is functional
A binary relation on a domain D describes/a function
from D iff there is exactly one pair (x,y) in the relation for each $x \in D$. D is called the domain of the function.

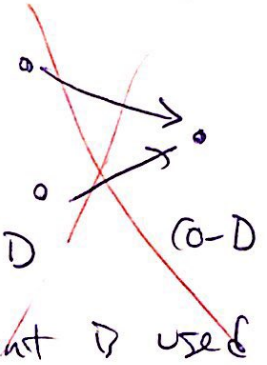
Each function must also specify a co-domain set from which output values are taken.

We write $f: S \to T$ to say f has domain S and codomain T and then specify f by giving a functional subset of $S \times T$.

* Functions using the same set of pairs are different functions if they have different co-domains

A function can be:

injective (one-to-one): each co-domain element is used at most ONCE.



surjective (ONTO): each co-domain element is used.

range = co-domain

bijective: injective and surjective.

The range of a function is the subset of the co-domain that is used.

A permutation of a set S is a bijection $S \to S$.

If $f: S \to T$ and $g: T \to U$

then $g \circ f: S \to U$ "the composition of g and f"

[we write $f(x)$ when $x \in D$ and $f: D \to S$ to represent
the unique $y \in S$ s.t. (x,y) is in the set of pairs for f]

"the image of x under f"

$g \circ f$ is defined so that $(g \circ f)(x) = g(f(x))$ for $x \in S$

Thm The composition of 2 bijections is itself a bijection.

$$|A| = |B| = |C|$$

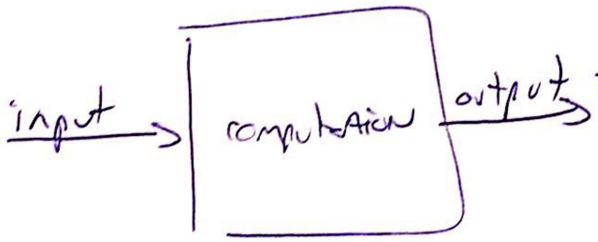
Early mathematicians of computation

Alan Turing — Turing machine

Alonzo Church — The lambda calculus (underlies LISP)

- exactly equivalent expressive power / computability.

suggests: The above both define "computation"



$\Longrightarrow$ limit to yes/no.

A specification for the computation would describe the desired input/output pairs. $\Longrightarrow$ input set of strings that should get "yes"

A desired computational behavior for yes/no computation (decision problems) can be specified by a set of strings, the inputs for which we want "yes".

typically infinite

a language

Obs: There are countably many finite strings. (Σ^*) ← set of finite strings over alphabet Σ

Obs. Thus there are countably many programs.

How many subsets of Σ^* are there? $|\mathcal{P}(\Sigma^*)|$ so uncountably

→ these are the languages

∴ There are uncountably many computational ~~tasks~~ task specifications (languages) but only countably many programs (each solves at most one)