Spartan Martin-Löf Type Theory

Andrej Bauer
University of Ljubljana

UniMath School
April 2019, Birmingham, UK

(Spartan Martin-Löf) Type Theory

Andrej Bauer
University of Ljubljana

UniMath School
April 2019, Birmingham, UK

Spartan (Martin-Löf Type Theory)

Andrej Bauer
University of Ljubljana

UniMath School
April 2019, Birmingham, UK

spartan | ˈspɑːt(ə)n |

adjective

showing or characterized by austerity or a lack of comfort or luxury: *the accommodation was fairly spartan*.

What is type theory?

A foundation of mathematics

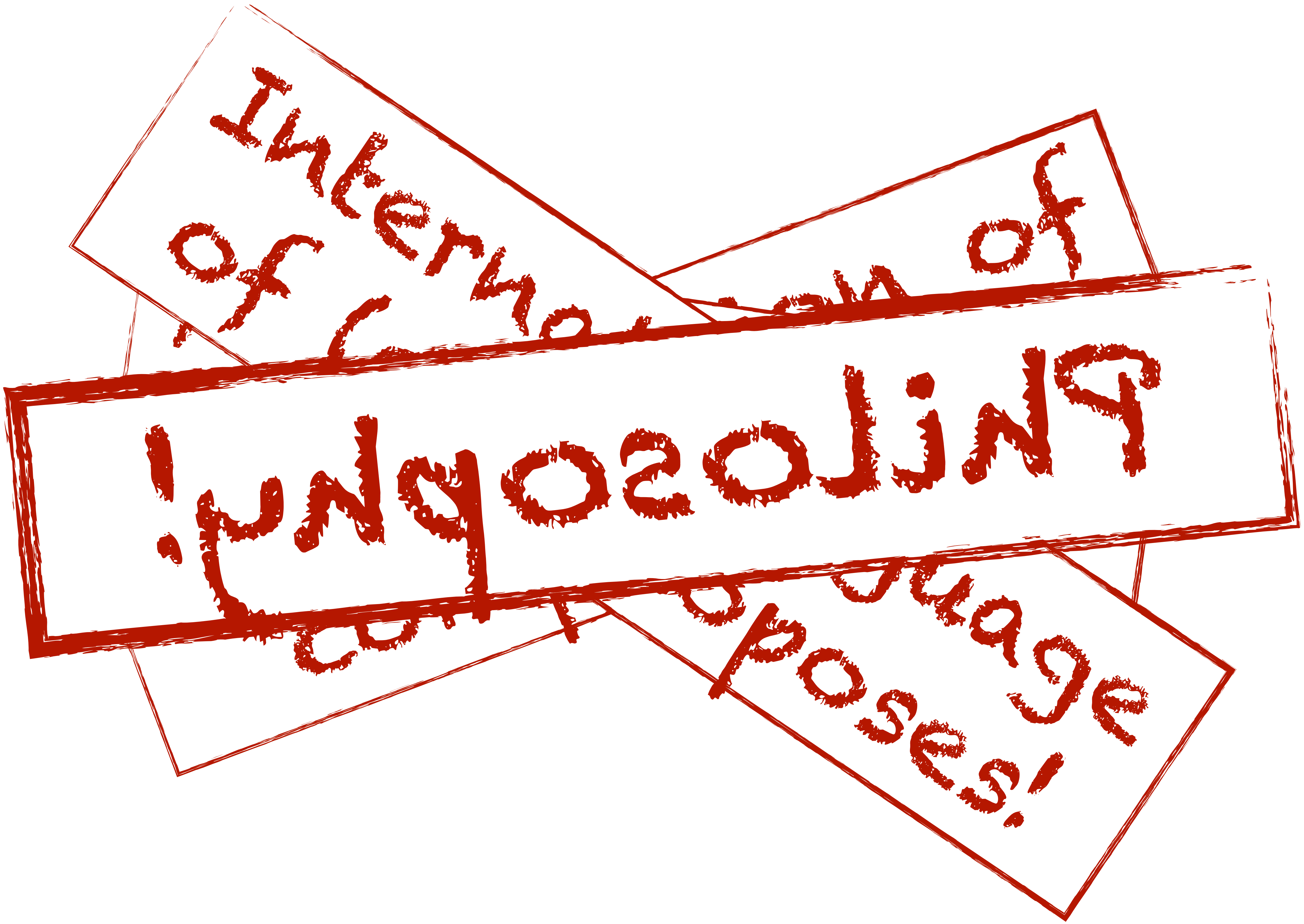
A f

atics

A foundation of computation!

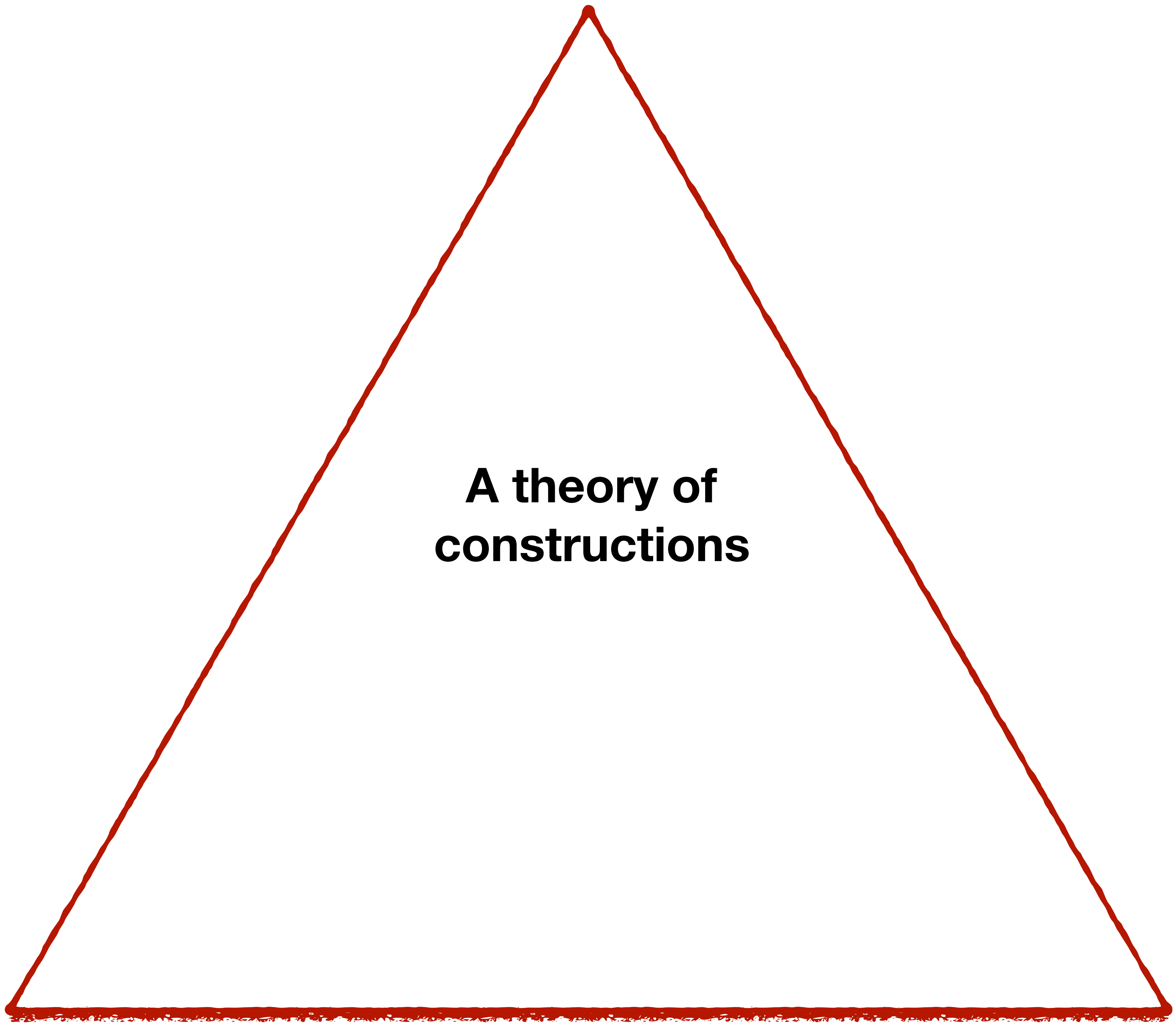
A programming language!

Internal language of $(\infty,1)$-toposes!

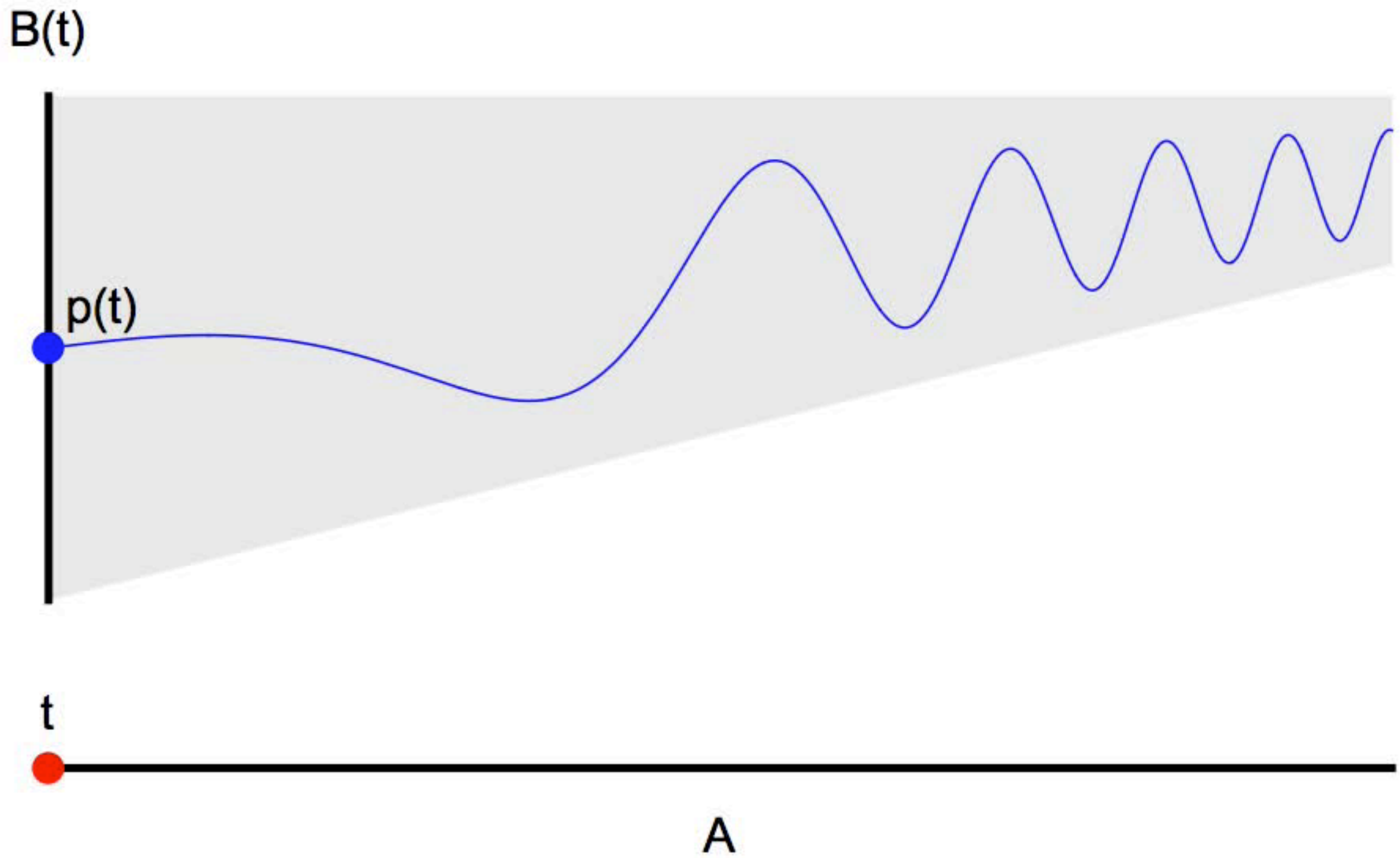

Philosophy!
of
of
uage
poses!

A theory of constructions

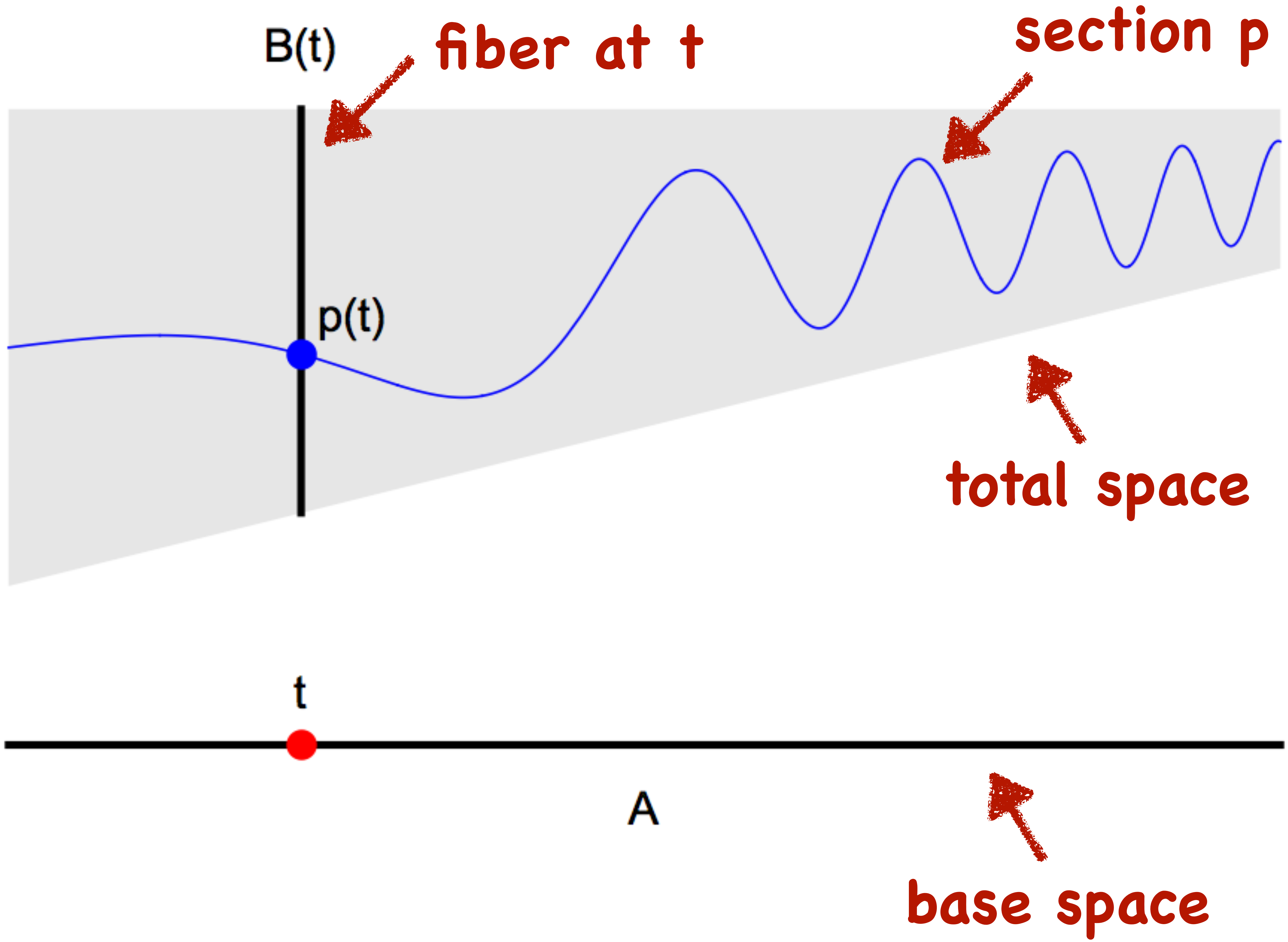
A theory of constructions


A theory of
constructions

A theory of *dependent* constructions

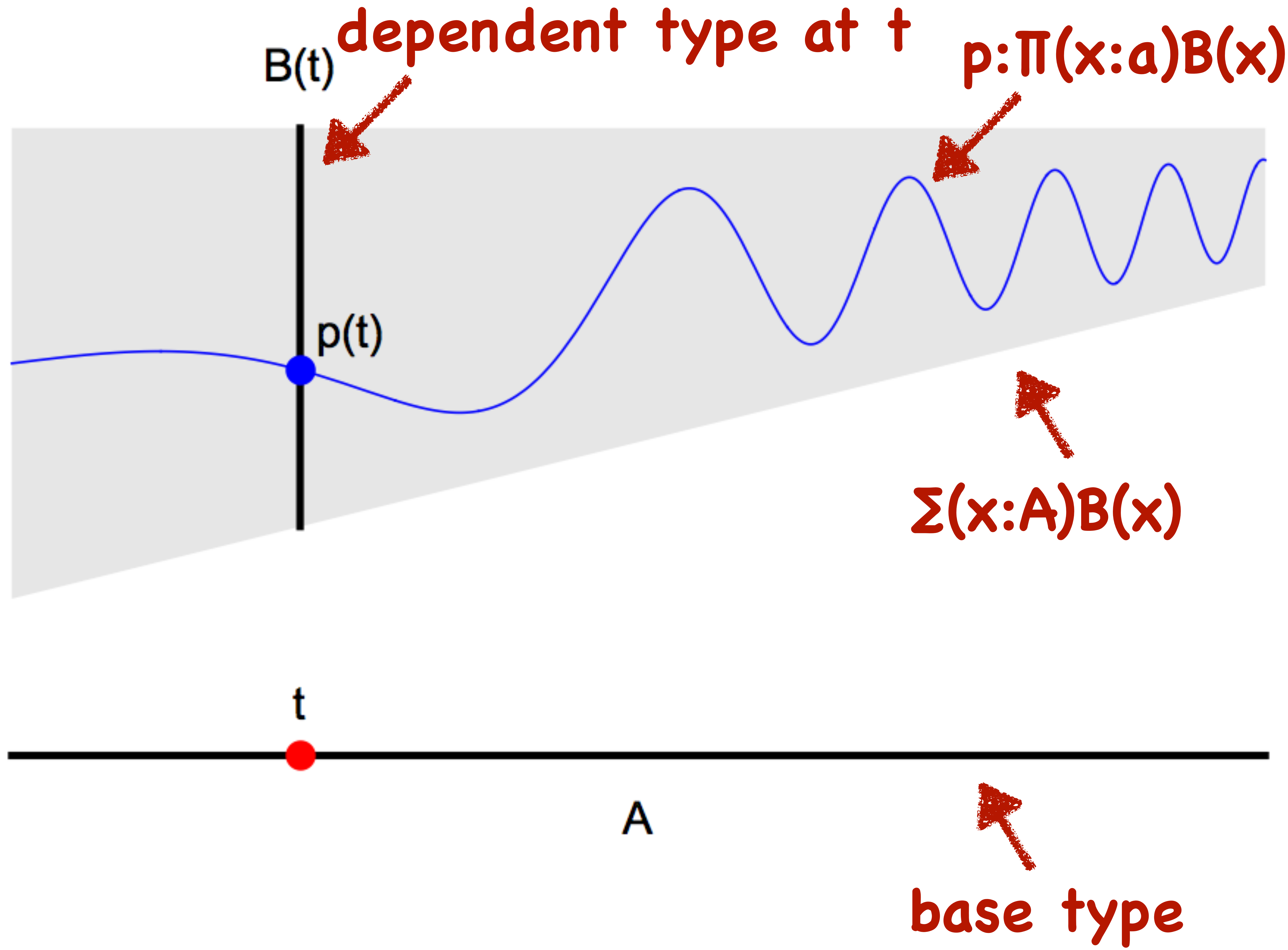


B(t)
p(t)
t
A


B(t)
fiber at t
section p
p(t)
total space
t
A
base space

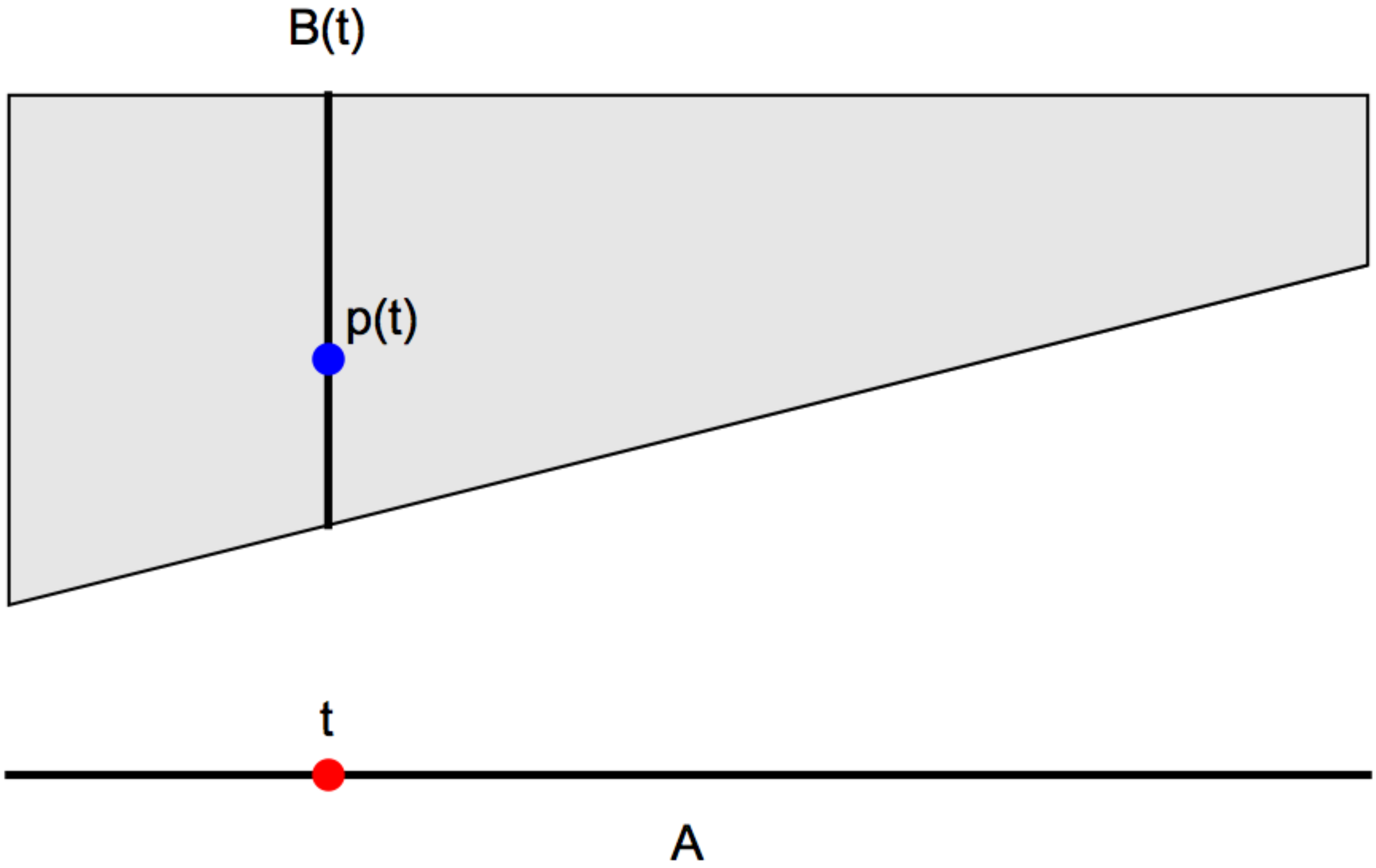
Type	A type
Element	$p : A$
Equal types	$A \equiv B$
Equal elements	$p \equiv_A q$

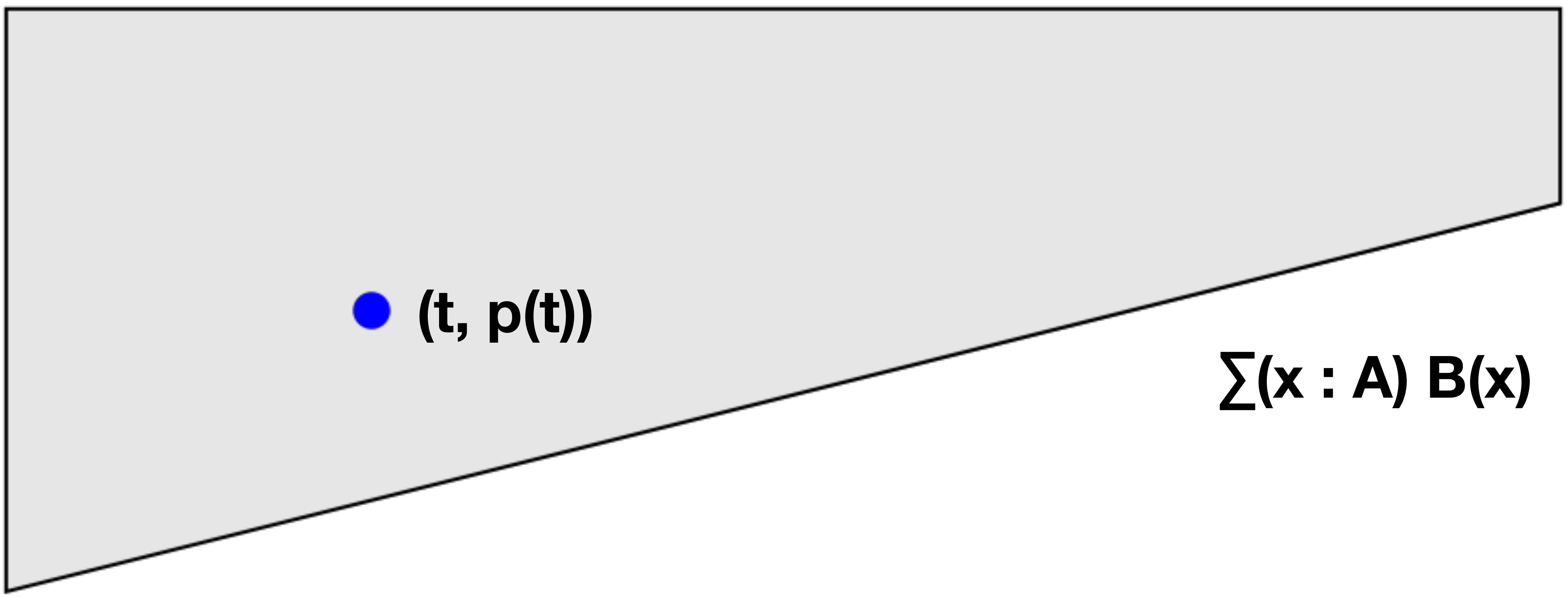
Space	A type
Point	$p : A$
Equal spaces	$A \equiv B$
Equal points	$p \equiv q : A$


dependent type at t
B(t)
p:Π(x:a)B(x)
p(t)
Σ(x:A)B(x)
t
A
base type

Sum $\sum(x{:}A)\ B(x)$

- **formation:**
  if type $\mathbf{B(x)}$ depends on $\mathbf{x : A}$, then $\mathbf{\sum(x : A)\ B(x)}$ is a type.
- **introduction:**
  if $\mathbf{t : A}$ and $\mathbf{u : B(t)}$ then $\mathbf{(t, u) : \sum(x : A)\ B(x)}$.
- **elimination:**
  - If $\mathbf{p : \sum(x : A)\ B(x)}$ then $\mathbf{\pi_1(p) : A}$.
  - If $\mathbf{p : \sum(x : A)\ B(x)}$ then $\mathbf{\pi_2(p) : B(\pi_1(p))}$.
- **equations:**
  - $\mathbf{\pi_1(t, u) \equiv t}$
  - $\mathbf{\pi_2(t, u) \equiv u}$
  - $\mathbf{(\pi_1(p), \pi_2(p)) \equiv p}$


B(t)
p(t)
t
A


(t, p(t))
∑(x : A) B(x)

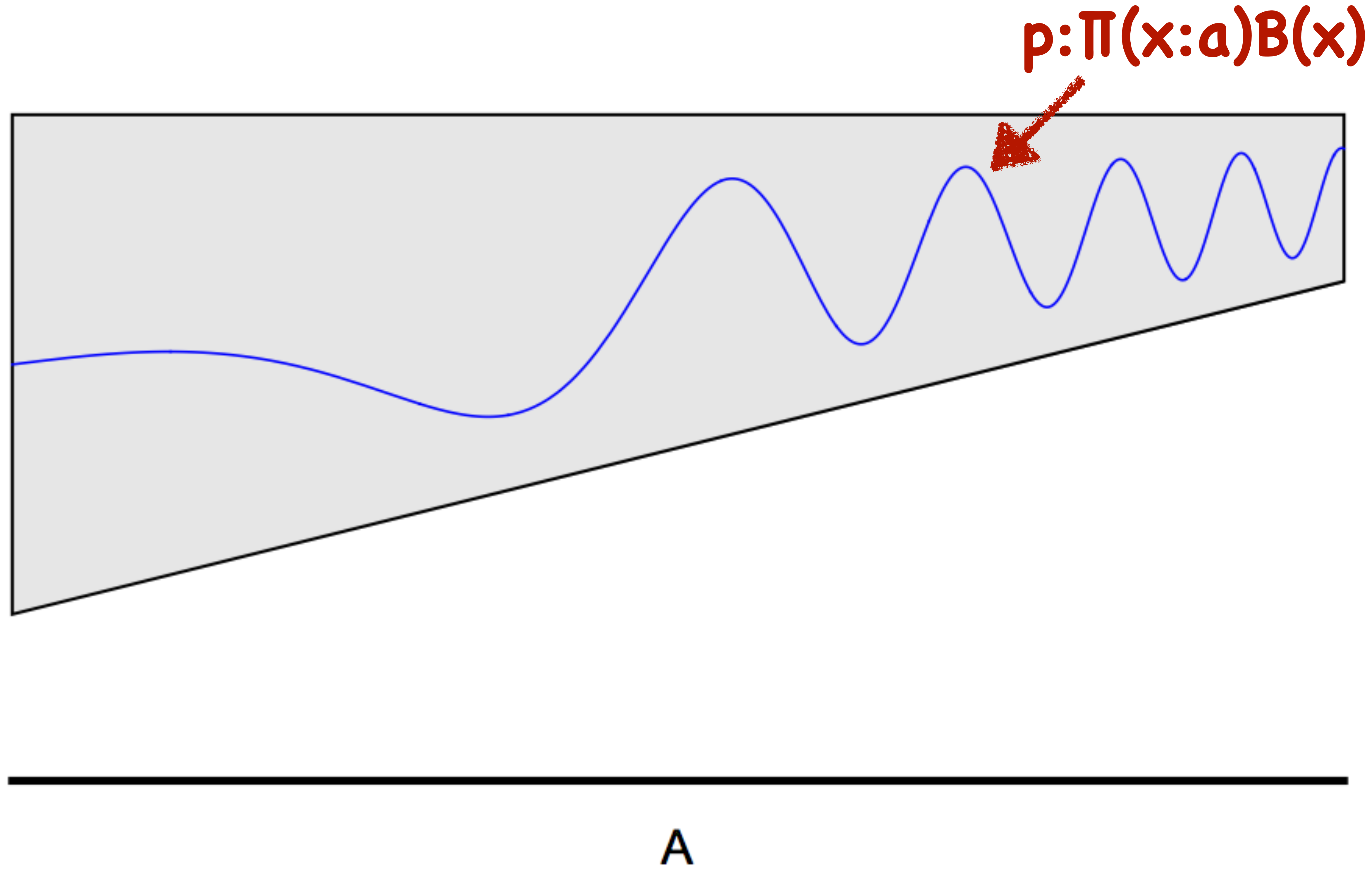
Binary product

$$A \times B \;\equiv\; \sum(\_ : A)\, B$$

Anonymous variable
(B does not depend on A)

Product $\prod(x{:}A)\ B(x)$

- given $\mathbf{B(x)}$ over $\mathbf{A}$, $\prod\mathbf{(x : A)\ B(x)}$ is a type
- if $\mathbf{p(x) : B(x)}$ then $\mathbf{\lambda(x{:}A)\ p(x) : \prod(x : A)\ B(x)}$.
- If $\mathbf{f : \prod(x : A)\ B(x)}$ and $\mathbf{t : A}$ then $\mathbf{f(t) : B(t)}$.
- $\mathbf{(\lambda(x{:}A)\ p(x))(t)} \equiv \mathbf{p(t)}$
- $\mathbf{\lambda(x{:}A)(f(x))} \equiv \mathbf{f}$



p:Π(x:a)B(x)
A

Function space

$$\mathbf{A \to B \equiv \prod(\_ : A)\, B}$$

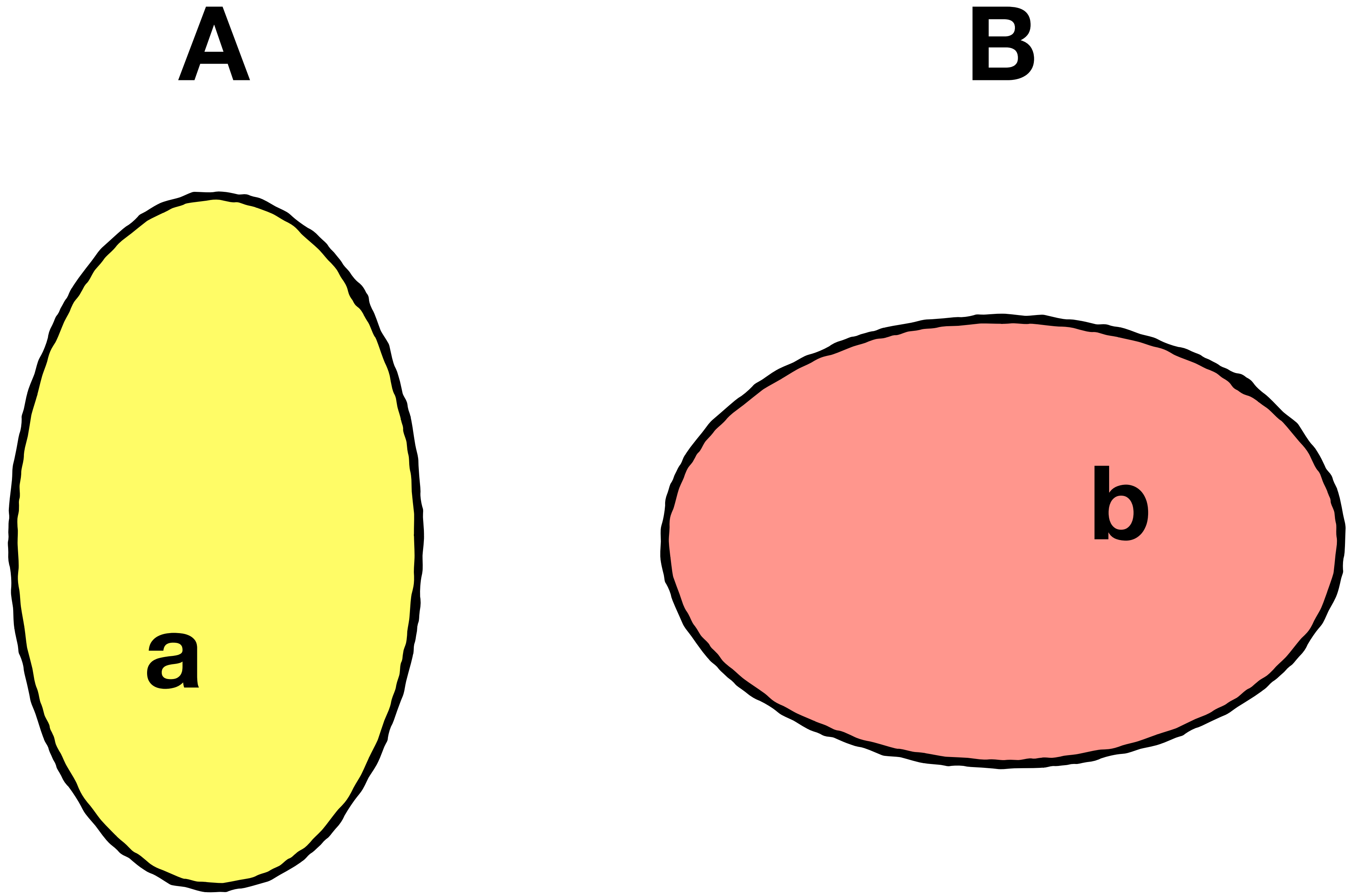
Universe

- There is a type **Type**.
- The elements of **Type** are types.
- A dependent type is a map $\mathbf{B : A \to Type}$.
- Beware of paradoxes:
  there can be no type of all types!
- $\mathbf{Type_0 : Type_1 : Type_2 : \ldots}$

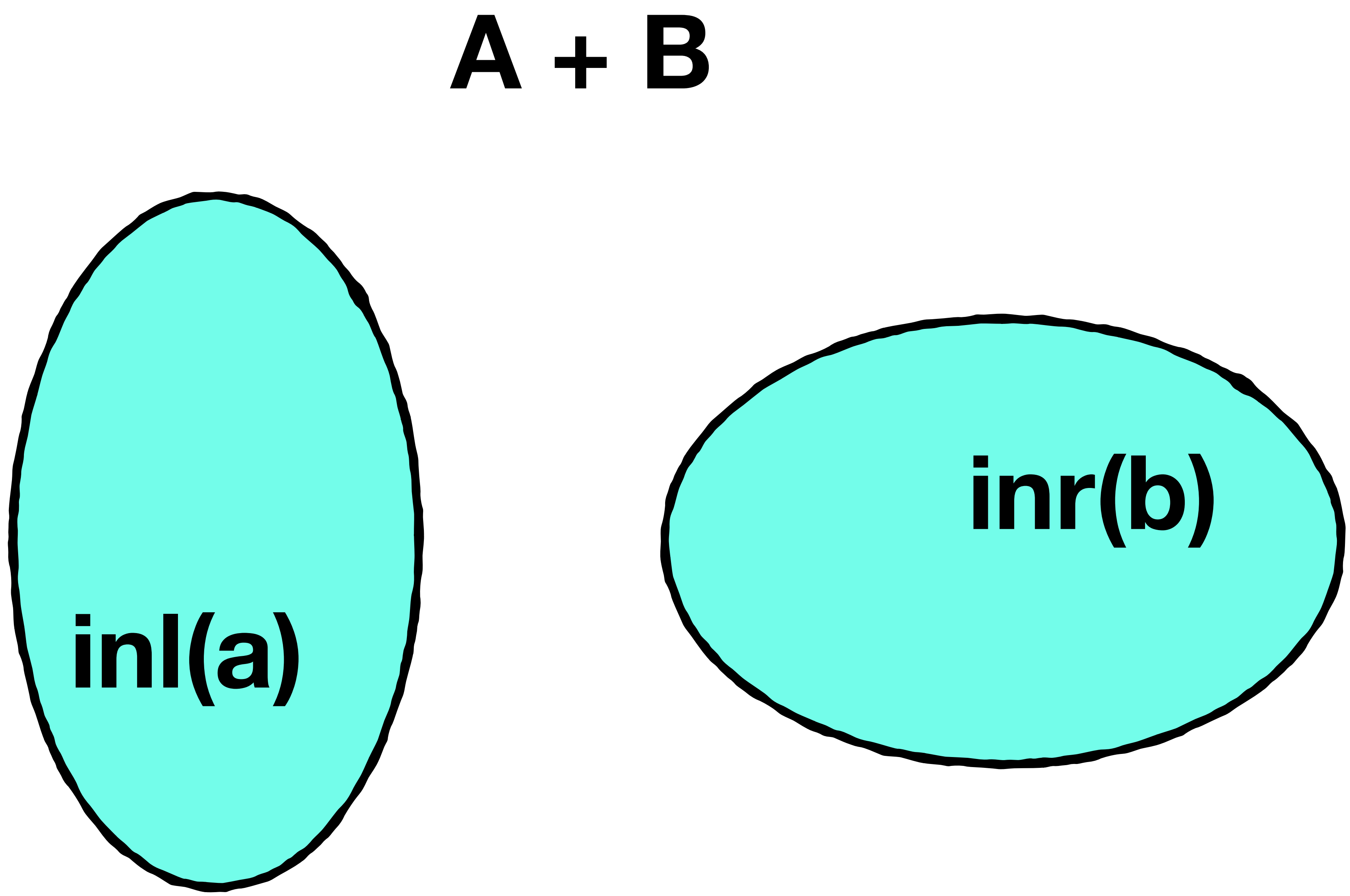
Basic types

- **Unit** – element is **tt**
- **Empty** – no elements
- **Bool** – elements **true** and **false**
- **N** – natural numbers

Simple sum A + B



Simple sum A + B



Natural numbers N

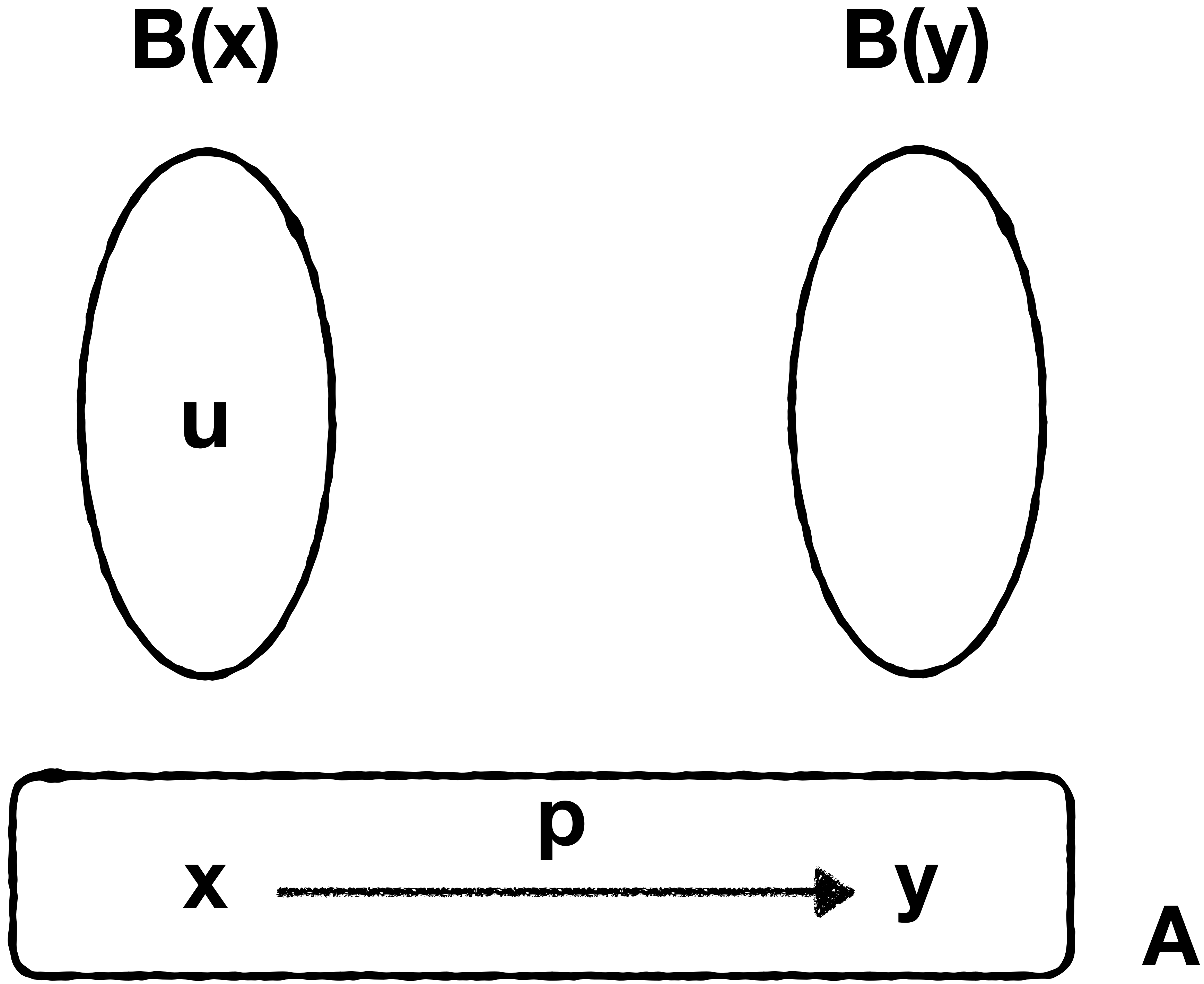
- **0 : N**
- If **n : N** then **S(n) : N**.
- If **P : N → Type** and **e : P(0)** and
  f : ∏(x:N) P(x) → P(S(x)) then
  ind_nat P e f : ∏(x:N) P(x).
- **ind_nat P e f 0 ≡ e**
- **ind_nat P e f (S n) ≡ f n (ind_nat P e f n)**

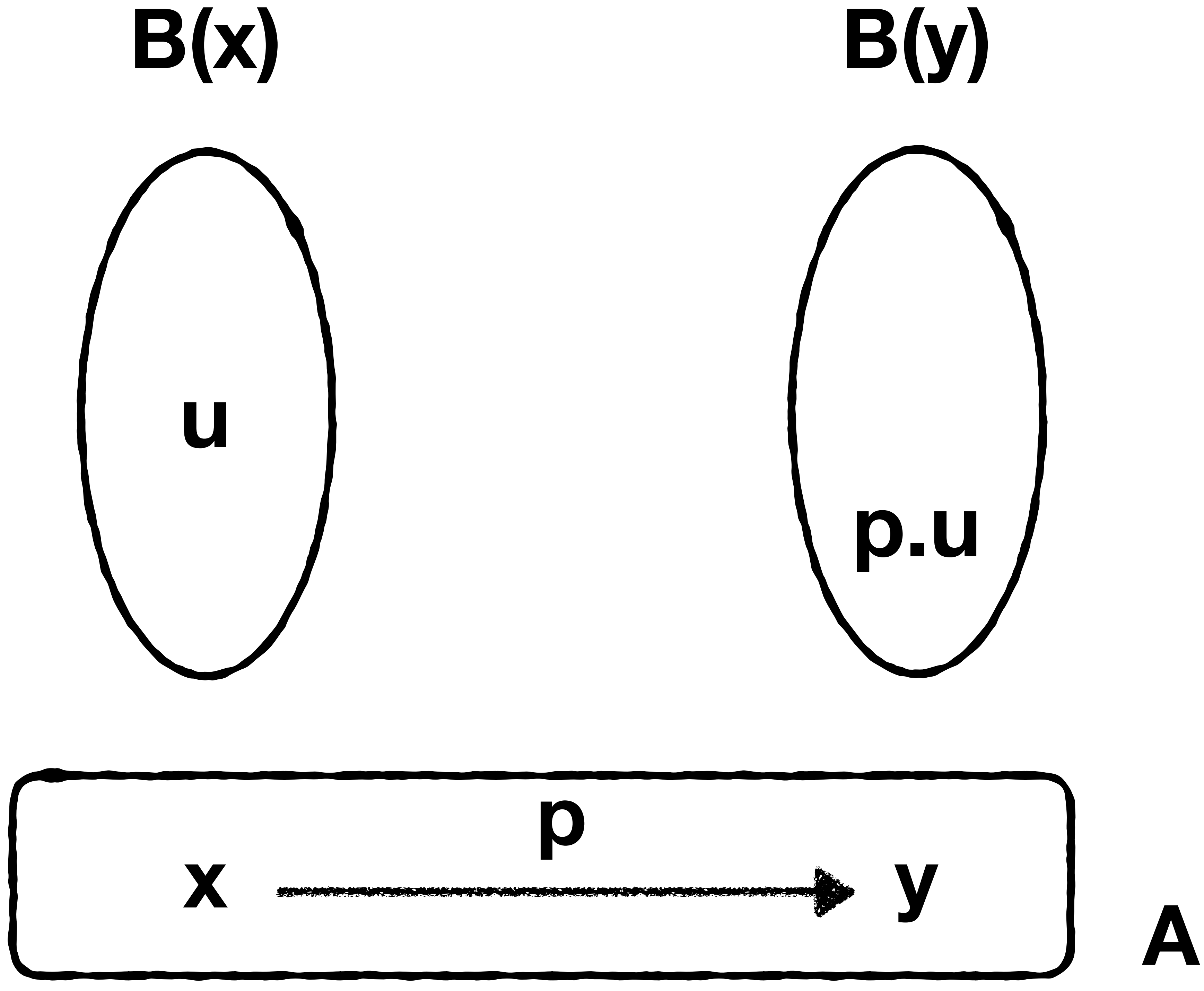
Path space

- If $\mathbf{t} : \mathbf{A}$ and $\mathbf{u} : \mathbf{A}$ then $\mathbf{Paths_A(t, u)}$ is a type.
- If $\mathbf{t} : \mathbf{A}$ then $\mathbf{idpath(t)} : \mathbf{Paths_A(t, t)}$.
- We write $\mathbf{t} = \mathbf{u}$ for $\mathbf{Paths_A(t, u)}$.

Transport

- Given a type **A** and **B : A → Type**
- If **α : Paths_A(x, y)** and **s : B(x)** then **transport B α s : B(y)**.
- **transport B (idpath(x)) s $\equiv$ s**.


B(x)
B(y)
u
p
x
y
A


B(x)
B(y)
u
p.u
p
x
y
A

Caveats

- Old people call path spaces *"identity types"* and use the notation $\mathbf{Id_A(x, y)}$.
- Older people call path spaces *"propositional equality"* and they call equality *"judgmental equality"*.
- Half of the definition of path spaces is missing. The other half will be given later this week.
- Path spaces *are* the equality you are used to. Really!

Proofs as constructions

“Every natural number is even or odd.”

Proofs as constructions

"Every natural number is even or odd."

$$\prod(n{:}\mathrm{nat})\ \sum(m{:}\mathrm{nat})\ (n = 2m) + (n = 2m+1)$$

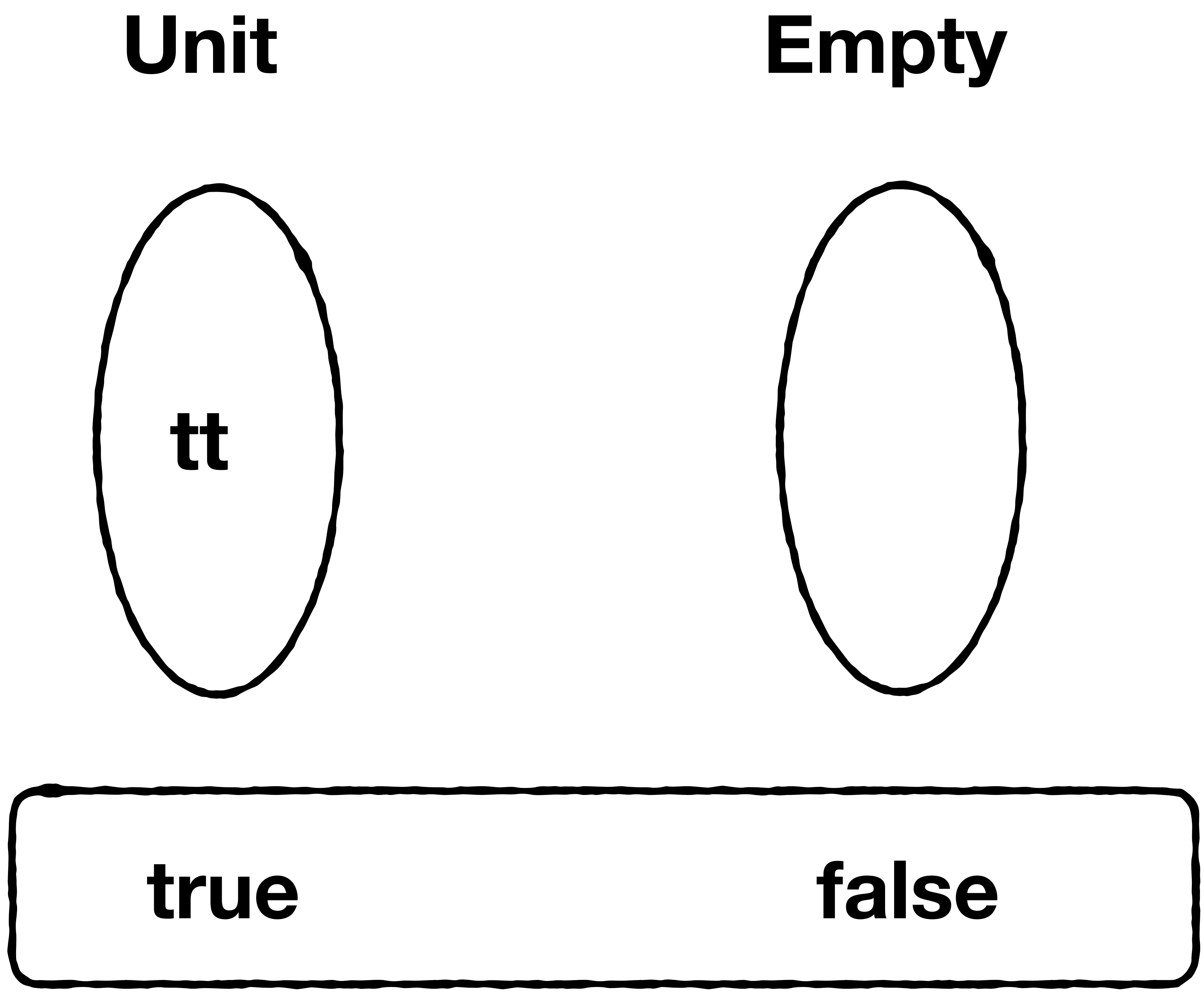
How do we *deny* P?

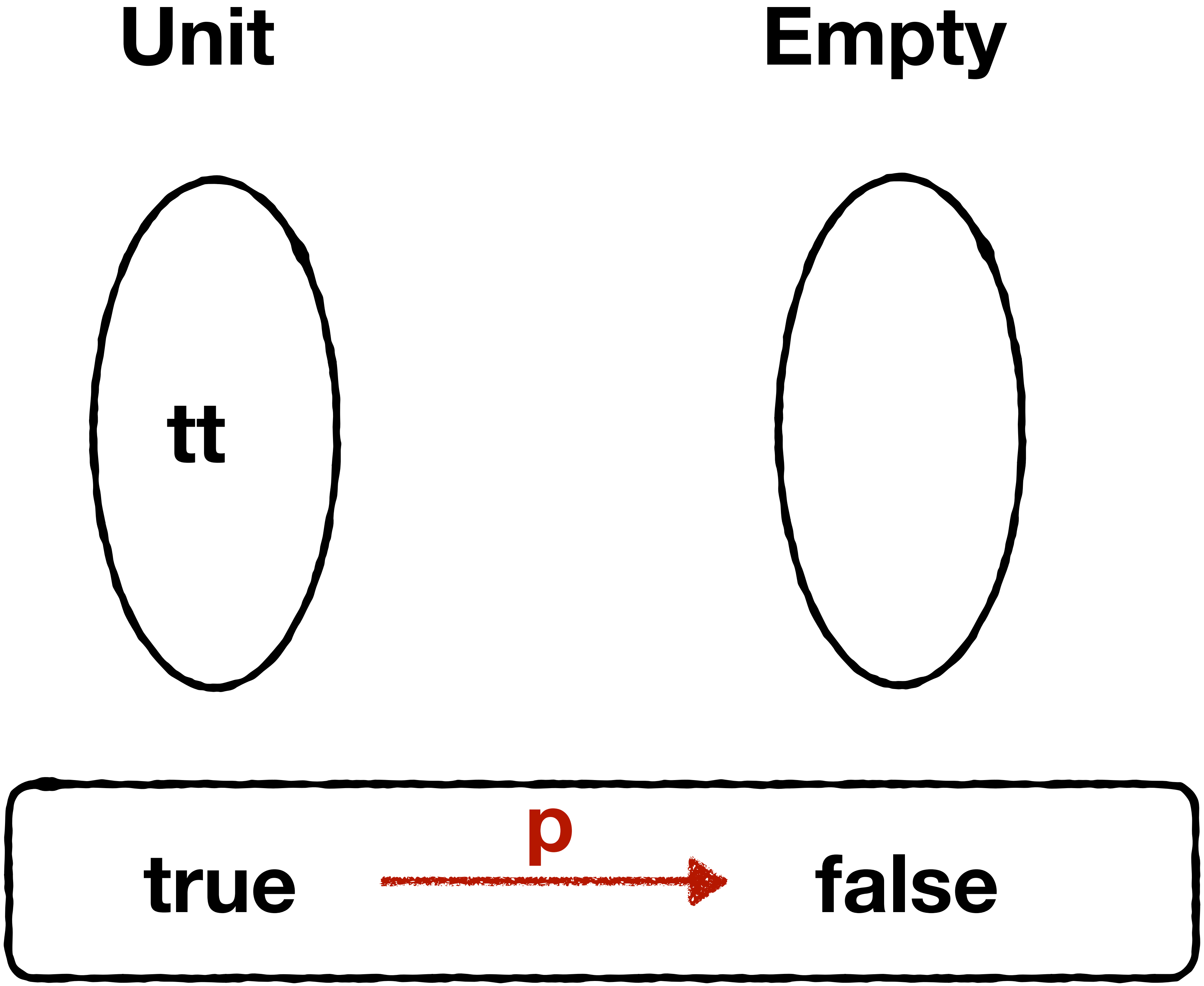
By constructing an element in

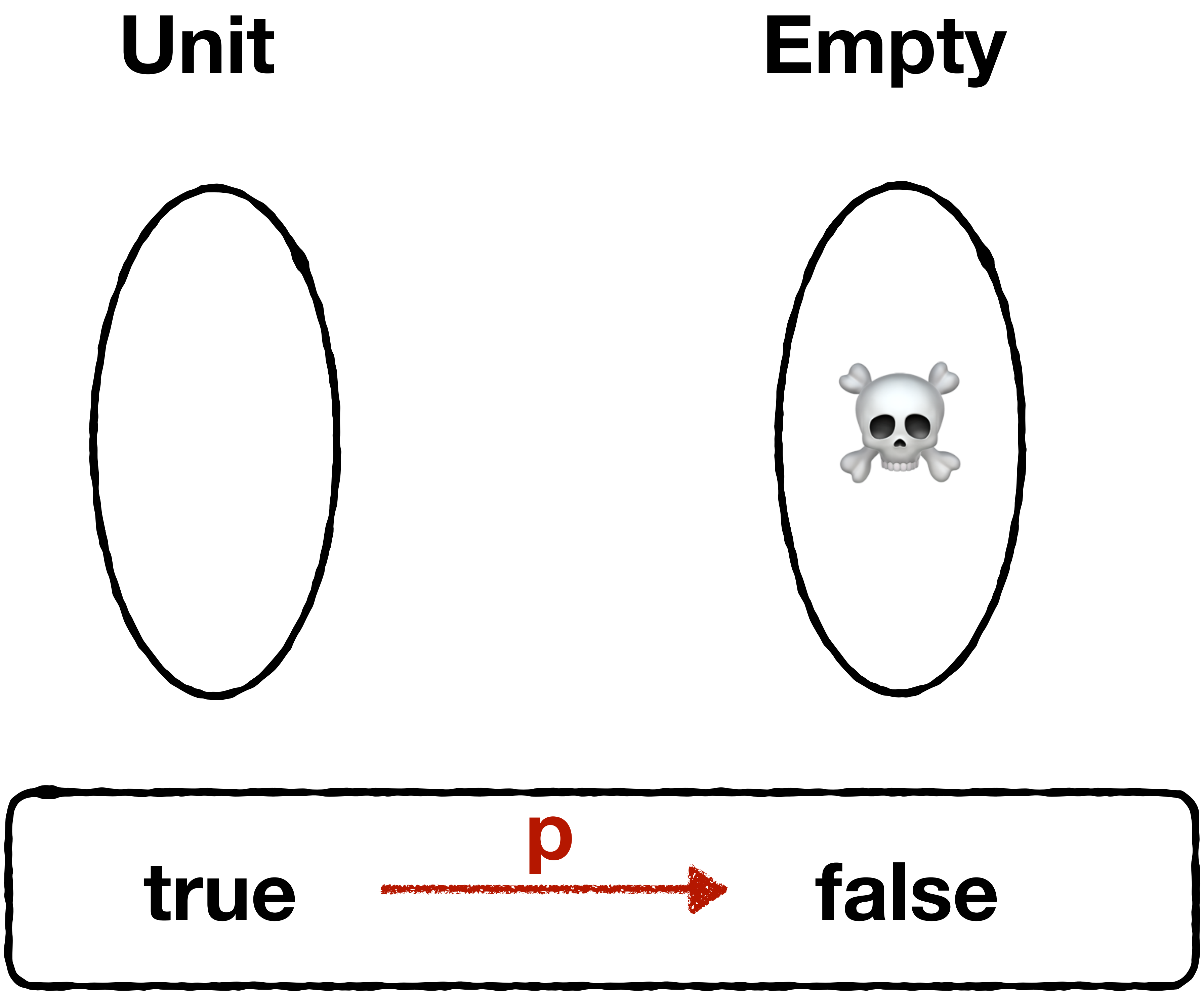
$P \to \text{Empty}$

(false = true) → Empty

```
Definition sanity : true = false → Empty :=
  fun (p : true = false) ⇒
    transport
      (ind_bool (fun _ ⇒ Type) Unit Empty)
      p
      tt.
```


Unit
Empty
tt
true
false


Unit
Empty
tt
true
p
false


Unit
Empty
true
p
false

We are not done!

- We left out precise rules!
- What about spheres, reals, groups, etc?
- Can paths be composed, or inverted?
- What is a path between paths?
- What is a path between types in **Type**?

Further material

- Euclid: *Elements*
- Daniel Grayson: *An introduction to univalent foundations for mathematicians* (arXiv:1711.01477)
- UniMath library
- Talk to people here!