Type Theory
Naive / Informal
Agda

What is the difference between naive set theory and naive type theory?

1. Elementhood

$3 \in \mathbb{N}$

Proposition

Python (dynamic typing)

$3 : \mathbb{N}$

Judgement
static property

Haskell (static typing)

$A \subseteq B \leftrightarrow \forall x.\ x \in A \to x \in B$

2. Logic

Set theory ∈ predicate logic

Prop = Bool

Explain logic by assigning the type of evidence to every proposition.

Propositions as types

explanation. Prop = Type

(Curry-Howard Equivalence)

Simple Types

Functions

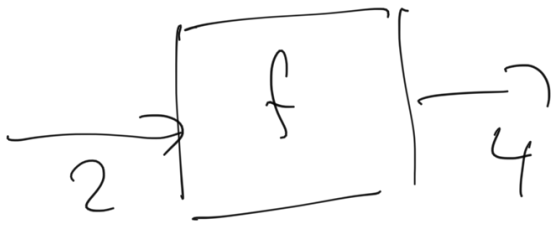
$A \to B$ $\quad f(x) = x + 2$

$f \in \mathbb{N} \to \mathbb{N}$

$f = \{(0,2), (1,3), (2,4), \ldots\}$

$\subset \mathbb{N} \times \mathbb{N}$

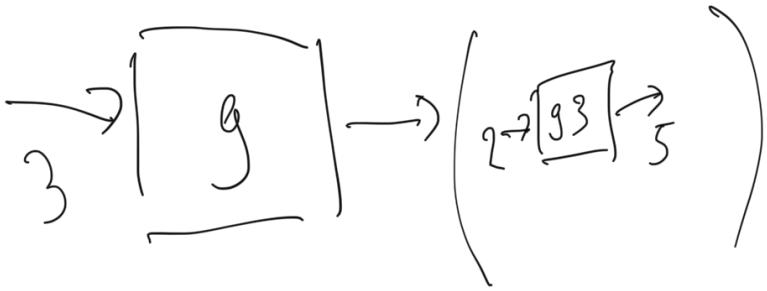
$f : \mathbb{N} \to \mathbb{N} \quad f\,x = x + 2$

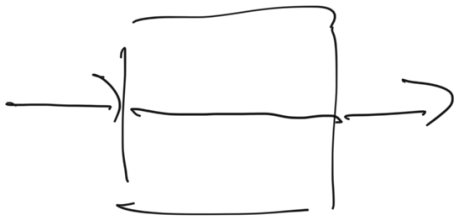


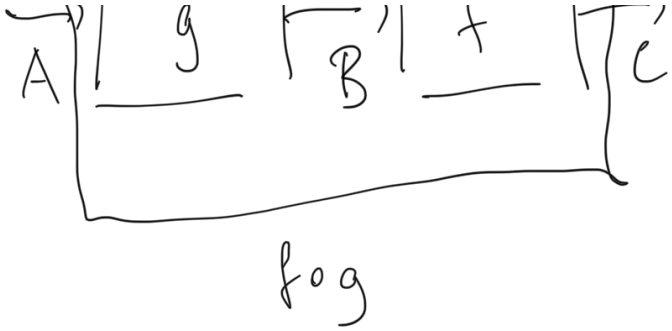
functions in type theory are primitive.

$R : \mathbb{N} \to \mathbb{N} \to \mathrm{Prop}$

~ functional programming






A
g
B
f
C
fog