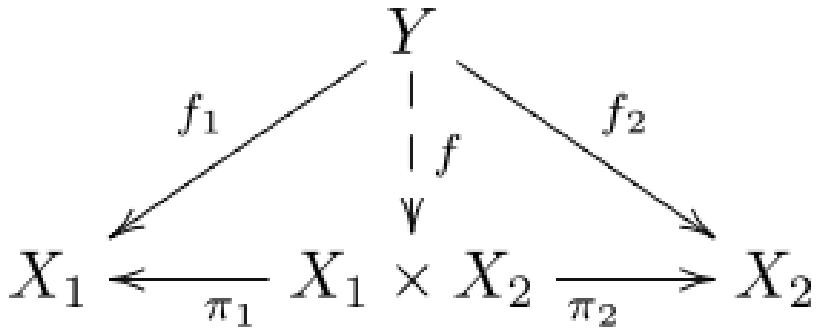
What are “Natural Numbers Objects”?

And what do they have to do with recursive arithmetic?

The setting: a Cartesian category

- A terminal object 1
- Binary products $A \times B$

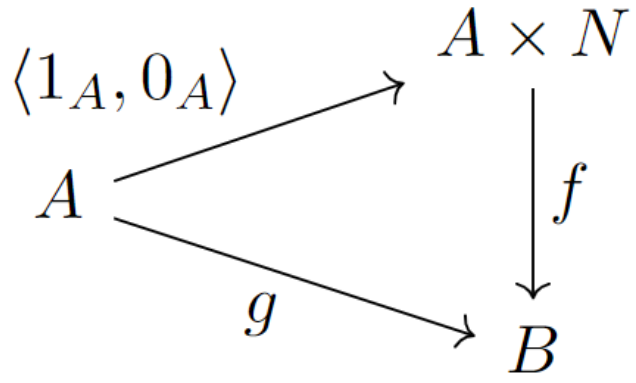


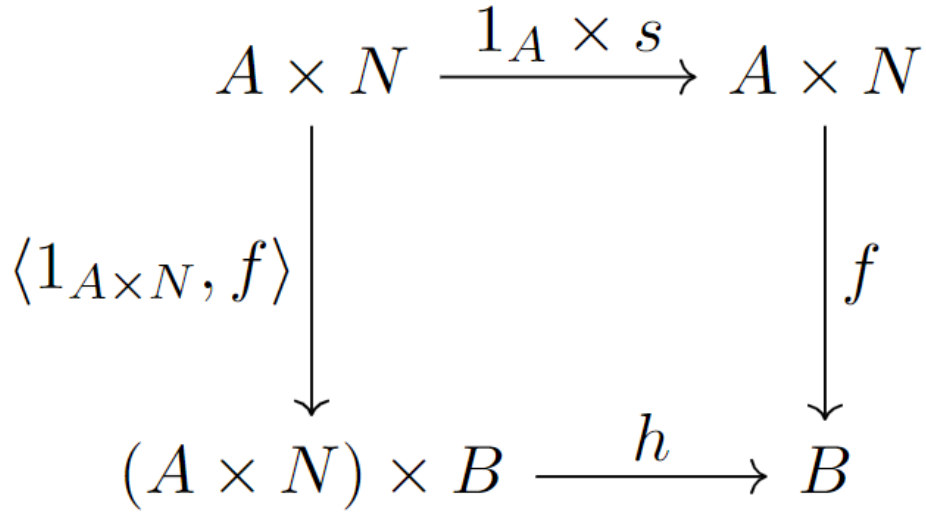
Natural Numbers Object (NNO)

- An object N
- Two arrows $1 \xrightarrow{0} N \xrightarrow{s} N$

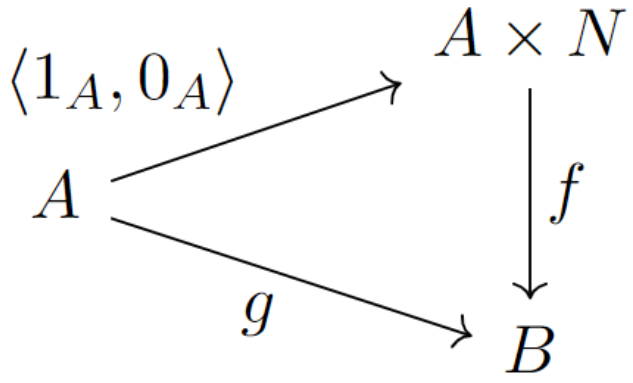
Natural Numbers Object (NNO)

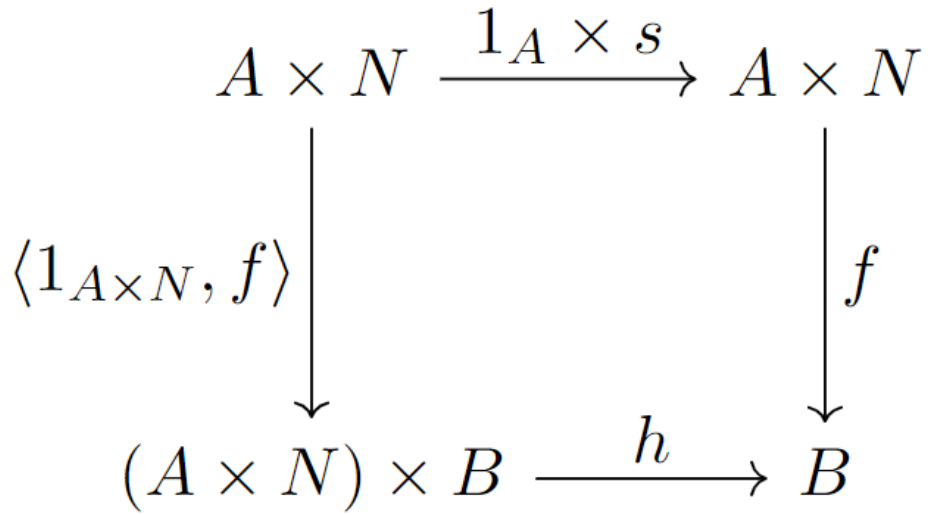
- An object N
- Two arrows $1 \xrightarrow{0} N \xrightarrow{s} N$
- For any arrows $A \xrightarrow{g} B$ and $(A \times N) \times B \xrightarrow{h} B$, there exists a unique arrow $A \times N \xrightarrow{f} B$ such that these diagrams commute:





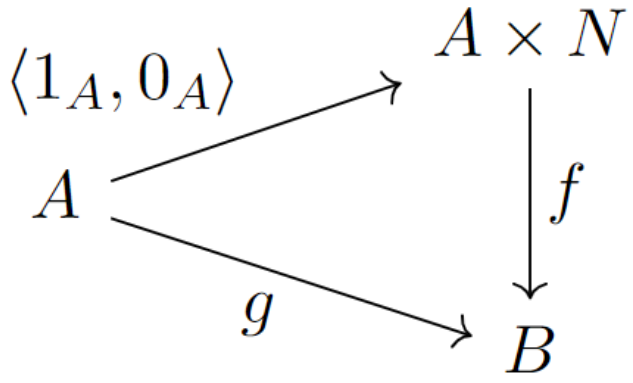
Huh?

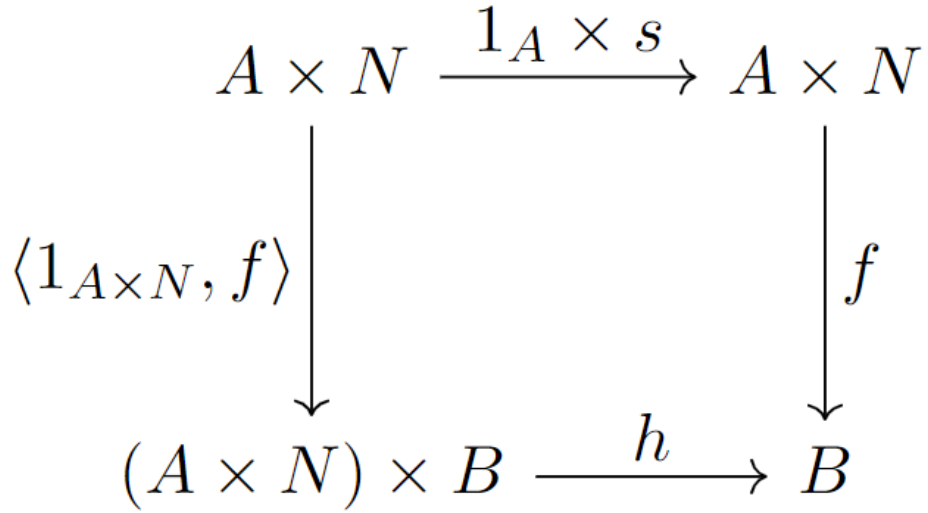

$A \times N$
$\langle 1_A, 0_A \rangle$
A
f
g
B


$A \times N$
$1_A \times s$
$A \times N$
$\langle 1_{A \times N}, f \rangle$
f
$(A \times N) \times B$
h
B

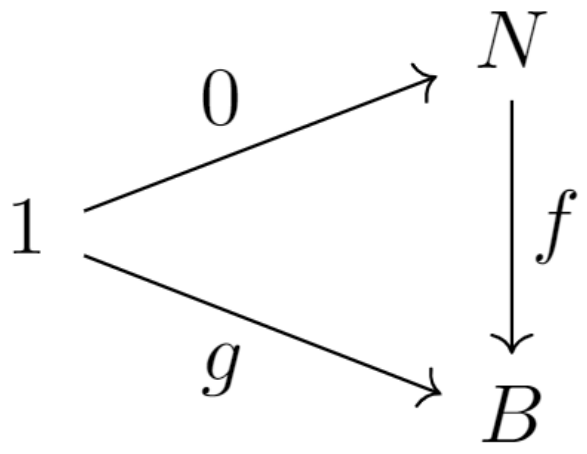
Huh?

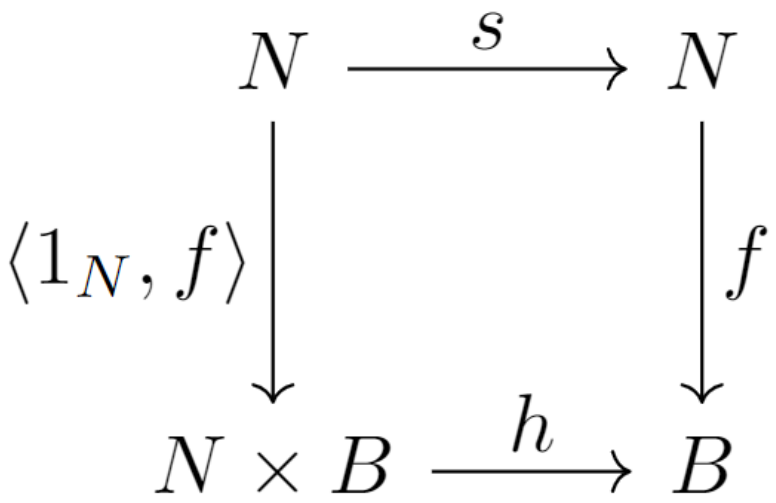
- Set $A = 1$


$A \times N$
$\langle 1_A, 0_A \rangle$
A
f
g
B


$A \times N$
$1_A \times s$
$A \times N$
$\langle 1_{A \times N}, f \rangle$
f
$(A \times N) \times B$
h
B

Huh?

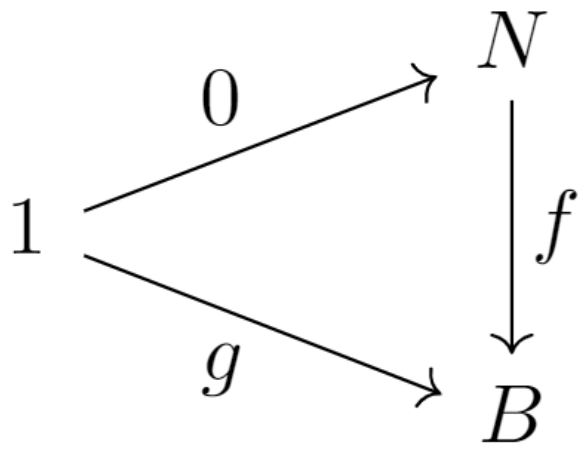


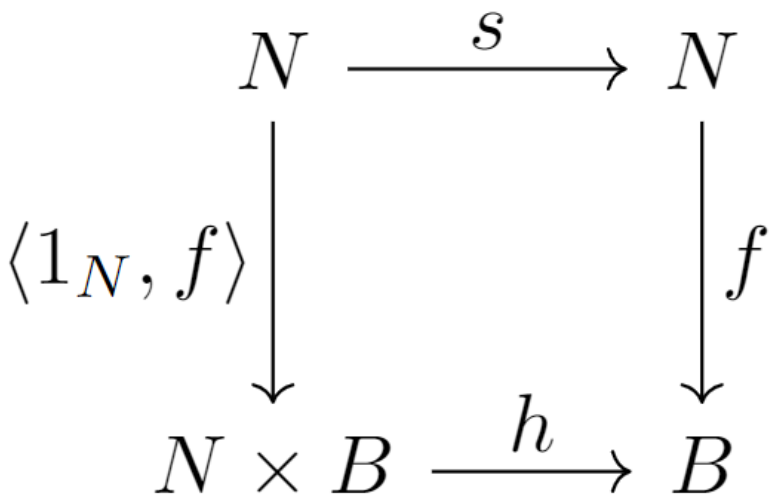


Huh?

- This reads:

$$f(0) = g \qquad f\big(s(n)\big) = h\big(n, f(n)\big)$$



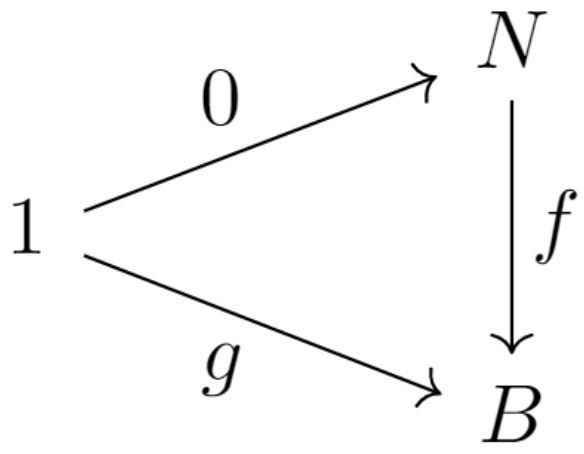


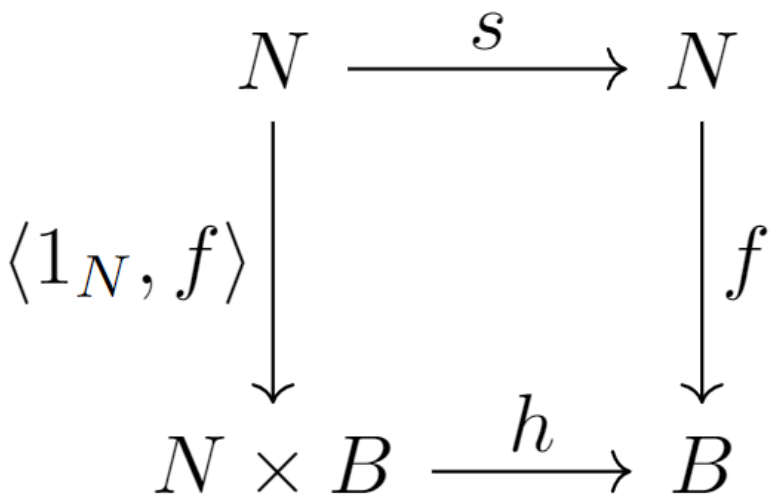
Huh?

- For example:

$$f(0) = 1 \qquad f\big(s(n)\big) = s(n) \cdot f(n)$$

This uniquely defines $f(n) = n!$

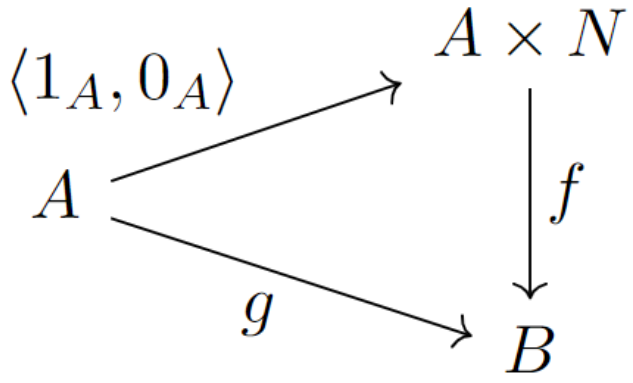


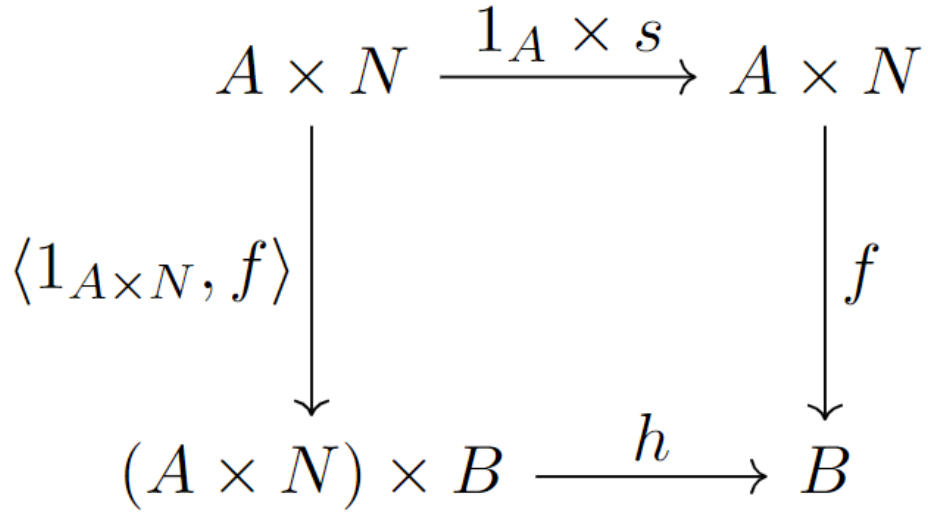


More generally

- Recursion with parameters:

$$f(a,0) = g(a) \qquad f\big(a,s(n)\big) = h\big(a,n,f(a,n)\big)$$

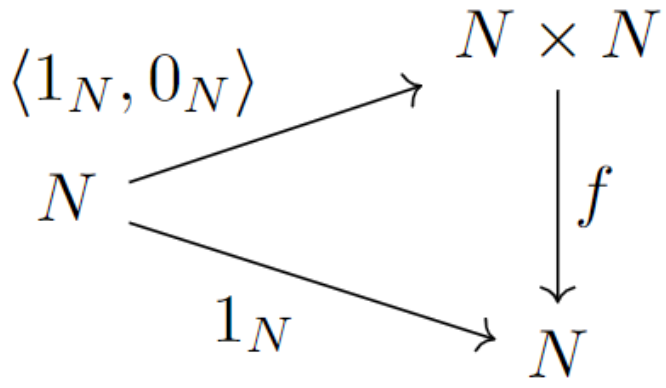


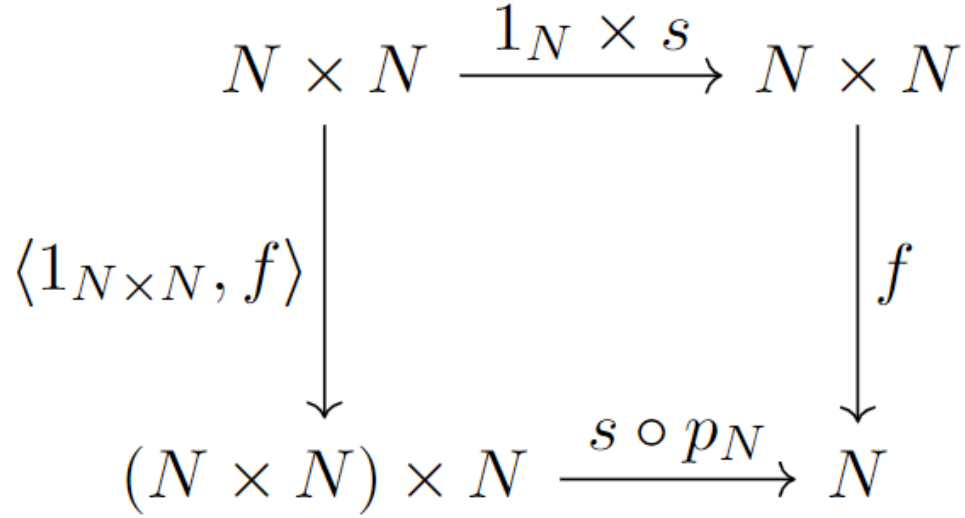


More generally

- Set $A = B = N$, $g = Id$, $h = s \circ p_N$.

$$f(m, 0) = m \qquad f\big(m, s(n)\big) = s\big(f(m, n)\big)$$



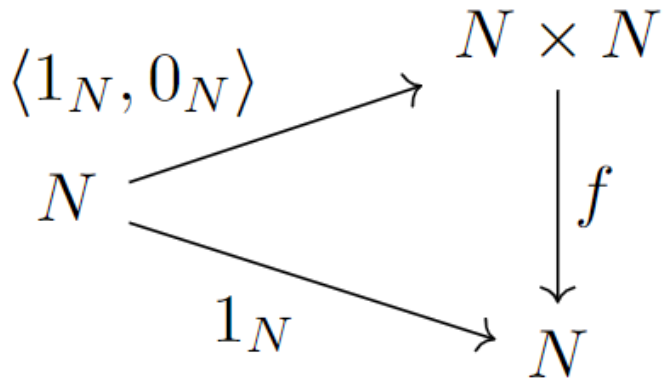


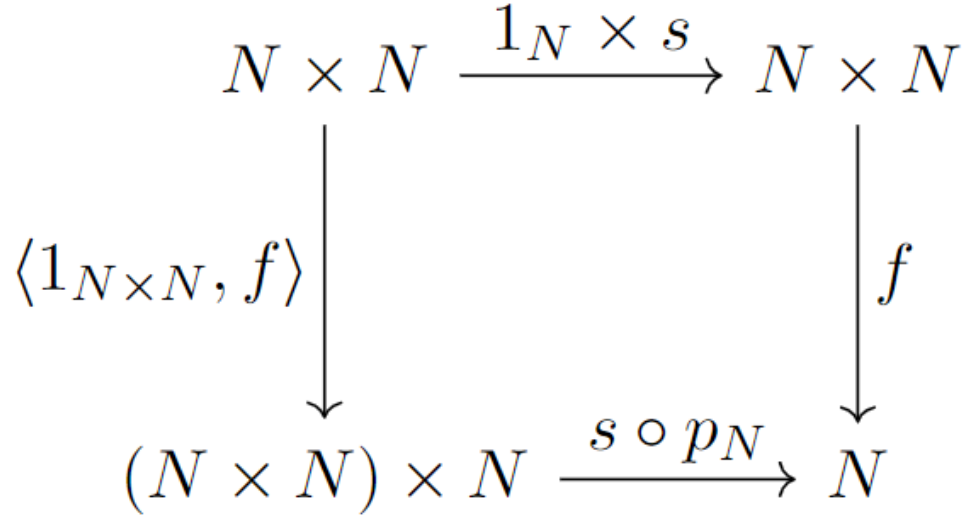
More generally

- Set $A = B = N$, $g = Id$, $h = s \circ p_N$.

- $f(m,0) = m$ $\qquad$ $f\big(m, s(n)\big) = s\big(f(m,n)\big)$

  $\mathrm{SUM}(m,0) = m$ $\qquad$ $\mathrm{SUM}\big(m, s(n)\big) = s\big(SUM(m,n)\big)$





Examples

Examples

- $\mathbb{N}$ is an NNO in the category of sets and functions

Examples

- $\mathbb{N}$ is an NNO in the category of sets and functions

- Let $\mathcal{C}$ be a cartesian closed category with coproducts. Let $N = \coprod_{n \in \mathbb{N}} 1$.

Notable example: primitive recursive arithmetic

Notable example: primitive recursive arithmetic

- Consider the following subcategory of **Set**:
  - Objects are powers of $\mathbb{N}$
  - Arrows are primitive recursive functions $\mathbb{N}^n \to \mathbb{N}^m$.
  - Call it **PRIM**

Notable example: primitive recursive arithmetic

- Basic functions $\mathbb{N}^n \to \mathbb{N}$:
  - $Z(x) = 0$
  - $S(x) = x + 1$
  - $Proj_k^n(x_1, \ldots, x_n) = x_k$

Notable example: primitive recursive arithmetic

- Basic functions $\mathbb{N}^n \to \mathbb{N}$:
  - $Z(x) = 0$
  - $S(x) = x + 1$
  - $Proj_k^n(x_1, \dots, x_n) = x_k$

- Generalized composition $f = g \circ \langle h_1, \dots, h_n \rangle$
  - $f(x_1, \dots, x_m) = g\big(h_1(x_1, \dots, x_m), \dots, h_n(x_1, \dots, x_m)\big)$

Notable example: primitive recursive arithmetic

- Basic functions $\mathbb{N}^n \to \mathbb{N}$:
  - $Z(x) = 0$
  - $S(x) = x + 1$
  - $Proj_k^n(x_1, \dots, x_n) = x_k$

- Generalized composition $f = g \circ \langle h_1, \dots, h_n \rangle$
  - $f(x_1, \dots, x_m) = g\big(h_1(x_1, \dots, x_m), \dots, h_n(x_1, \dots, x_m)\big)$

- Primitive recursion $f = PR[g, h]$
  - $f(x_1, \dots, x_m, 0) = g(x_1, \dots, x_m)$
  - $f\big(x_1, \dots, x_m, S(k)\big) = h\big(x_1, \dots, x_m, k, f(x_1, \dots, x_m, k)\big)$

Notable example: primitive recursive arithmetic

- These are functions $\mathbb{N}^n \to \mathbb{N}$. For functions $\mathbb{N}^n \to \mathbb{N}^m$, we use:

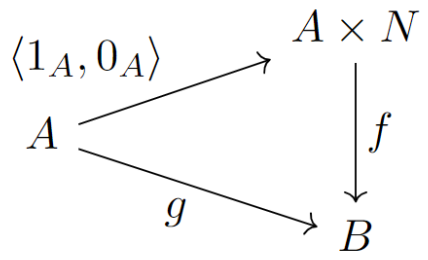
$$\langle f_1, \ldots, f_m \rangle : \ \mathbb{N}^n \to \mathbb{N}^m$$

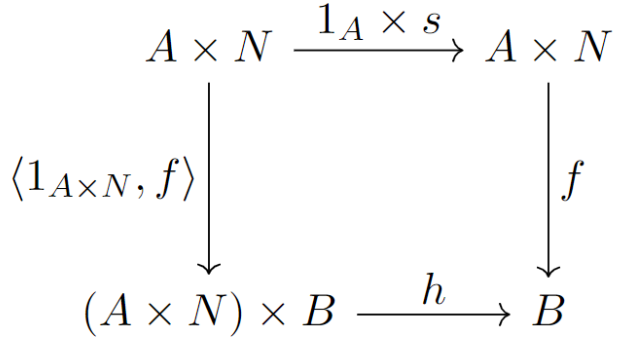
Notable example: primitive recursive arithmetic

- These are functions $\mathbb{N}^n \to \mathbb{N}$. For functions $\mathbb{N}^n \to \mathbb{N}^m$, we use:

$$\langle f_1, \ldots, f_m \rangle : \mathbb{N}^n \to \mathbb{N}^m$$

- The powers of $\mathbb{N}$, together with these functions, give us a category with NNO!





Recap

- We've seen the definition
- We've seen examples

Recap

- We’ve seen the definition
- We’ve seen examples
- How to study categories with NNO?

Make it a category!

- Objects: Cartesian categories with NNO

Make it a category!

- Objects: Cartesian categories with NNO
- Arrows: functors which preserve products and NNO

$$A \xleftarrow{p_A} A \times B \xrightarrow{p_B} B \quad \Rightarrow \quad F(A) \xleftarrow{F(p_A)} F(A \times B) \xrightarrow{F(p_B)} F(B)$$

$$1 \xrightarrow{0} N \xrightarrow{s} N \quad \Rightarrow \quad 1 \xrightarrow{F(0)} F(N) \xrightarrow{F(s)} F(N)$$

The category of cart. cats with NNO

The category of cart. cats with NNO

The category of cart. cats with NNO

- What is the initial object?

The initial object

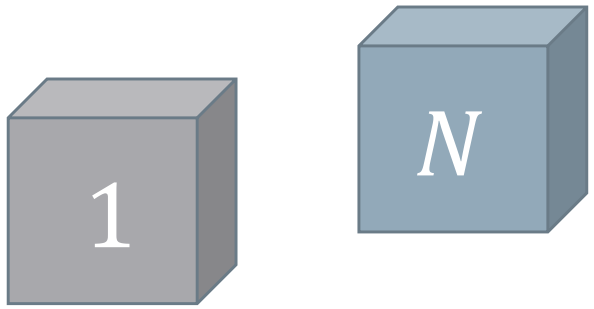
(in the category of cartesian categories with NNO)

- Mindset: freely generate our category using the minimal starting building blocks.

The initial object

(in the category of cartesian categories with NNO)

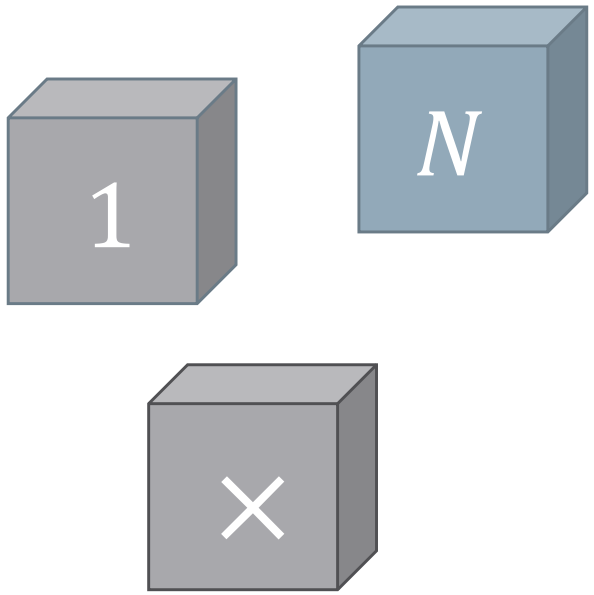
- Mindset: freely generate our category using the minimal starting building blocks.



The initial object

(in the category of cartesian categories with NNO)

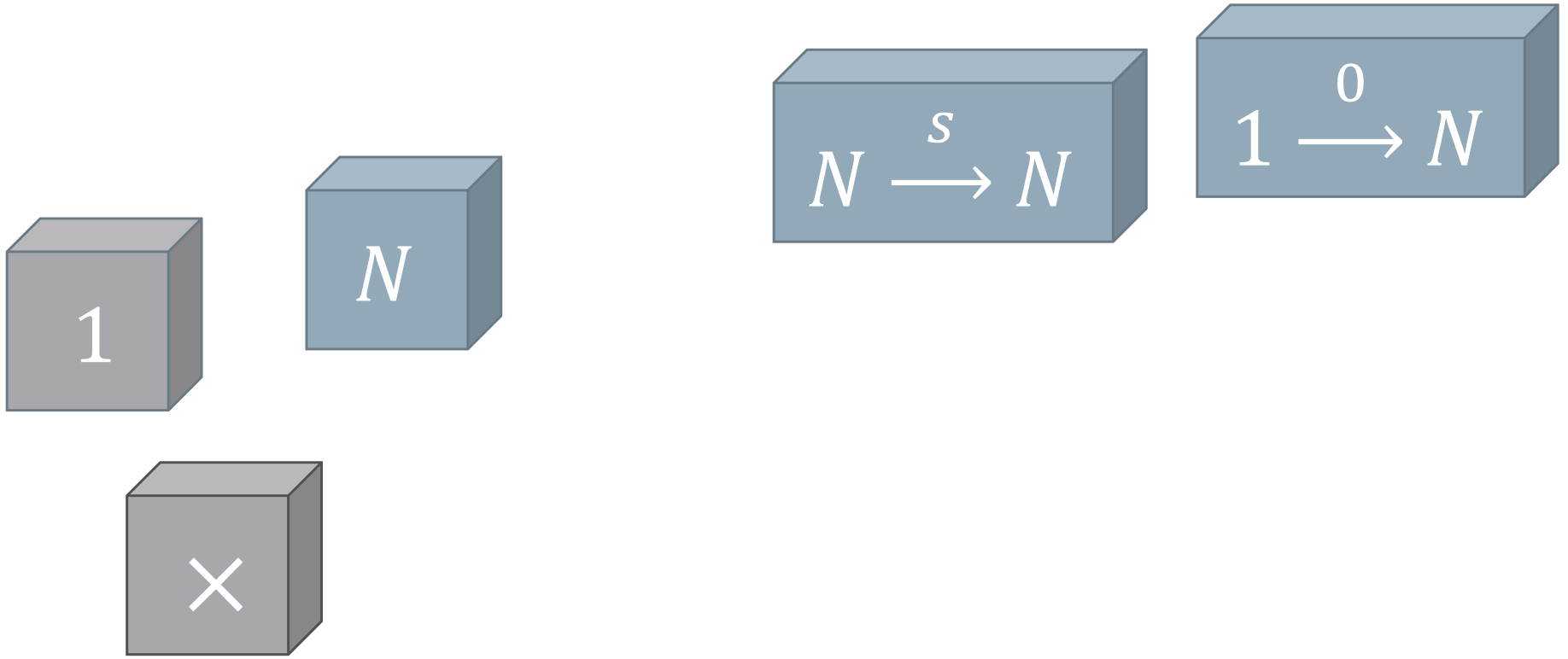
- Mindset: freely generate our category using the minimal starting building blocks.



The initial object

(in the category of cartesian categories with NNO)

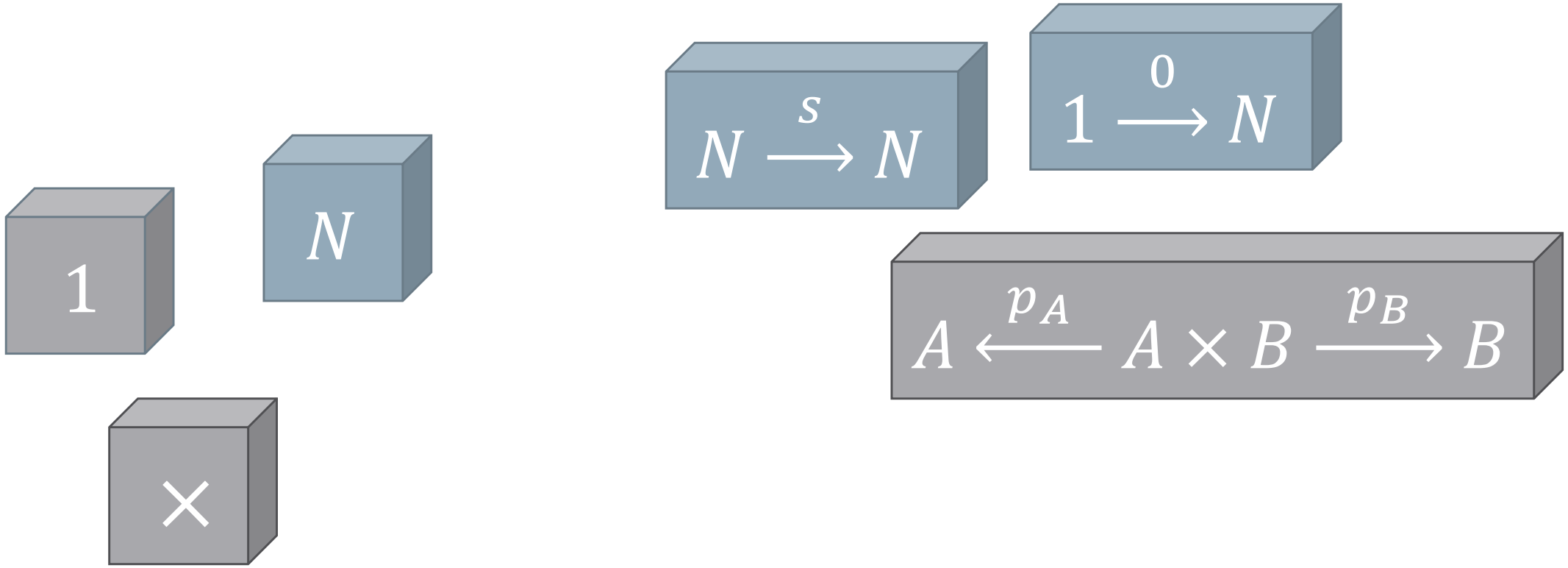
- Mindset: freely generate our category using the minimal starting building blocks.



The initial object

(in the category of cartesian categories with NNO)

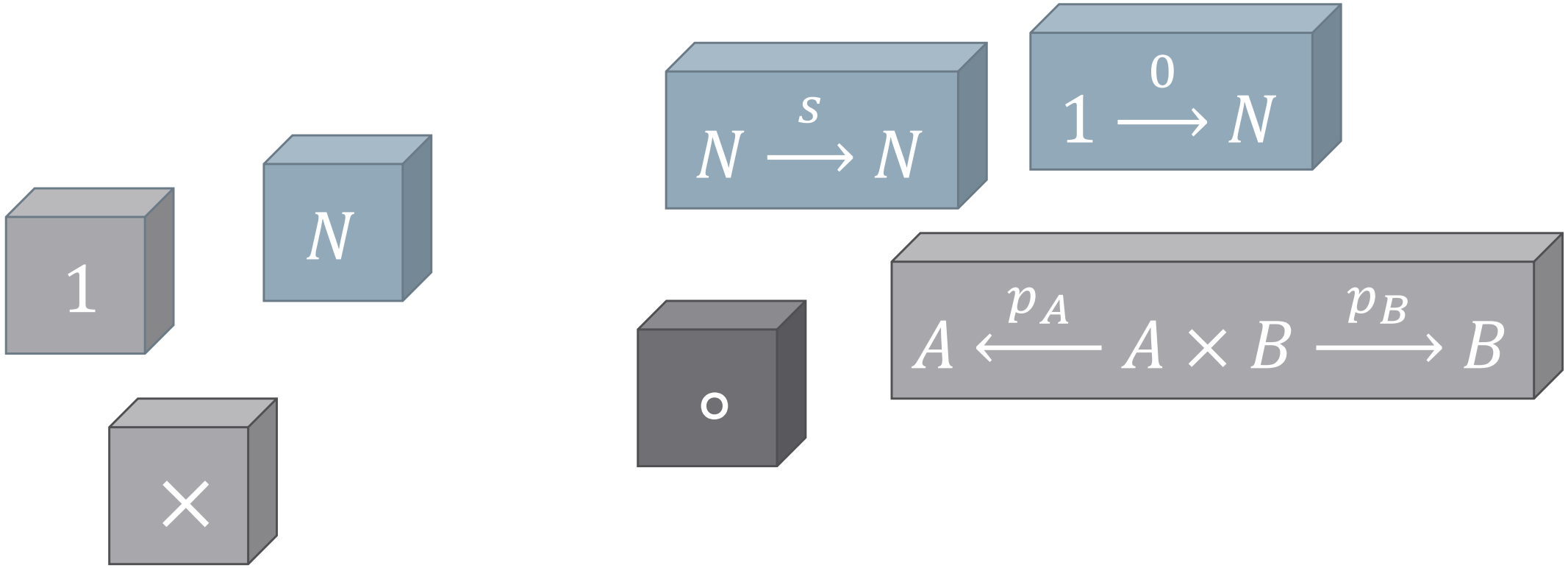
- Mindset: freely generate our category using the minimal starting building blocks.



The initial object

(in the category of cartesian categories with NNO)

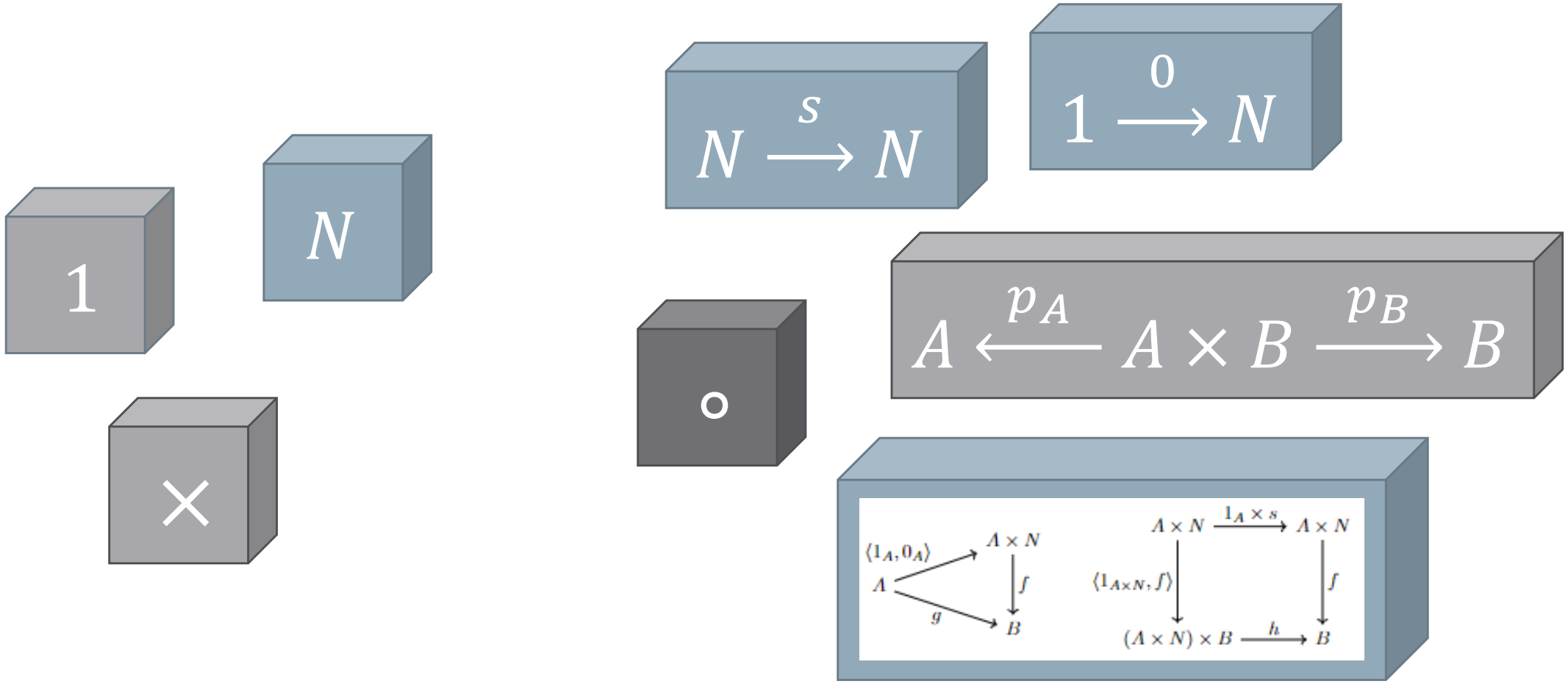
- Mindset: freely generate our category using the minimal starting building blocks.



The initial object

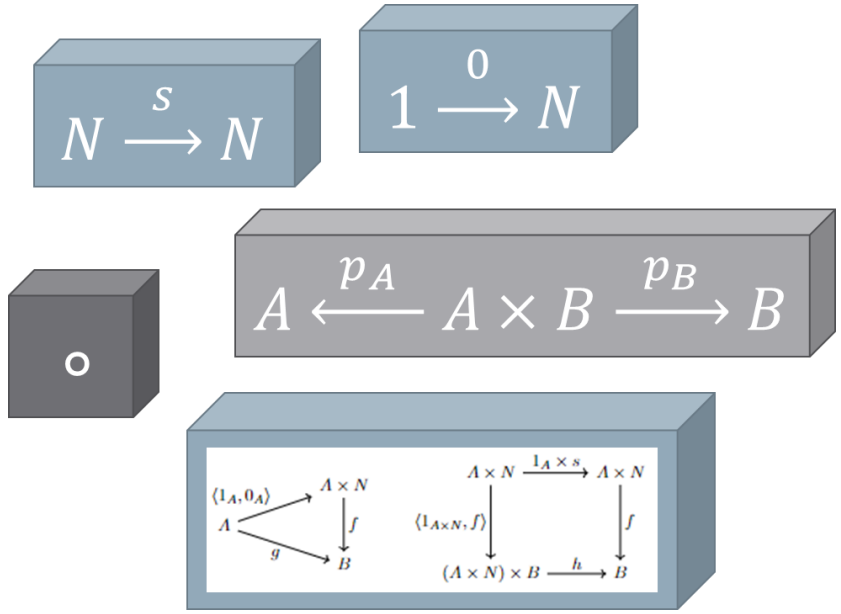
(in the category of cartesian categories with NNO)

- Mindset: freely generate our category using the minimal starting building blocks.



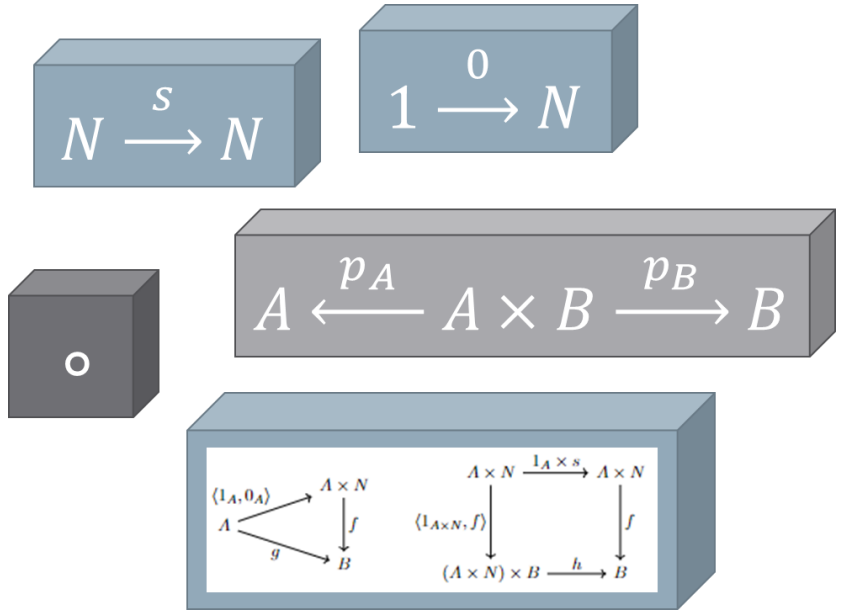
This sounds familiar…

- Objects are powers of N
- Arrows are built out of:
  - Zero, Successor, Projections
  - Composition
  - Induction



This sounds familiar…

- Objects are powers of N
- Arrows are built out of:
  - Zero, Successor, Projections
  - Composition
  - Induction



- Is the initial object **PRIM**??

NO!

NO!

- The arrows are not built freely!

NO!

- The arrows are not built freely!

- We need a functor $F : PRIM \to \mathcal{C}$.
- Given $f : \mathbb{N}^n \to \mathbb{N}$, we'd define $F(f)$ by induction on f.
- Why does $f = g$ guarantee $F(f) = F(g)$??

What do we do instead?

- We need a different notion of equality…

What do we do instead?

- We need a different notion of equality…

- We have to turn to **Goodstein's arithmetic**!

Goodstein's arithmetic

- A formal logical system for doing arithmetic

Goodstein's arithmetic

- A formal logical system for doing arithmetic

$$x + 0 = x$$
$$x + (y + 1) = (x + y) + 1$$

$$F(x, 0) = a(x)$$
$$F(x, Sy) = b(x, y, F(x, y))$$

$$\mathbf{Prod}(x, 0) = 0$$
$$\mathbf{Prod}(x, Sy) = \mathbf{Sum}(x, \mathbf{Prod}(x, y))$$

$$x + 0 = x$$
$$Sx + 0 = Sx$$
$$Sx = Sx$$
$$S(x + 0) = Sx$$
$$x + Sy = S(x + y)$$
$$Sx + Sy = S(Sx + y)$$
$$S(x + Sy) = S(x + Sy)$$
$$S(x + Sy) = SS(x + y)$$
$$Sx + y = S(x + y).$$

Goodstein's arithmetic

- A formal logical system for doing arithmetic

$$x + 0 = x$$
$$x + (y + 1) = (x + y) + 1$$

$$F(x, 0) = a(x)$$
$$F(x, Sy) = b(x, y, F(x, y))$$

$$\mathbf{Prod}(x, 0) = 0$$
$$\mathbf{Prod}(x, Sy) = \mathbf{Sum}(x, \mathbf{Prod}(x, y))$$

- Equality here is not set-theoretic!

$$x + 0 = x$$
$$Sx + 0 = Sx$$
$$Sx = Sx$$
$$S(x + 0) = Sx$$
$$x + Sy = S(x + y)$$
$$Sx + Sy = S(Sx + y)$$
$$S(x + Sy) = S(x + Sy)$$
$$S(x + Sy) = SS(x + y)$$
$$Sx + y = S(x + y).$$

Recap

- We've defined NNO and seen examples
- We've defined the category of categories with NNO
- We're searching for the initial object
- **PRIM** doesn't work
- So: we turn to Goodstein!

Going forward…

- We'll see how Goodstein's system is defined
- Then, we'll use it to build a cartesian category with NNO
- This will be our initial object!

Goodstein's arithmetic: What are the components?

- We need to define:
  - Terms
  - Equality between terms

$$\mathbf{Prod}(x, 0) = 0$$
$$\mathbf{Prod}(x, Sy) = \mathbf{Sum}(x, \mathbf{Prod}(x, y))$$

Goodstein's arithmetic: terms

- Terms: on either side of an equality

$$\mathbf{Prod}(x, 0) = 0$$
$$\mathbf{Prod}(x, Sy) = \mathbf{Sum}(x, \mathbf{Prod}(x, y))$$

Goodstein's arithmetic: terms

- Terms: on either side of an equality
  - 0
  - Variables
  - ?

$$\mathbf{Prod}(x, 0) = 0$$
$$\mathbf{Prod}(x, Sy) = \mathbf{Sum}(x, \mathbf{Prod}(x, y))$$

Goodstein's arithmetic: terms

- **Terms: on either side of an equality**
- **Functions: can be applied to terms**

$$\mathbf{Prod}(x, 0) = 0$$
$$\mathbf{Prod}(x, Sy) = \mathbf{Sum}(x, \mathbf{Prod}(x, y))$$

Goodstein's arithmetic: terms

- **Terms: on either side of an equality**
  - 0 is a term
  - Variables are terms
  - If f is a function of arity n, and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.
- **Functions: can be applied to terms**

$$\mathbf{Prod}(x, 0) = 0$$
$$\mathbf{Prod}(x, Sy) = \mathbf{Sum}(x, \mathbf{Prod}(x, y))$$

Goodstein's arithmetic: terms

- **Terms: on either side of an equality**
  - 0 is a term
  - Variables are terms
  - If f is a function of arity n, and $t_1, \dots, t_n$ are terms, then $f(t_1, \dots, t_n)$ is a term.
- **Functions: can be applied to terms**
  - S is a function of arity 1
  - If t is a term containing only variables from $x_1, \dots, x_n$, then we can form a function f of arity n defined by $f(x_1, \dots, x_n) = t$.
  - Induction

$$\mathbf{Prod}(x, 0) = 0$$
$$\mathbf{Prod}(x, Sy) = \mathbf{Sum}(x, \mathbf{Prod}(x, y))$$

An example: defining addition

1. x is a term.
2. $S(z)$ is a term.

An example: defining addition

1. x is a term.
2. $S(z)$ is a term.

Then we can define $f(x, y)$ by:

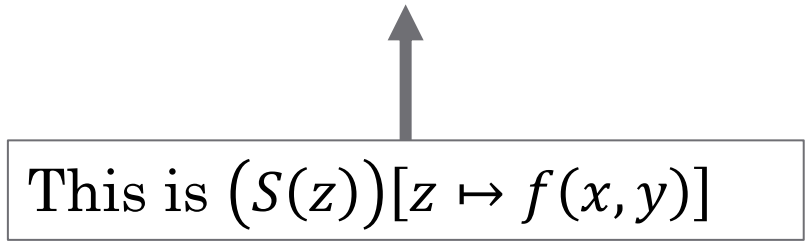
- $f(x, 0) = x$
- $f\big(x, S(y)\big) = S\big(f(x, y)\big)$

An example: defining addition

1. x is a term.
2. $S(z)$ is a term.

Then we can define $f(x, y)$ by:

- $f(x, 0) = x$
- $f\big(x, S(y)\big) = S\big(f(x, y)\big)$



An example: defining addition

1. x is a term.
2. $S(z)$ is a term.

Then we can define $f(x, y)$ by:

- $f(x, 0) = x$
- $f\big(x, S(y)\big) = S\big(f(x, y)\big)$

This is $\big(S(z)\big)[z \mapsto f(x, y)]$

I.e.:

- $x + 0 = x$
- $x + S(y) = S(x + y)$

Goodstein's arithmetic: terms

- **Terms**
  - 0 is a term
  - Variables are terms
  - If f is a function of arity n, and $t_1, \dots, t_n$ are terms, then $f(t_1, \dots, t_n)$ is a term.
- **Functions**
  - S is a function of arity 1
  - If t is a term containing only variables from $x_1, \dots, x_n$, then we can form a function f of arity n defined by $f(x_1, \dots, x_n) = t$.
  - Induction:
    - If T_b is a term containing only variables from $x_1, \dots, x_n$;
    - If T_i is a term containing only variables from $k, z, x_1, \dots, x_n$;
    - Then we can form f of arity $n + 1$ defined by $f(0, x_1, \dots, x_n) = T_b$
      $f(S(k), x_1, \dots, x_n) = T_i[z \mapsto f(k, x_1, \dots, x_n)]$

Goodstein's arithmetic: equality

- Equality is a relation on terms.
- It is defined by its rules of inference.

Goodstein's arithmetic: equality

- Equality is a relation on terms.
- It is defined by its rules of inference.

Rule 1: $$\frac{A = B \qquad A = C}{B = C}$$

Rule 2: $$\frac{A = B}{F[x \mapsto A] = F[x \mapsto B]}$$

Rule 3: $$\frac{F = G}{F[x \mapsto A] = G[x \mapsto A]}$$

Goodstein's arithmetic: equality

- Equality is a relation on terms.
- It is defined by its rules of inference.

Rule 1: $\dfrac{A = B \qquad A = C}{B = C}$

Rule 2: $\dfrac{A = B}{F[x \mapsto A] = F[x \mapsto B]}$

Rule 3: $\dfrac{F = G}{F[x \mapsto A] = G[x \mapsto A]}$

Rule 4: Function definitions

Rule 5: Uniqueness of inductive definitions

Equality: function definitions

- Suppose f is a function defined explicitly by $f(x_1, \ldots, x_n) = t$. Then:

Rule 4a: $$\frac{}{f(x_1, \ldots, x_n) = t}$$

- Suppose f is a function defined inductively with T_b and T_i. Then:

Rule 4b: $$\frac{}{f(0, x_1, \ldots, x_n) = T_b}$$

Rule 4c: $$\frac{}{f(S(k), x_1, \ldots, x_n) = T_i[z \mapsto f(k, x_1, \ldots, x_n)]}$$

Equality: induction uniqueness

- If f, g are functions and T_i is a term, then:

Rule 5:
$$\frac{\begin{array}{c} f(0, x_1, \ldots, x_n) = g(0, x_1, \ldots, x_n) \\ f(S(k), x_1, \ldots, x_n) = T_i[z \mapsto f(k, x_1, \ldots, x_n)] \\ g(S(k), x_1, \ldots, x_n) = T_i[z \mapsto g(k, x_1, \ldots, x_n)] \end{array}}{f(k, x_1, \ldots, x_n) = g(k, x_1, \ldots, x_n)}$$

Formalization in Coq

- Coq is an interactive theorem prover, which uses the theory of the calculus of inductive constructions.

Coq: terms and functions

- The definitions of terms and functions are intertwined. In Coq, we need to define them simultaneously! Then specify which is which.

```
Inductive PRA_object : Type :=
| var (x : variable)
| zero
| succ_func
| explicit_func (lvar : list variable) (def : PRA_object)
| inductive_func (k x : variable) (lvar : list variable) (base ind : PRA_object)
| func_app (func : PRA_object) (lterm : list PRA_object).
```

Coq: terms and functions

- The definitions of terms and functions are intertwined. In Coq, we need to define them simultaneously! Then specify which is which.

```
Inductive PRA_object : Type :=
| var (x : variable)
| zero
| succ_func
| explicit_func (lvar : list variable) (def : PRA_object)
| inductive_func (k x : variable) (lvar : list variable) (base ind : PRA_object)
| func_app (func : PRA_object) (lterm : list PRA_object).
```

```
Definition is_func_b (f : PRA_object) :=
  match f with
  | succ_func => true
  | explicit_func _ _ => true
  | inductive_func _ _ _ _ _ => true
  | _ => false
  end.
```

```
Definition is_term_b (t : PRA_object) :=
  match t with
  | var _ => true
  | zero => true
  | func_app _ _ => true
  | _ => false
  end.
```

Coq: terms and functions

- We need to specify which terms and functions are valid.
- (We also need to define arity!)

```
Fixpoint valid_object_b (f : PRA_object) :=
  match f with
  | var x => true
  | zero => true
  | succ_func => true

  | explicit_func lvar def => (valid_object_b def) &&
                              (is_term_b def) &&
                              (has_no_repeats_b lvar) &&
                              (only_vars_from_b lvar def)

  | inductive_func k x lvar base ind =>
                       (valid_object_b base) && (valid_object_b ind) &&
                       (is_term_b base) && (is_term_b ind) &&
                       (has_no_repeats_b (k :: x :: lvar)) &&
                       (only_vars_from_b lvar base) &&
                       (only_vars_from_b (k :: x :: lvar) ind)

  | func_app func lterm => (valid_object_b func) &&
                           (is_func_b func) &&
                           (forallb valid_object_b lterm) &&
                           (forallb is_term_b lterm) &&
                           (object_arity func =? length lterm)
  end.
```

Coq: term equality

- Equality is defined inductively.
- All the equality conditions from before are here!

```
(* Equality is denoted =t= *)
Inductive PRA_term_eq : PRA_object -> PRA_object -> Prop :=

(* explicit function definition *)
| (explicit_func lvar def) << lvar >> =t= def

(* inductive function definition, base case *)
| (inductive_func k x lvar base ind) << 0 :: lvar >> =t= base

(* inductive function definition, inductive case *)
(* Set I = (inductive_func k x lvar base ind) *)
| I << S << k >> :: lvar >> =t= ind [x |-> I << k :: lvar >>]

(* A =t= B and A =t= C imply B =t= C *)
| (A =t= B) -> (A =t= C) -> (B =t= C)

(* A =t= B implies F(A) =t= F(B) for any F(x) *)
| (A =t= B) -> F [x |-> A] =t= F [x |-> B]

(* F(x) =t= G(x) implies F(A) =t= G(A) for any A *)
| (F =t= G) -> F [x |-> A] =t= G [x |-> A]

(* Induction uniqueness *)
| (f1 << 0 :: lvar >> =t= f2 << 0 :: lvar >>) ->
  (f1 << S << k >> :: lvar >> =t= ind [x |-> f1 << k :: lvar >>) ->
  (f2 << S << k >> :: lvar >> =t= ind [x |-> f2 << k :: lvar >>) ->
  (f1 << k :: lvar >> =t= f2 << k :: lvar >>)
```

Coq: some difficulties

Does $\dfrac{F(x) = G(x)}{F(T) = G(T)}$ imply $\dfrac{F(x_1, \ldots, x_n) = G(x_1, \ldots, x_n))}{F(T_1, \ldots, T_n) = G(T_1, \ldots, T_n)}$?

Coq: some difficulties

Does $\dfrac{F(x) = G(x)}{F(T) = G(T)}$ imply $\dfrac{F(x_1, \ldots, x_n) = G(x_1, \ldots, x_n))}{F(T_1, \ldots, T_n) = G(T_1, \ldots, T_n)}$?

$$\dfrac{\dfrac{F(x_1, x_2) = G(x_1, x_2)}{F(T_1, x_2) = G(T_1, x_2)}}{F(T_1[x_2 \mapsto T_2], T_2) = G(T_1[x_2 \mapsto T_2], T_2)}$$

Wait, wasn’t this about category theory?

Wait, wasn't this about category theory?

- We want to use Goodstein's arithmetic to form a category, which we'll call PRA.

Wait, wasn't this about category theory?

- We want to use Goodstein's arithmetic to form a category, which we'll call PRA.
- Objects: formal objects N^n for $n \geq 0$ (denote N^0 by 1).

Wait, wasn't this about category theory?

- We want to use Goodstein's arithmetic to form a category, which we'll call PRA.
- Objects: formal objects N^n for $n \geq 0$ (denote N^0 by 1).
- Arrows: an arrow $f : N^n \to N^m$ is a list $[f_1, \ldots, f_m]$ where each f_i is a **function** in Goodstein's arithmetic.

Wait, wasn't this about category theory?

- We want to use Goodstein's arithmetic to form a category, which we'll call PRA.
- Objects: formal objects N^n for $n \geq 0$ (denote N^0 by 1).
- Arrows: an arrow $f : N^n \to N^m$ is a list $[f_1, \dots, f_m]$ where each f_i is a **function** in Goodstein's arithmetic.
- Two functions f, g are defined to be equal if they are both of the same arity n and $f(x_1, \dots, x_n) = g(x_1, \dots, x_n)$ as terms.

Wait, wasn't this about category theory?

- We want to use Goodstein's arithmetic to form a category, which we'll call PRA.
- Objects: formal objects N^n for $n \geq 0$ (denote N^0 by 1).
- Arrows: an arrow $f : N^n \to N^m$ is a list $[f_1, \dots, f_m]$ where each f_i is a **function** in Goodstein's arithmetic.
- Two functions f, g are defined to be equal if they are both of the same arity n and $f(x_1, \dots, x_n) = g(x_1, \dots, x_n)$ as terms.
- It might be that we can prove $f(S^k 0) = g(S^k 0)$ for each k, but we CAN'T prove $f(x) = g(x)$!

PRA in Coq

- We define function equality, and then objects and arrows.

```
(* Function equality is denoted =f= *)
Inductive PRA_func_eq : PRA_object -> PRA_object -> Prop :=
| (has_no_repeats lvar) ->
  (f1 << lvar >> =t= f2 << lvar >>) ->
  (f1 =f= f2)
```

```
Inductive object : Type :=
| ob (n : nat).

Inductive arrow : Type :=
| ar (n : nat) (l : list PRA_object).
```

```
Definition valid_arrow (a : arrow) :=
  match a with
  | ar n l => forall f, (In f l) -> ((valid_func f) /\ (arity f n))
  end.
```

PRA in Coq

- To form a category, we need to define the source and target maps, as well as composition.

```
Definition source (a : arrow) :=
  match a with
  | ar n l => ob n
  end.
```

```
Definition target (a : arrow) :=
  match a with
  | ar n l => ob (length l)
  end.
```

```
Definition composition (f g : arrow) : arrow :=
(* Complicated *).

Notation "f @ g" := (composition f g).
```

PRA in Coq

- Finally, we also need to prove basic properties:
  - Source, target, composition are preserved by equality
  - Identity arrows exist
  - Composition is associative

```
Theorem composition_eq : forall (f1 f2 g1 g2 : arrow),
  (target f1 = source g1) ->
  (f1 == f2) -> (g1 == g2) -> (f1 @ g1) == (f2 @ g2).
```

```
Theorem composition_assoc : forall (f g h : arrow),
  (valid_arrow f) -> (valid_arrow g) -> (valid_arrow h) ->
  (target f = source g) -> (target g = source h) ->
  (f @ (g @ h)) == ((f @ g) @ h).
```

PRA in Coq

- Finally, we also need to prove basic properties:
  - Source, target, composition are preserved by equality
  - Identity arrows exist
  - Composition is associative

```
Theorem composition_eq : forall (f1 f2 g1 g2 : arrow),
  (target f1 = source g1) ->
  (f1 == f2) -> (g1 == g2) -> (f1 @ g1) == (f2 @ g2).
```

```
Theorem composition_assoc : forall (f g h : arrow),
  (valid_arrow f) -> (valid_arrow g) -> (valid_arrow h) ->
  (target f = source g) -> (target g = source h) ->
  (f @ (g @ h)) == ((f @ g) @ h).
```

- All together, this took about 4000 lines of code.

Recap

- We're looking for the initial object in the category of Cartesian categories with NNOs
- I defined Goodstein's arithmetic: terms, functions, and equality
- Goodstein's system can be used to build a category, **PRA**, whose arrows are lists of function codes modulo equality

Recap

- We're looking for the initial object in the category of Cartesian categories with NNOs
- I defined Goodstein's arithmetic: terms, functions, and equality
- Goodstein's system can be used to build a category, **PRA**, whose arrows are lists of function codes modulo equality
- I claim PRA is the initial object we want!

PRA is a Cartesian category with NNO

- PRA has a terminal object: 1
- PRA has products:

$$N^n \times N^m = N^{n+m}$$

- The NNO of PRA is N.

PRA is a Cartesian category with NNO

- PRA has a terminal object: 1
- PRA has products:

$$N^n \times N^m = N^{n+m}$$

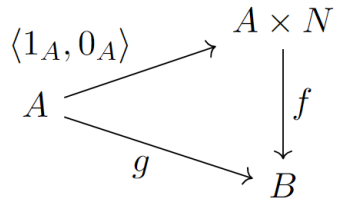
- The NNO of PRA is N.
  - This is nontrivial! Compare the induction of Goodstein's arithmetic to the induction of NNOs.

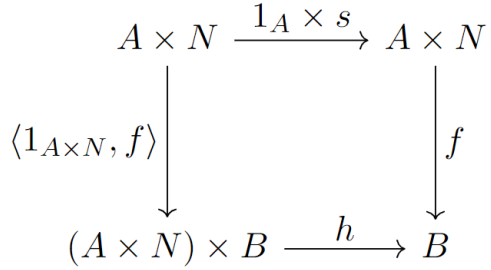
Goodstein:

$$f(x_1, \dots, x_k, 0) = B(x_1, \dots, x_k)$$
$$f\big(x_1, \dots, x_k, S(n)\big) = I\big(x_1, \dots, x_k, n, f(x_1, \dots, x_k, n)\big)$$

NNOs:

$$f(a, 0) = g(a)$$
$$f\big(a, s(n)\big) = h\big(a, n, f(a, n)\big)$$





PRA is a Cartesian category with NNO

- To get induction with arbitrary domain, we just need to establish an isomorphism $N \cong N^n$.
- Luckily, the usual isomorphism $N \cong N^2$ is definable in PRA!

So, what does it mean to be initial?

- If $\mathcal{C}$ is a Cartesian category with NNO, we want to show there is a unique functor $F : PRA \to \mathcal{C}$ which preserves products and NNO.

$$\mathbf{PRA} \overset{\exists!}{\longrightarrow} \mathcal{C}$$

So, what does it mean to be initial?

- If $\mathcal{C}$ is a Cartesian category with NNO, we want to show there is a unique functor $F : PRA \to \mathcal{C}$ which preserves products and NNO.
- This isn't true! What if there are two distinct NNOs in $\mathcal{C}$? Which do we map our objects to?

$$\mathbf{PRA} \xrightarrow{\exists?} \mathcal{C}$$

“Pseudo-initial”

- If $\mathcal{C}$ is a Cartesian category with NNO, then:
  - There exists a functor $F : PRA \to \mathcal{C}$ which preserves products and NNO
  - If $F, G : PRA \to \mathcal{C}$ are functors which preserve products and NNO, then there exists a unique natural transformation $\eta: F \Rightarrow G$.

$$\mathbf{PRA} \xrightarrow{\exists!\approx} \mathcal{C}$$

“Pseudo-initial”

- If $\mathcal{C}$ is a Cartesian category with NNO, then:
  - There exists a functor $F : PRA \to \mathcal{C}$ which preserves products and NNO
  - If $F, G : PRA \to \mathcal{C}$ are functors which preserve products and NNO, then there exists a unique natural transformation $\eta: F \Rightarrow G$.

- The category PRA is initial in the **2-category** of:
  - Objects: Cartesian categories with NNO
  - Arrows: functors which preserve products and NNO
  - Arrows between arrows: natural transformations

$$\mathbf{PRA} \xrightarrow{\exists!\approx} \mathcal{C}$$

Summary

- We defined a “Natural Numbers Object” in a category
- We considered the category of Cartesian categories with NNO
- We saw how Goodstein’s system of arithmetic is constructed
- We used Goodstein’s arithmetic to build a category PRA
- We saw that PRA is initial in the 2-category of Cartesian categories with NNO

Summary

- We defined a “Natural Numbers Object” in a category
- We considered the category of Cartesian categories with NNO
- We saw how Goodstein’s system of arithmetic is constructed
- We used Goodstein’s arithmetic to build a category PRA
- We saw that PRA is initial in the 2-category of Cartesian categories with NNO
- The next challenge: finding the initial object for categories with NNO and finite limits!

Thank you!

Special thanks to my supervisors, Professors Simon Henry and Philip Scott