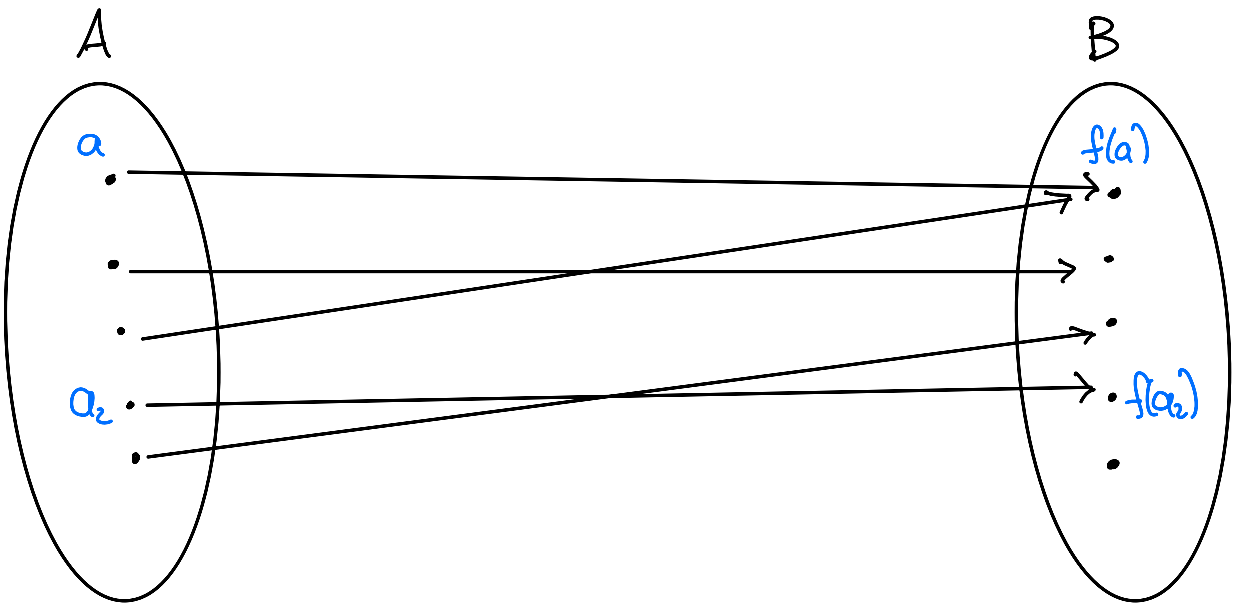
sets

Think of the domain as a set of inputs and the codomain as a set of outputs.

$$\boxed{\forall a \in A . \left(f(a) \in B \right)}$$

Functions

$$f : A \to B$$

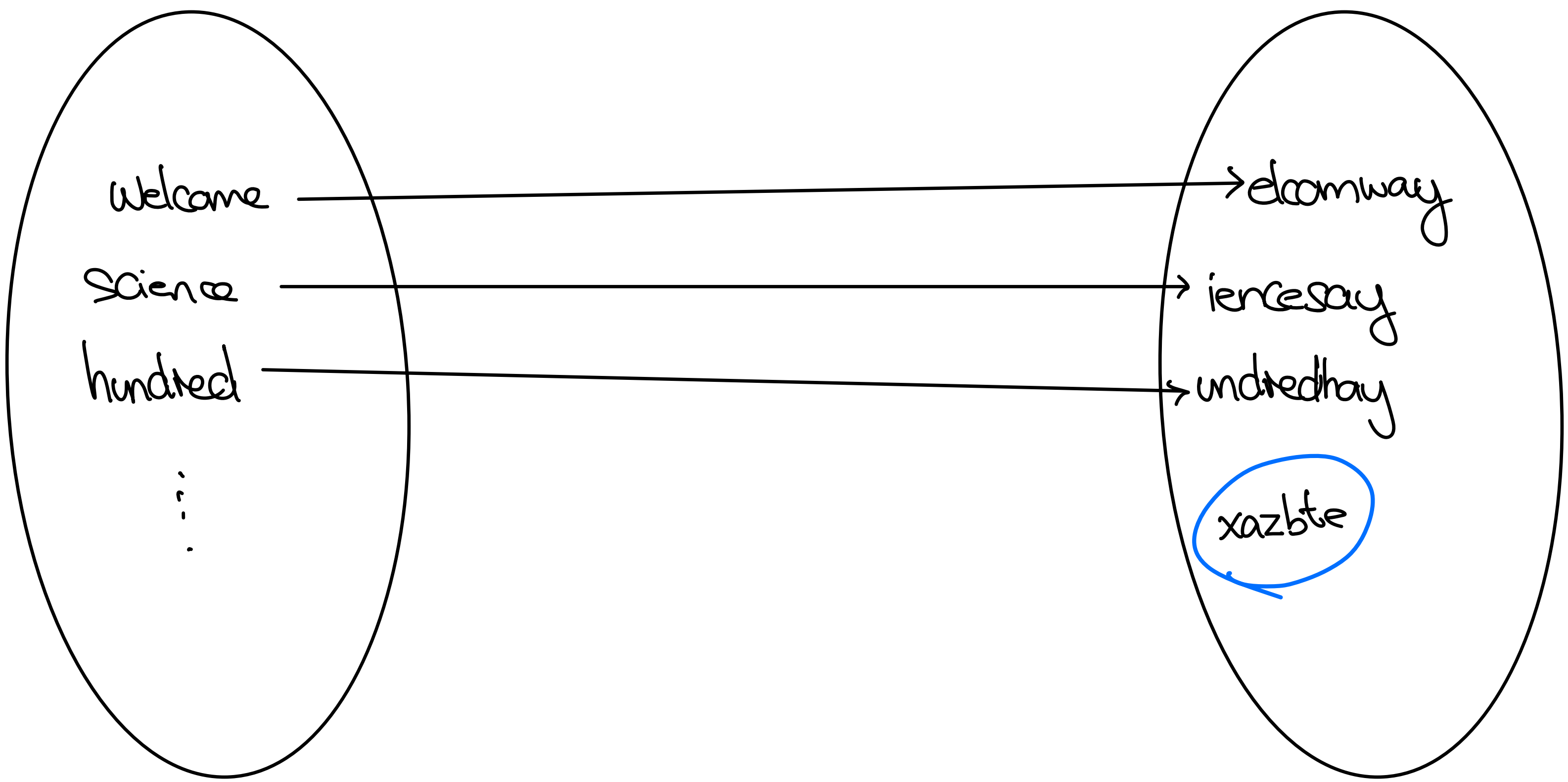


Everything in A is mapped to something in B.

Not everything in B needs to be mapped to.

Examples

PigLatin: English $\longrightarrow$ Strings



Examples

```python
def product(xs: List[int]) -> int:
    accumulator = 1
    for x in xs:
        accumulator *= x
    return accumulator
```

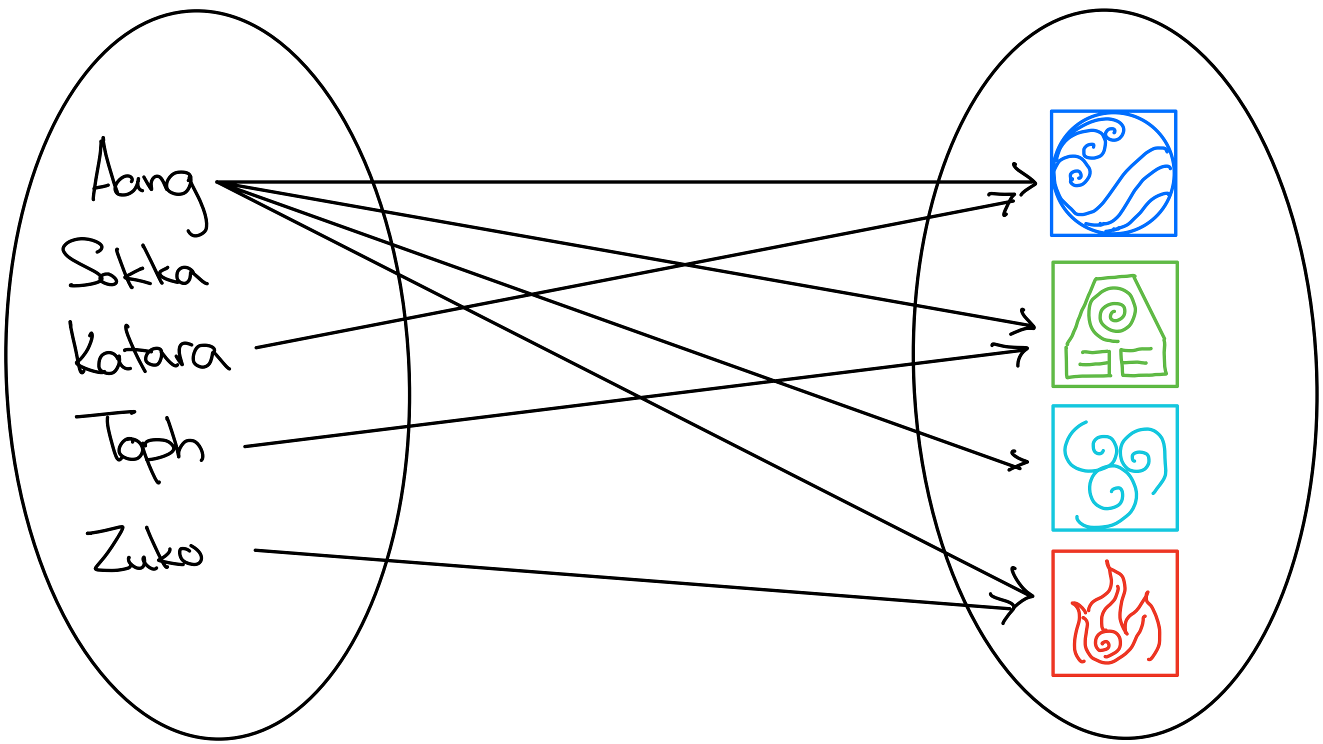
What is the domain/codomain here?

set of inputs $\{l : l \text{ is a list of integers}\}$.

codomain $= \mathbb{R}$, $\mathbb{Z}$

Is this a function?

Bender : Gaang → Elements.

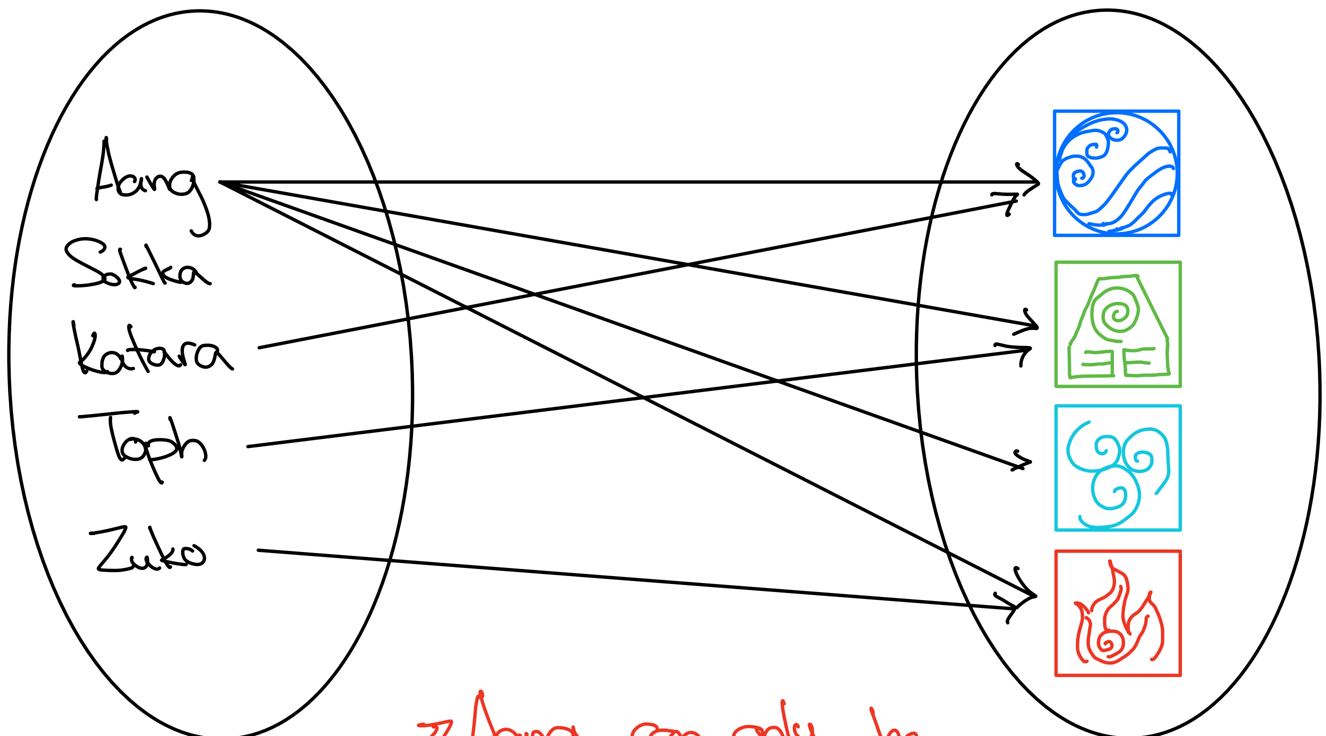


1.) Aang can't be mapped to more than 1 thing!

2.) Sokka need to be mapped to something.

Is this a function?

Bender : Gaang $\to$ Elements.



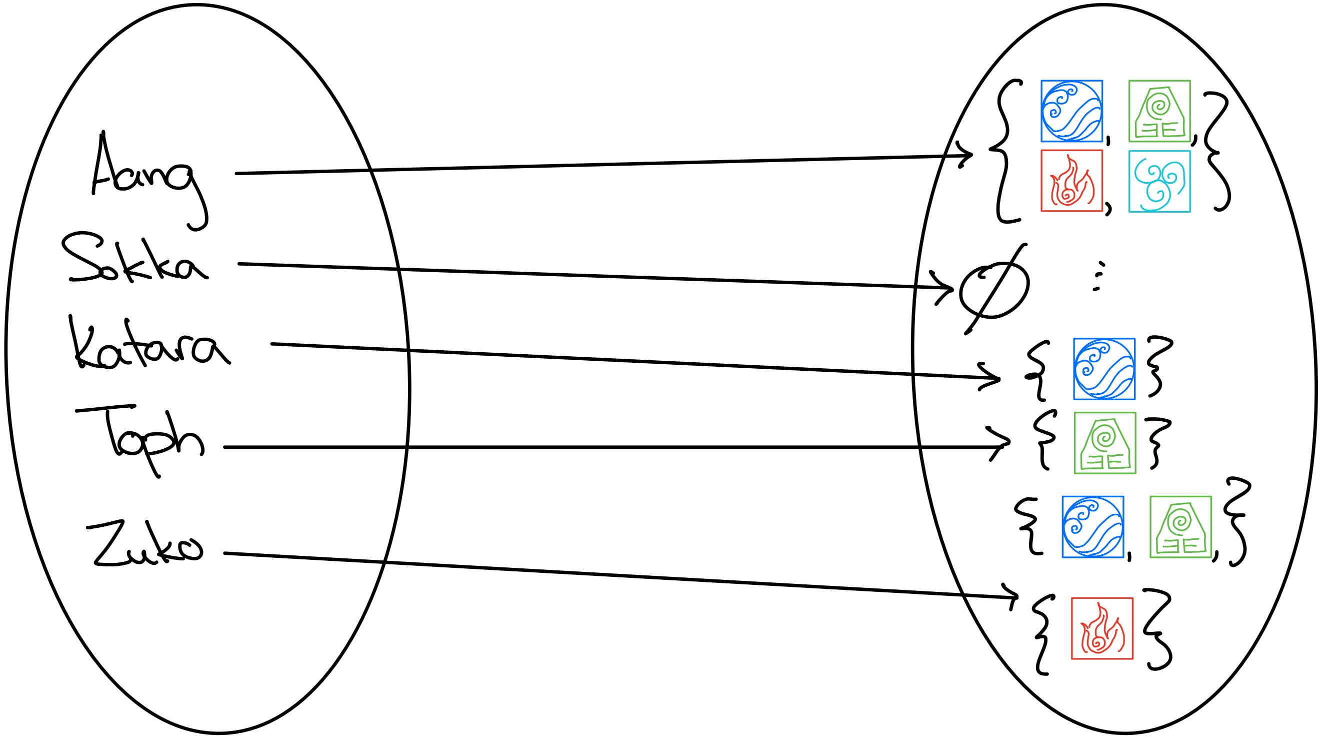
Not a function!

Aang can only be mapped to one thing.

Sokka must be mapped to something!

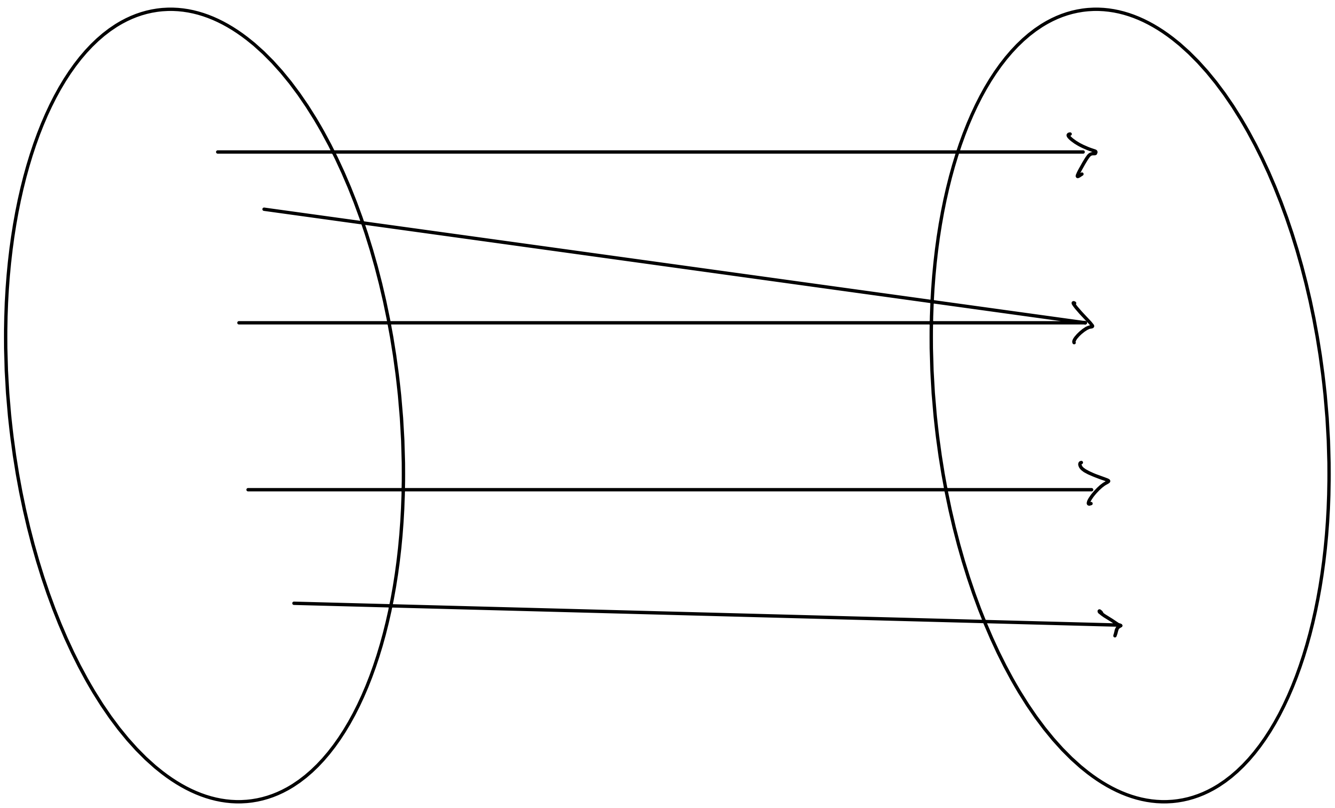
A fix - change the codomain!

Bends : Gaang $\to \mathcal{P}$(Elements)



A function that we're interested in

$$\text{Solves} : \text{Program} \longrightarrow \text{Problems}$$



Injective

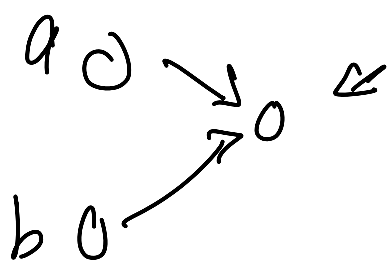
A function is injective if nothing in the codomain is hit more than once. Formally,

Definition (Injective)

A function $f : A \to B$ is injective if

$$\forall x, y \in A.(x \neq y \implies f(x) \neq f(y))$$

"If the inputs are different, the outputs are different"



Injective

A function is injective if nothing in the codomain is hit more than once. Formally,

Definition (Injective)

A function $f : A \to B$ is injective if

$$\forall x, y \in A. (x \neq y \implies f(x) \neq f(y))$$

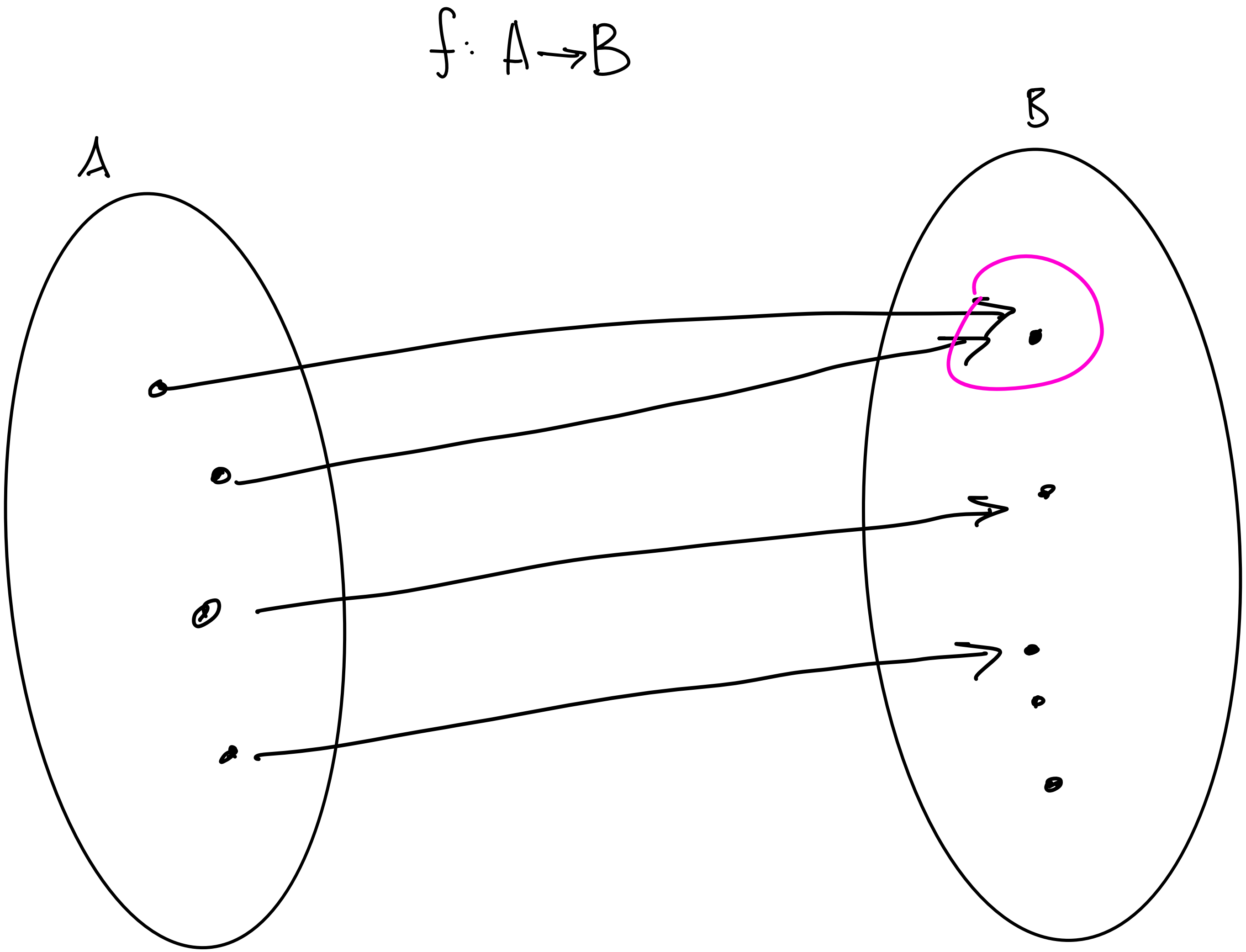
"If the inputs are different, the outputs are different"

Sometimes, the equivalent contrapositive is easier to work with

$$\forall x, y \in A. (f(x) = f(y) \implies x = y)$$

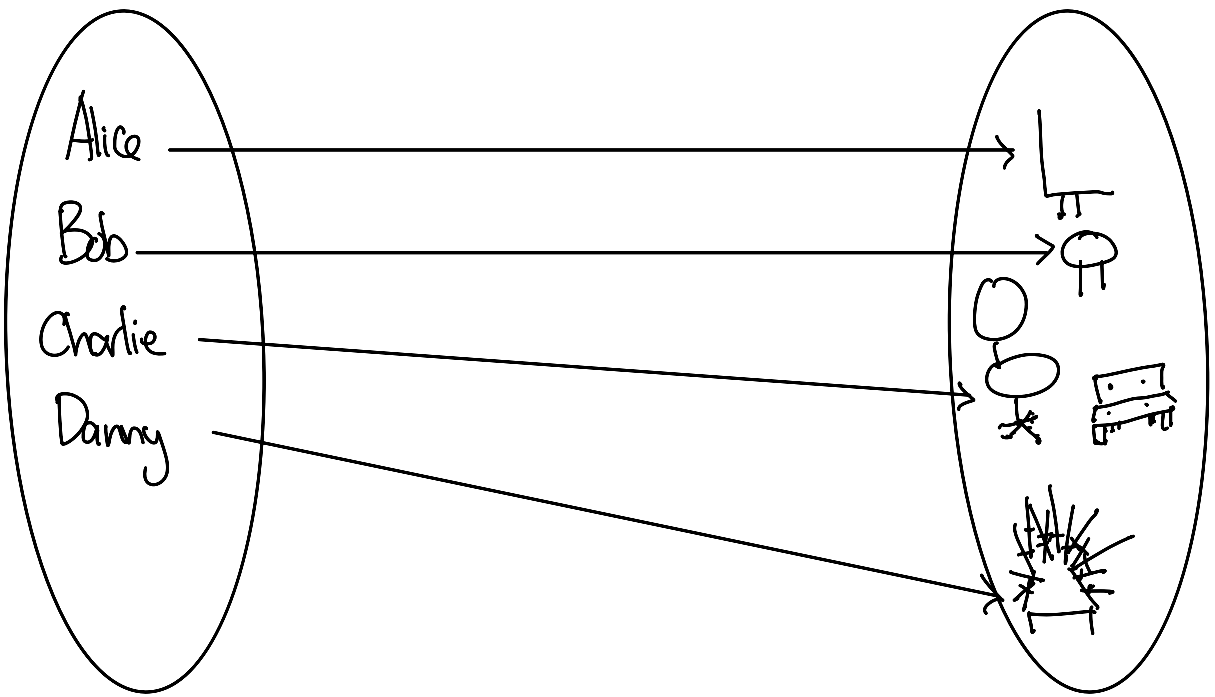
"If the outputs are the same, the inputs are the same"

$f : A \to B$ is injective if
$\forall x, y \in A.(x \neq y \implies f(x) \neq f(y))$



Example - People and Chairs

$f : A \to B$ is injective if
$\forall x, y \in A.(x \neq y \implies f(x) \neq f(y))$

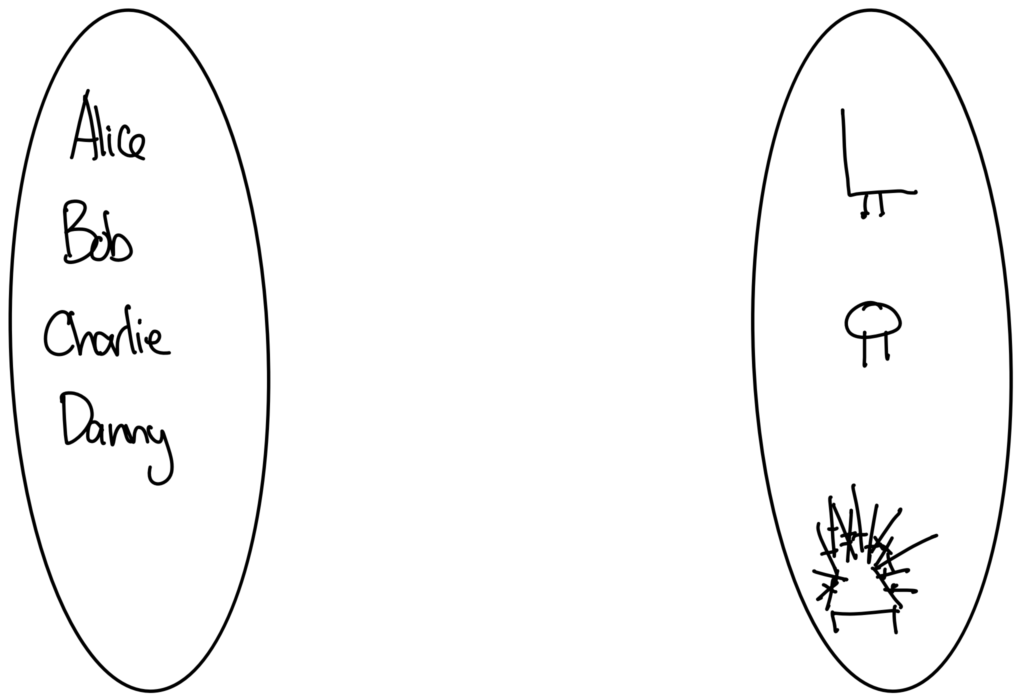


"Different people $\implies$ different chairs"

Example - Musical Chairs

$f : A \to B$ is injective if
$\forall x, y \in A.(x \neq y \implies f(x) \neq f(y))$

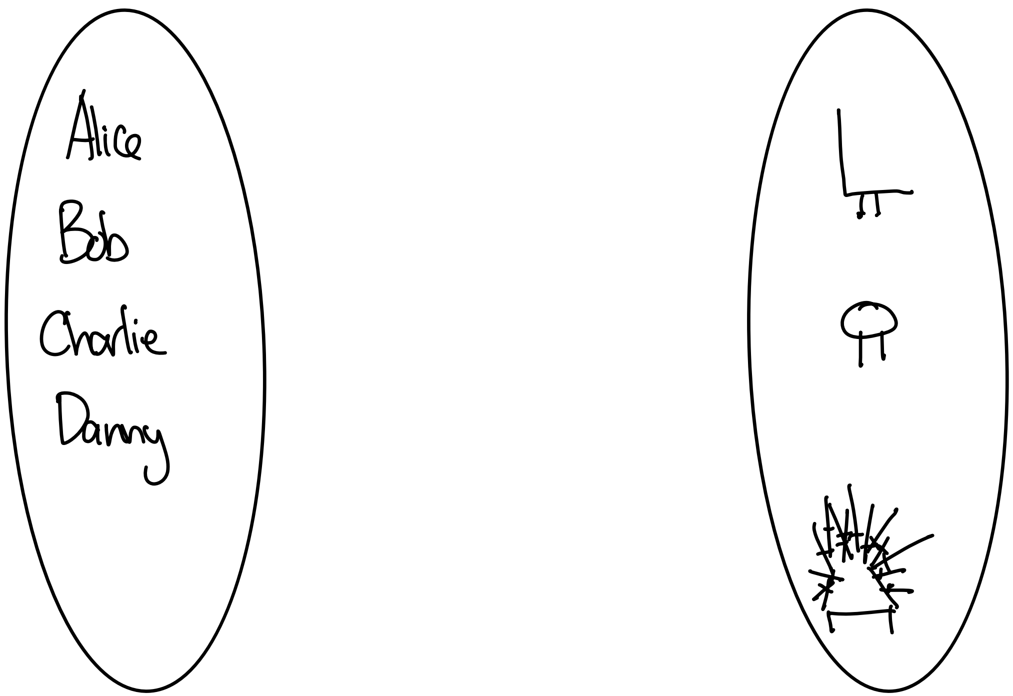
Is there an injective function between these two sets?



Example - Musical Chairs

$f : A \to B$ is injective if
$\forall x, y \in A.(x \neq y \implies f(x) \neq f(y))$

Is there an injective function between these two sets?



No! If there are fewer chairs than people, no matter how you assign people to chairs, at least one person will have to share. I.e., someone goes out after every round of musical chairs. This phenomenon has a special name...

The Pigeonhole Principle

$f : A \to B$ is injective if $\forall x, y \in A.(x \neq y \implies f(x) \neq f(y))$

Theorem (The Pigeonhole Principle)

Let A, B be finite sets where $|A| > |B|$. Then there is no injective function $f : A \to B$.

Think of A as a set of pigeons and B as a set of pigeonholes. The pigeonhole principle is a fancy way of saying that if you have more pigeons than you have pigeonholes, no matter how you assign pigeons to pigeonholes, some pigeonhole will have at least two pigeons.

We will see more of this in tutorial!

Example - Hilbert's Hotel

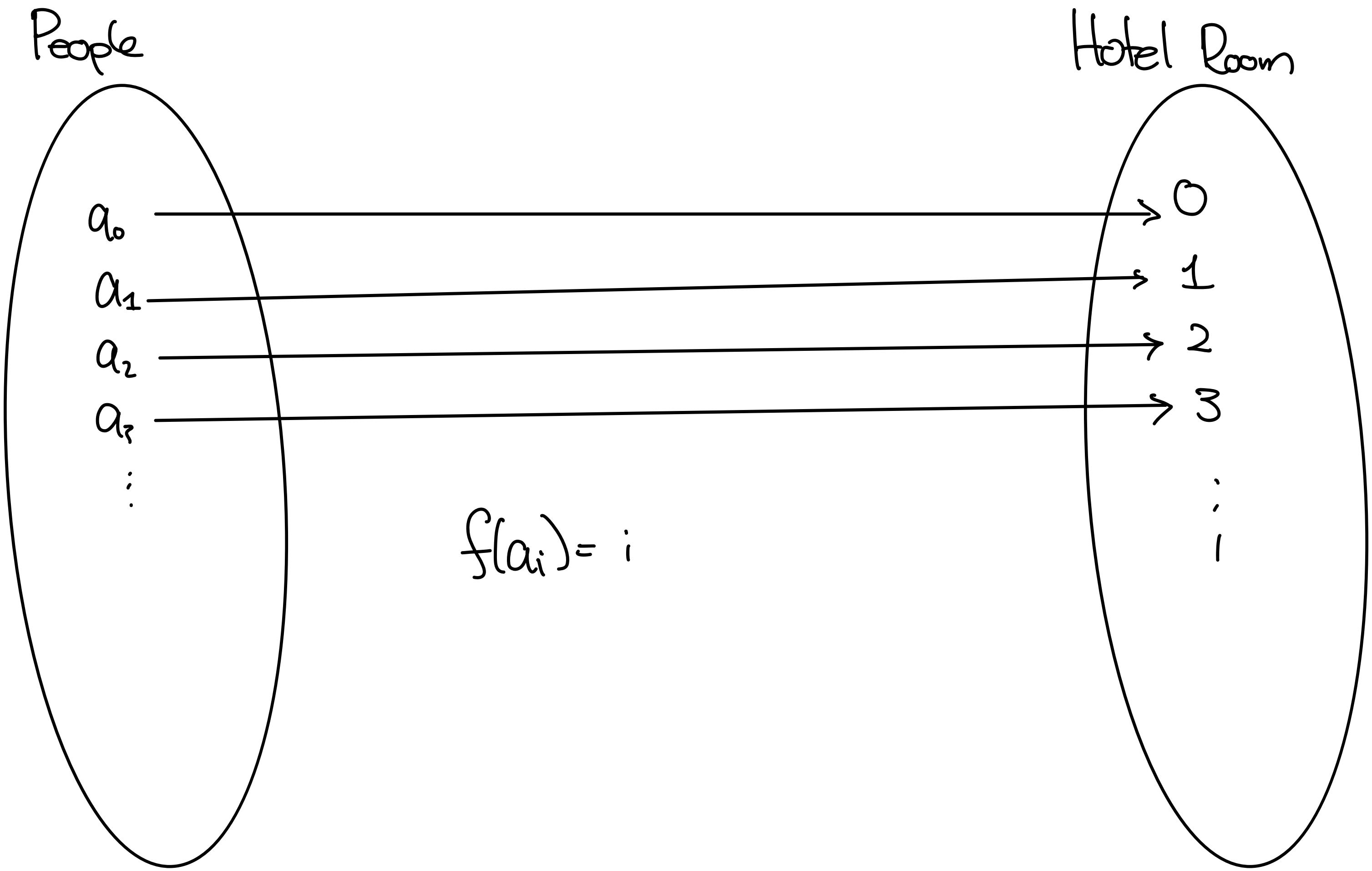
$f : A \to B$ is injective if $\forall x, y \in A.(x \neq y \implies f(x) \neq f(y))$

Imagine you're working at a unique hotel. The hotel has an infinite number of rooms. To be precise, it has one (single person) room for every natural number $0, 1, 2, ...$

Your job is to assign customers to rooms. Just when you thought your job was easy, a bus containing an infinite number of people, let's call them $a_0, a_1, a_2, ...$ shows up and requests rooms. Assume the hotel is empty to start. How do you assign the customers to rooms? Since only one person can stay in each room, we need the assignment to be injective.

Example - Hilbert's Hotel

$f : A \to B$ is injective if $\forall x, y \in A.(x \neq y \implies f(x) \neq f(y))$



Example - Hilbert's Hotel

$f : A \to B$ is injective if $\forall x, y \in A.(x \neq y \implies f(x) \neq f(y))$

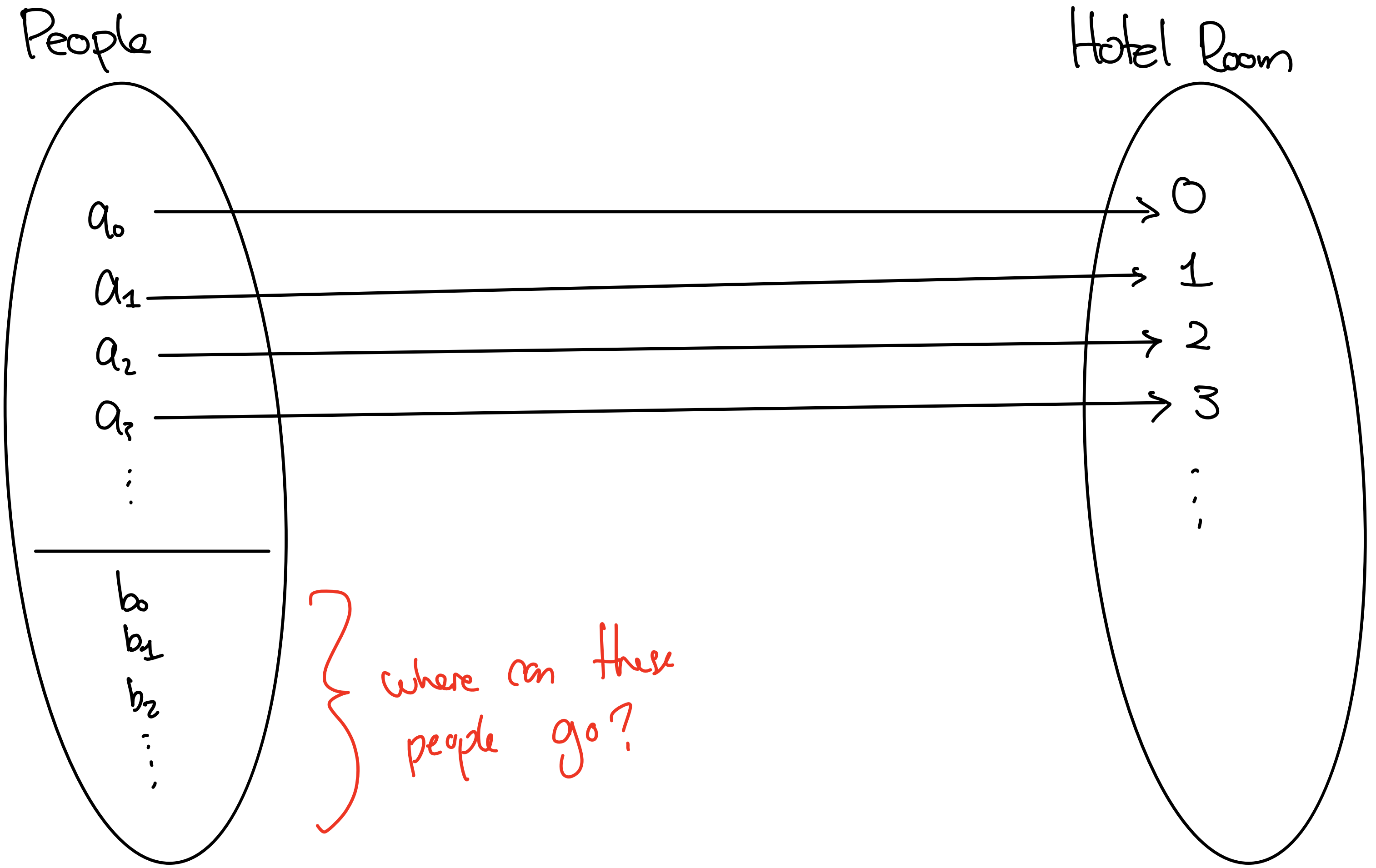
Nice! You managed to assign an infinite number of people to rooms! However, another bus arrives, and it again contains an infinite number of people. Let's call them $b_0, b_1, b_2 \ldots$.

You look at the rooms. Currently, each room is occupied! In particular, room number i is taken by customers a_i.

However, eager to impress your boss, you try to think of a way to do the impossible - fit an infinite number of people into an already filled hotel. So how do you do it?

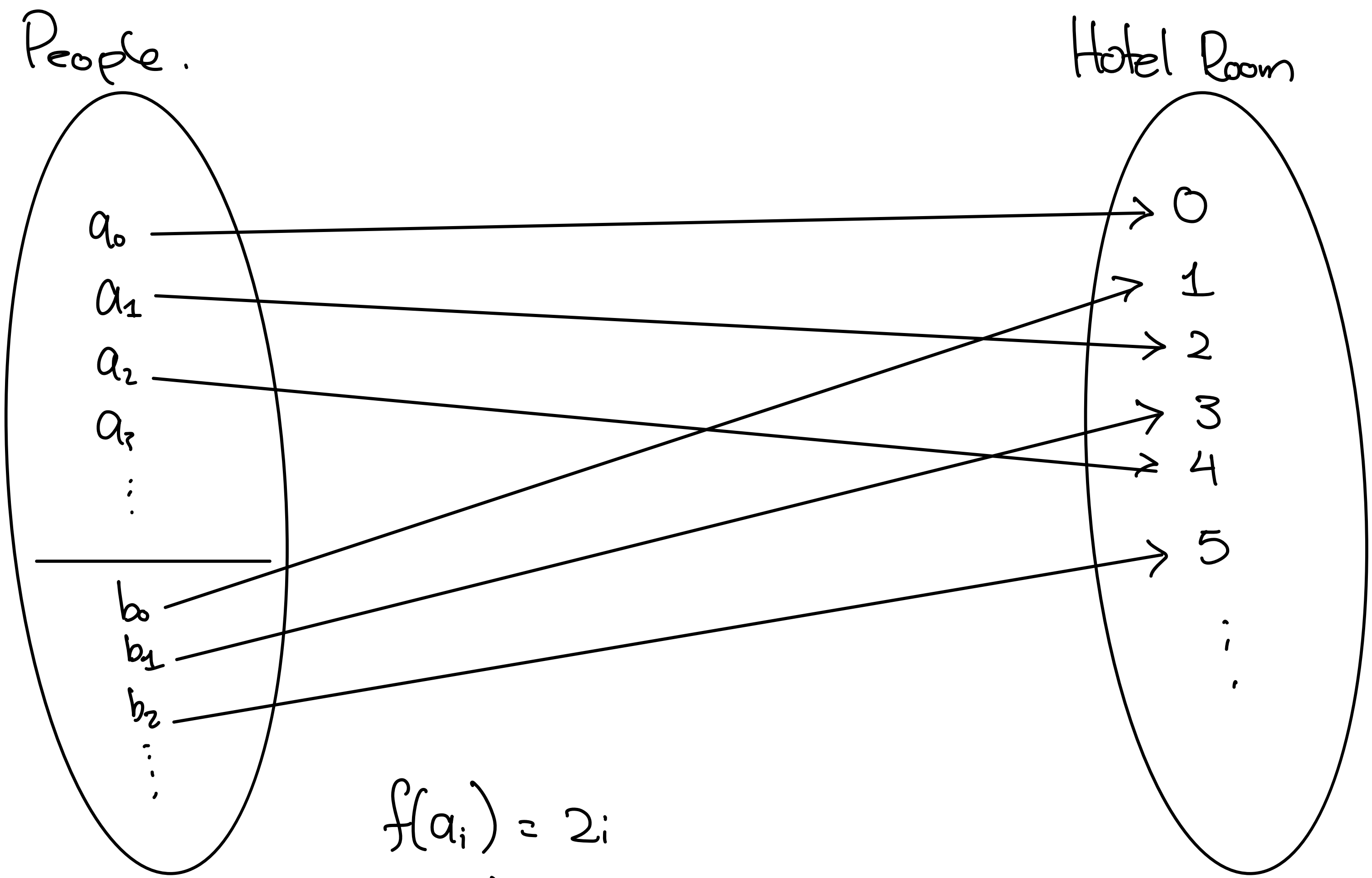
Example - Hilbert's Hotel

$f : A \to B$ is injective if $\forall x, y \in A.(x \neq y \implies f(x) \neq f(y))$



Example - Hilbert's Hotel

$f : A \to B$ is injective if $\forall x, y \in A.(x \neq y \implies f(x) \neq f(y))$



$f(a_i) = 2i$

$f(b_i) = 2i + 1$

Proof: f is injective

$$f(x) = \begin{cases} 2i & x = a_i \\ 2i+1 & x = b_i \end{cases}$$

$f : A \to B$ is injective if
$\forall x, y \in A.(x \neq y \implies f(x) \neq f(y))$

WTS $x \neq y \Rightarrow f(x) \neq f(y)$.

if $x \neq y$, there are several cases.

1.) if $x = a_i$, $y = b_j$ $\Rightarrow$ $f(x)$ is even, $f(y)$ is odd, $\Rightarrow f(x) \neq f(y)$.

2.) if $x = a_i$, $y = a_j$ note $i \neq j$ b/c $x \neq y$.

$\Rightarrow 2i \neq 2j \Rightarrow f(a_i) \neq f(a_j)$

Surjective

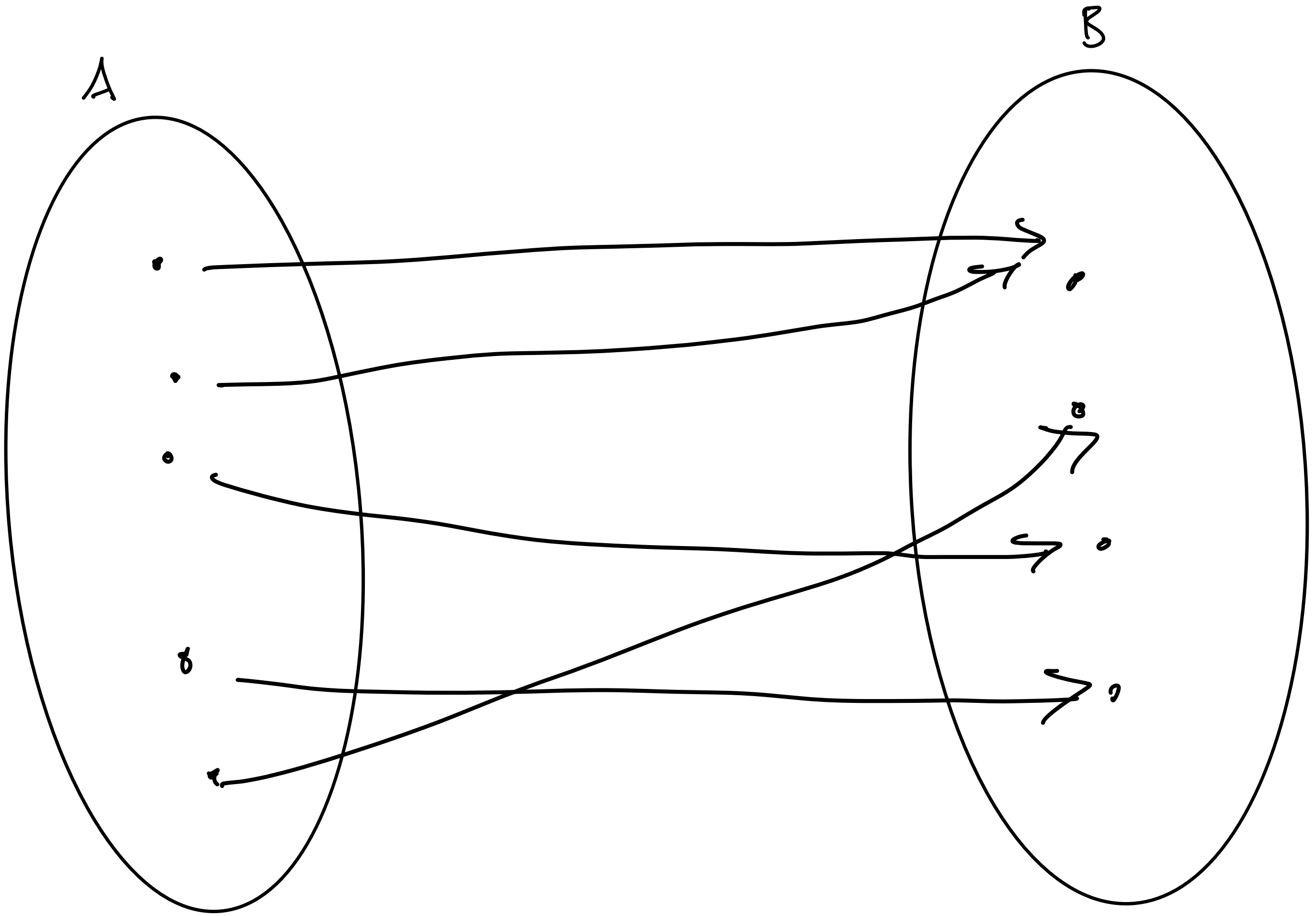
A function is surjective if everything in the codomain is hit.

Formally, $f : A \to B$ is surjective if

$$\forall b \in B . \exists a \in A . (f(a) = b)$$

f is surjective if
$\forall b \in B. \exists a \in A. (f(a) = b)$.

$f : A \to B$



Example

f is surjective if
$\forall b \in B . \exists a \in A . (f(a) = b)$.

Is f defined by $f(n) = n$ surjective?

Example

f is surjective if
$\forall b \in B . \exists a \in A . (f(a) = b)$.

Is f defined by $f(n) = n$ surjective?

It depends on the domain/codomain! For example, $f : \mathbb{N} \to \mathbb{N}$ defined by $f(n) = n$ is surjective, but $f : \mathbb{N} \to \mathbb{R}$ defined by $f(n) = n$ is not!

It's essential to **always specify the domain and codomain when defining a function**.

Bijective

A function is bijective if everything in the codomain is hit exactly once. Formally,

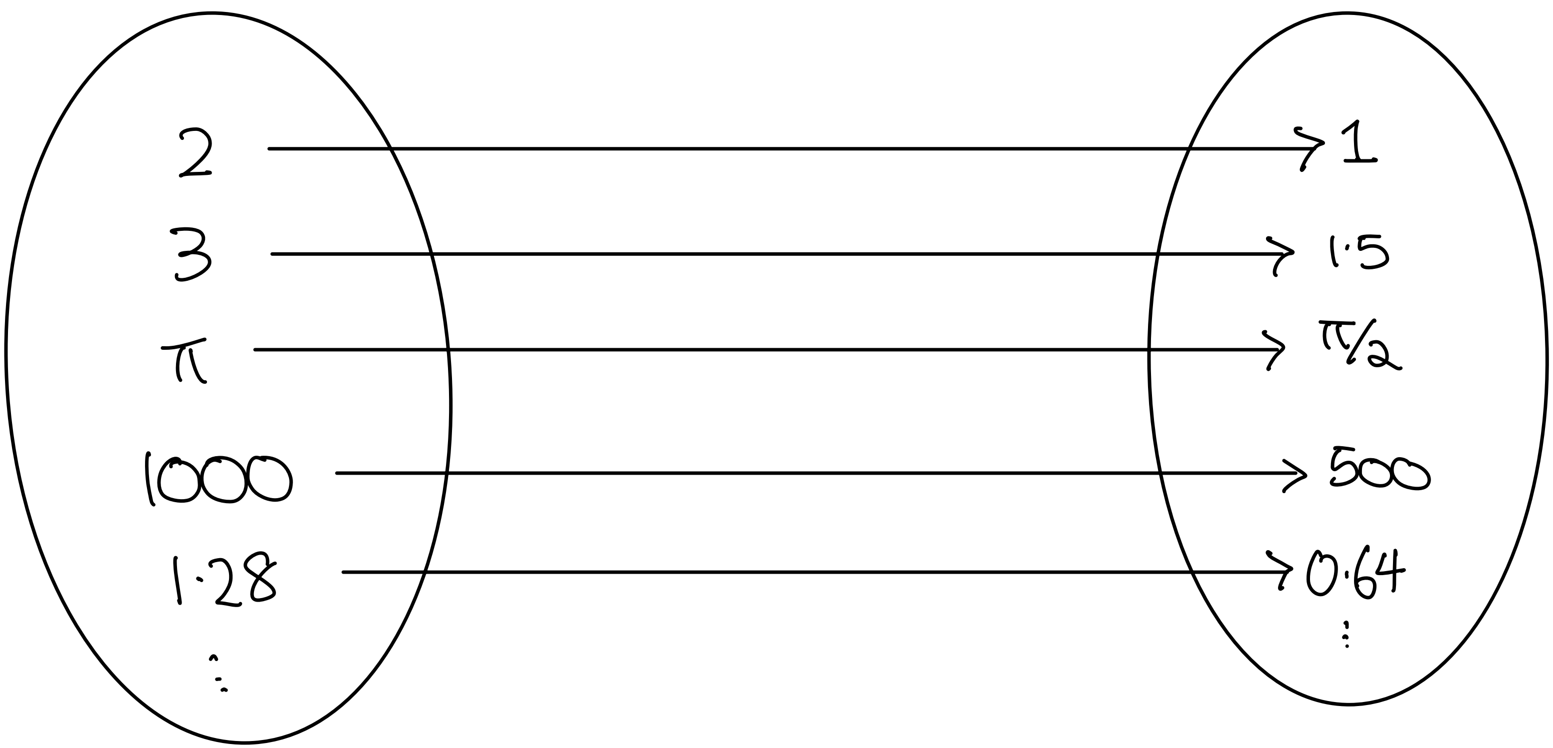
$f : A \to B$ is bijective if f is **both injective and surjective**.

Examples

(\mathbb{R})

$$f : \mathbb{R} \to \mathbb{R}$$ ← "real numbers"

$$f(x) = x/2$$



Proof - Half is Bijective

injective:
$\forall x, y \in A.(x \neq y \implies f(x) \neq f(y))$
surjective:
$\forall b \in B.\exists a \in A.(f(a) = b)$.

Injective : use the contrapositive definition: WTS: $f(x) = f(y) \Rightarrow x = y$

Suppose $f(x) = f(y) \Rightarrow x/2 = y/2$

$\Rightarrow x = y$ ✓

Surjective : let $b \in \mathbb{R}$, then $2b \in \mathbb{R}$ claim $f(2b) = b$.

$f(2b) = 2b/2 = b$ ✓

Summary of definitions

injective if no

surjectie if no

Bijective : all

Let $f : A \to B$ be a function.

f is...	if $\forall b \in B$, b is hit...	Formally...
Injective	1 or 0 times	$\forall x, y \in A.(x \neq y \implies f(x) \neq f(y))$
Surjective	at least 1 time	$\forall b \in B.\exists a \in A.(f(a) = b)$
Bijective	exactly 1 time	Injective and Surjective

$b \in B$ is hit k times by f if there are k distinct $a \in A$ are such that $f(a) = b$. I.e.

$$|\{a \in A : f(a) = b\}| = k.$$

Cantor's Theorem

- f is surjective if $\forall b \in B. \exists a \in A. (f(a) = b)$
- $\wp(A) = \{B : B \subseteq A\}$

Theorem (Cantor's Theorem)

For any set A, there is no surjection between A and $\wp(A)$

$P(\{1, 2\}) = \{\{\}, \{1\}, \{2\}, \{1, 2\}\}$.

↪ especially when A is infinite!

Proof of Cantor's Theorem

f is surjective if
$\forall b \in B. \exists a \in A. (f(a) = b)$.

Let $f : A \to \mathcal{P}(A)$ be any function.

f

	a_1	a_2	a_3	a_4	$\cdots$
$f(a_1)$:	✓	×	✓	✓	$\cdots$
$f(a_2)$:	×	×	✓	×	$\cdots$
$f(a_3)$:	×	×	×	×	$\cdots$
$f(a_4)$:	✓	✓	✓	×	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

× here means $a_1 \notin f(a_2)$.

✓ here means $a_3 \in f(a_4)$

Proof of Cantor's Theorem

f is surjective if
$\forall b \in B. \exists a \in A. (f(a) = b)$.

	a_1	a_2	a_3	a_4	$\cdots$
$f(a_1)$:	✗ ✓	✗	✓	✓	$\cdots$
$f(a_2)$:	✗	✓ ✗	✓	✗	$\cdots$
$f(a_3)$:	✗	✗	✓ ✗	✗	$\cdots$
$f(a_4)$:	✓	✓	✓	✓ ✗	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

D.

$D = \{a \in A : a \notin f(a)\}$

Proof of Cantor's Theorem

$f : A \to \mathcal{P}(A)$.

surjective:
$\forall b \in B. \exists a \in A. (f(a) = b)$.

$D = \{a \in A : a \notin f(a)\}$

By contradiction, suppose $D = f(a)$ for some $a \in A$.

is $a \in D$?

By def of D … By

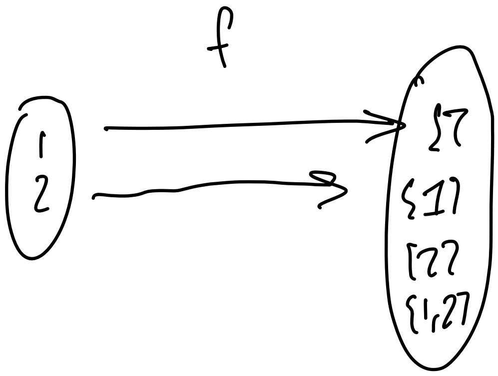
if $a \in D \Rightarrow a \notin f(a) \Rightarrow a \notin D$. ⚡

else $a \notin D \Rightarrow a \in f(a) \Rightarrow a \in D$. $\Rightarrow\Leftarrow$

Since we have reached a contradiction, the assumption must have been false, therefore $D \neq f(a)$.

Thus, f is not surjective.

Suppose $A = \{1,2\}$,



	1	2
$f(1)$	✗ ✓	✗
$f(2)$	✓	✗ ✓

$\Rightarrow D = \{a : a \notin f(a)\}$.

$D = \{1, 2\}$

my claim: $D \neq f(a)$ for any a.

Are the problems computers can't solve?

Let's apply our knowledge of functions to answer the question!

Formalizing the question - Problems

Let $\mathrm{Strings}$ be the set of all possible strings. For each subset $A \subseteq \mathrm{Strings}$ (each $A \in \wp(\mathrm{Strings})$), there is the problem of determining whether or not a given input is in A or not in A.

Formalizing the question - Problems

Let $\mathrm{Strings}$ be the set of all possible strings. For each subset $A \subseteq \mathrm{Strings}$ (each $A \in \wp(\mathrm{Strings})$), there is the problem of determining whether or not a given input is in A or not in A.

For example

$$A = \{w \in \mathrm{Strings} : w \text{ is a palindrome}\},$$

Formalizing the question - Problems

Let $\mathrm{Strings}$ be the set of all possible strings. For each subset $A \subseteq \mathrm{Strings}$ (each $A \in \wp(\mathrm{Strings})$), there is the problem of determining whether or not a given input is in A or not in A.

For example

$$A = \{w \in \mathrm{Strings} : w \text{ is a palindrome}\},$$

or

$$A = \{w : w \text{ is a C program with no syntax errors}\}.$$

Formalizing the question - Problems

Let $\mathrm{Strings}$ be the set of all possible strings. For each subset $A \subseteq \mathrm{Strings}$ (each $A \in \wp(\mathrm{Strings})$), there is the problem of determining whether or not a given input is in A or not in A.

For example

$$A = \{w \in \mathrm{Strings} : w \text{ is a palindrome}\},$$

or

$$A = \{w : w \text{ is a C program with no syntax errors}\}.$$

Let's just consider problems of this type. So set

$$\mathrm{Problems} = \wp(\mathrm{Strings}).$$

Formalizing the question - Programs

For concreteness, let's say, a program P solves a problem $A \subseteq \text{Strings}$, if $\forall w \in \text{Strings}$,

$$w \in A \iff P \text{ run on input } w \text{ prints 1 and nothing else}$$

Identify every program with its source code (a string), so $\text{Programs} \subseteq \text{Strings}$.

Formalizing the question - Solves

Let $\mathrm{Solves} : \mathrm{Programs} \to \wp(\mathrm{Strings})$ be the function that maps each program to the problem it solves. Since each program solves at most one problem, this function is well defined.

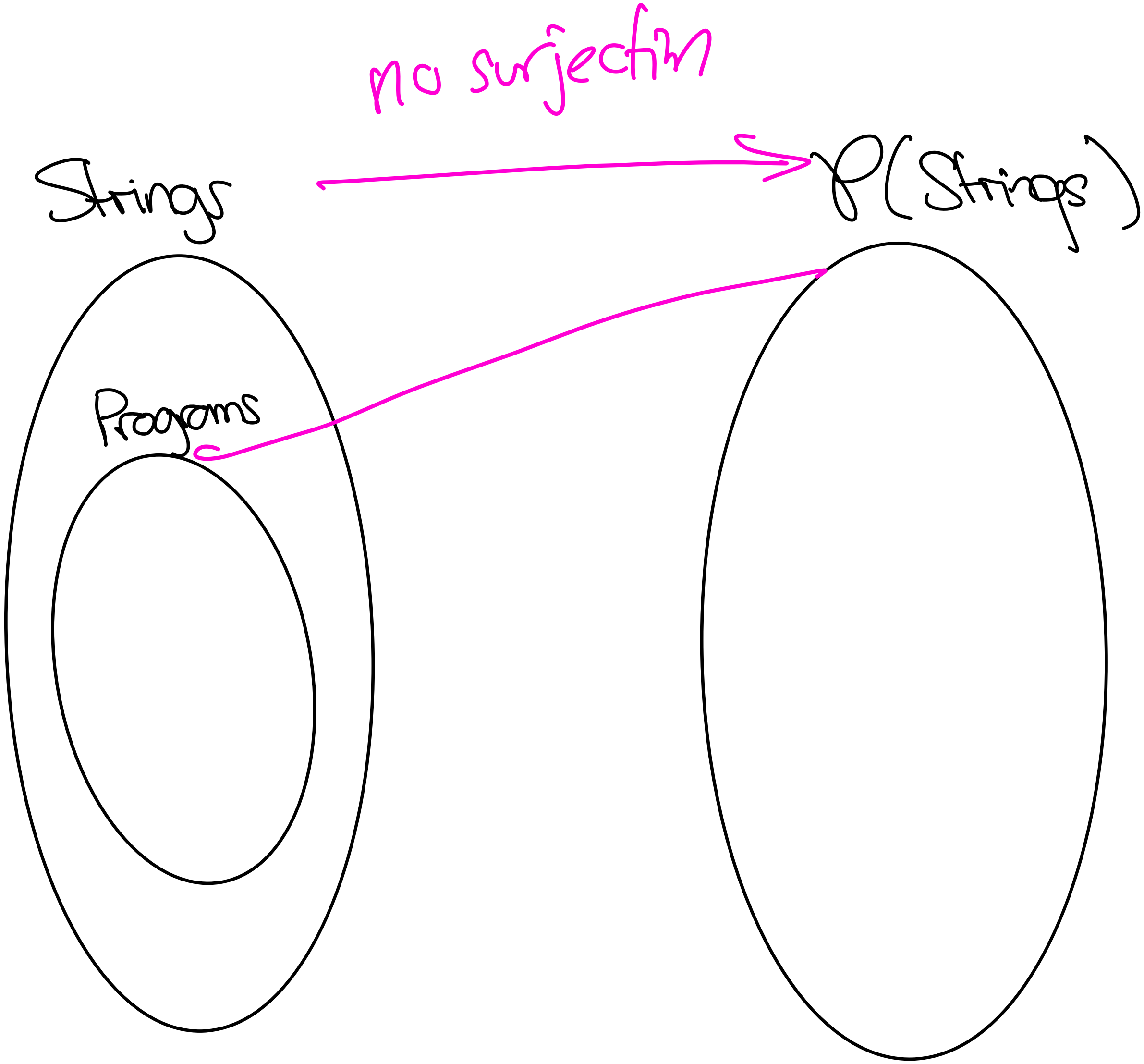
Our question about whether or not computers can solve all problems is the question of whether or not Solves is ...

Formalizing the question - Solves

Let $\mathrm{Solves} : \mathrm{Programs} \to \wp(\mathrm{Strings})$ be the function that maps each program to the problem it solves. Since each program solves at most one problem, this function is well defined.

Our question about whether or not computers can solve all problems is the question of whether or not Solves is surjective!

Proof (Picture)



Proof - There is a problem that computers can't solve

By contradiction, assume Solves : Programs $\to \mathcal{P}$(Strings) is surjective, then define

$$f : \text{Strings} \to \mathcal{P}(\text{Strings})$$

$$f(a) = \begin{cases} \text{Solves}(a) & \text{if } a \in \text{Programs} \\ \emptyset & \end{cases}$$

then, I claim f is surjective: Let $b \in \mathcal{P}$(Strings),

since Solves is surjective $\exists\, a \in$ Programs s.t. Solves$(a) = b$..

$f(a) =$ Solves(a), since $a \in$ Programs $\Rightarrow f(a) = b$.

$\Rightarrow f$ is surjective. contradicts Cantor's theorem.

There is a problem that computers can't solve

What are your questions?

FAQ

- “How many problems can/can’t computers solve?”
- “What is a particular problem that we can’t solve with computers”
- “How can we tell if a problem can or can’t be solved by computers?”
- “Some problems can be solved and other can not - so some problems are computationally “harder” than others. Are there more ways to compare how computationally hard problems are?”

FAQ

- "How many problems can/can't computers solve?"
- "What is a particular problem that we can't solve with computers"
- "How can we tell if a problem can or can't be solved by computers?"
- "Some problems can be solved and other can not - so some problems are computationally "harder" than others. Are there more ways to compare how computationally hard problems are?"

Answer: Take CSC448 and CSC463 :)

Tutorials!

Meet your TAs!

Announcement

If you are in tutorial 5101 (Room BA2165), please go to the following room instead

- If your birthday is on the 1-10th of the month, go to BA2195
- If your birthday is on the 11-20th of the month, go to BA 2159
- If your birthday is on the 21-31st of the month, go to BA2139

Additional Notes

- ‘hits’ in the definitions of injective/surjective/bijective is not standard terminology. The standard way to express the same meaning is to use the word ‘preimage.’ In your proofs you should use the formal FOL definitions.
- The visual part of the proof of Cantor’s Theorem is a little misleading. It makes an additional assumption that you can list the elements of A using the natural numbers ($\mathbb{N}$) (this property is called ‘countable’). But this is not true for all sets! For example, apply Cantor’s Theorem to $\mathbb{N}$ to show that $\wp(\mathbb{N})$ can not be listed using the natural numbers!
- Cantor’s Theorem is fundamental and deep. In particular, it implies that there are bigger and smaller infinities (!). I.e., the powerset of an infinite set is strictly bigger. The powerset of that set is strictly bigger again, and so on. If this interests you, take a class on Set Theory!

Suggested Reading

- VH Ch 0.1-0.3, 0.6