A cartesian bicategory of polynomial functors in homotopy type theory

Samuel Mimram
June 18, 2021
This is joint work with Eric Finster, Maxime Lucas and Thomas Seiller.
Part I

Polynomials and polynomial functors
A polynomial is a sum of monomials

\[ P(X) = \sum_{0 \leq i < k} X^{n_i} \]

(no coefficients, but repetitions allowed)
A polynomial is a sum of monomials

\[ P(X) = \sum_{0 \leq i < k} X^{n_i} \]

(no coefficients, but repetitions allowed)

We can categorify this notion: replace natural numbers by elements of a set.

\[ P(X) = \sum_{b \in B} X^{E_b} \]
This data can be encoded as a **polynomial** $P$, which is a diagram in $\textbf{Set}$:

$$
E \xrightarrow{p} B
$$

where

- $b \in B$ is a monomial
- $E_b = P^{-1}(b)$ is the set of instances of $X$ in the monomial $b$. 

![Diagram of polynomial functor]({})
Polynomial functors

This data can be encoded as a polynomial $P$, which is a diagram in $\text{Set}$:

$$
\begin{array}{c}
E \\
\xrightarrow{p} \\
B
\end{array}
$$

where

- $b \in B$ is a monomial
- $E_b = P^{-1}(b)$ is the set of instances of $X$ in the monomial $b$.

It induces a polynomial functor

$$
[P] : \text{Set} \to \text{Set}
$$

$$
X \mapsto \sum_{b \in B} X^{E_b}
$$
Polynomial functors

For instance, consider the polynomial corresponding to the function

\[ E \xrightarrow{p} B \]

\[ \bullet \]

\[ \bullet \]

\[ \bullet \]

\[ \bullet \]

\[ \bullet \]

The associated polynomial functor is

\[ \llbracket P \rrbracket (X) : \text{Set} \to \text{Set} \]

\[ X \leftrightarrow X \times X \sqcup X \times X \times X \]
Polynomial functors

For instance, consider the polynomial corresponding to the function

\[
\begin{array}{c}
\mathbb{N} \\
\vdots \\
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\bullet
\end{array}
\xrightarrow{p} 1
\]

The associated polynomial functor is

\[
[P](X) : \textbf{Set} \to \textbf{Set}
\]

\[
X \mapsto X \times X \times X \times \ldots
\]
Polynomial functors

For instance, consider the polynomial corresponding to the function

\[
\mathbb{N} \xrightarrow{p} 1
\]

\[
\vdots
\]

\[
\bullet
\]

\[
\bullet
\]

\[
\bullet
\]

The associated polynomial functor is

\[
\lbrack P \rbrack(X) : \textbf{Set} \to \textbf{Set}
\]

\[
X \mapsto X \times X \times X \times \ldots
\]

A polynomial is \textbf{finitary} when each monomial is a finite product.
We will more generally consider a “typed variant” of polynomials \( P \)
\[
I \xleftarrow{s} E \xrightarrow{p} B \xrightarrow{t} J
\]
this means that

- each monomial \( b \) has a “type \( s(b) \in J \)”,
- each occurrence of a variable \( e \in E \) has a type \( s(e) \in I \).
Polynomial functors: typed variant

We will more generally consider a “typed variant” of polynomials $P$

$$I \xleftarrow{s} E \xrightarrow{p} B \xrightarrow{t} J$$

this means that

- each monomial $b$ has a “type $s(b) \in J$”,
- each occurrence of a variable $e \in E$ has a type $s(e) \in I$.

It induces a polynomial functor

$$[[P]](X) : \text{Set}^I \rightarrow \text{Set}^J$$

$$(X_i)_{i \in I} \mapsto \left( \sum_{b \in t^{-1}(j)} \prod_{e \in p^{-1}(b)} X_{s(e)} \right)_{j \in J}$$
The category of polynomial functors

Given a set $I$, we have an “identity” polynomial functor:

$$
\begin{array}{c}
I \\
\downarrow^\text{id} \\
I
\end{array} \quad \begin{array}{c}
I \\
\downarrow^\text{id} \\
I
\end{array} \quad \begin{array}{c}
I \\
\downarrow^\text{id} \\
I
\end{array}
$$
The category of polynomial functors

Given a set \( I \), we have an “identity” polynomial functor:

\[
\begin{array}{ccc}
I & \xleftarrow{id} & I \\
\end{array}
\]

\[
\begin{array}{ccc}
I & \xrightarrow{id} & I \\
\end{array}
\]

\[
\begin{array}{ccc}
I & \xrightarrow{id} & I \\
\end{array}
\]

**Proposition**

*The composite of two polynomial functors is again polynomial:*

\[
\text{Set}^I \xrightarrow{[P]} \text{Set}^J \xrightarrow{[Q]} \text{Set}^K
\]

\[
[Q] \circ [P]
\]
The category of polynomial functors

Given a set $I$, we have an “identity” polynomial functor:

$$I \xleftarrow{id} I \xrightarrow{id} I \xrightarrow{id} I$$

**Proposition**

The composite of two polynomial functors is again polynomial:

$$\text{Set}^I \xrightarrow{[P]} \text{Set}^J \xrightarrow{[Q]} \text{Set}^K$$

**Proof.**

Basically the usual one:

$$[Q] \circ [P](X_i) = \sum \prod \prod \prod X_i$$

$$\equiv \sum \sum \prod \prod X_i$$

$$\equiv \sum \prod X_i$$
The category of polynomial functors

We can thus build a category $\text{PolyFun}$ of sets and polynomial functors:

- an object is a set $I$,
- a morphism $F : I \to J$

is a polynomial functor

$$[[P]] : \text{Set}^I \to \text{Set}^J$$
A polynomial $P$

\[ I \xleftarrow{s} E \xrightarrow{p} B \xrightarrow{t} J \]

induces a polynomial functor

\[ [P] : \text{Set}^I \to \text{Set}^J \]

We have mentioned that composition is defined for polynomials. However, on polynomials, it is not strictly associative: we can build a bicategory $\text{Poly}$ of sets and polynomial functors.

This suggests that 2-cells are an important part of the story!
Morphisms between polynomials

A morphism between two polynomials is

\[
\begin{array}{c}
I \\ \\
\downarrow \varepsilon \\
E \\
\downarrow \\
B \\
\downarrow \beta \\
J
\end{array}
\quad \quad
\begin{array}{c}
I \\ \\
\downarrow \varepsilon' \\
E' \\
\downarrow \\
B' \\
\downarrow \\
J
\end{array}
\]

We send operations to operators, preserving typing and arities:

\[
\begin{array}{c}
\begin{array}{c}
\vdots \\
b
\end{array} \\
\downarrow \\
i
\end{array}
\quad \quad
\begin{array}{c}
\begin{array}{c}
\vdots \\
\beta(b)
\end{array} \\
i
\end{array}
\]
Morphisms between polynomials

A morphism between two polynomials is

\[
\begin{array}{ccc}
I & \xleftarrow{s} & E \\
\downarrow{\varepsilon} & \downarrow{\iota} & \downarrow{\beta} \\
I & \xleftarrow{s'} & E'
\end{array}
\quad \rightarrow \quad
\begin{array}{ccc}
B & \xrightarrow{t} & J \\
\downarrow{\beta} & & \\
B' & \xrightarrow{t'} & J
\end{array}
\]

We send operations to operators, preserving typing and arities:

\[
\begin{array}{cccc}
j_1 & j_2 & \cdots & j_n
\end{array}
\mapsto
\begin{array}{c}
\beta(b)
\end{array}
\]

We can build a bicategory \textbf{Poly} of sets, polynomials and morphisms of polynomials.
Morphisms between polynomial functors

A morphism between polynomial functors

$$[P], [Q] : \text{Set}^I \to \text{Set}^J$$

is a “suitable” natural transformation, and we can build a 2-category $\text{PolyFun}$. 
The category PolyFun is cartesian. Namely, given two polynomial functors in Poly

\[ P : I \to J \quad \text{and} \quad Q : I \to K \]

i.e., in Cat,

\[ [P] : \text{Set}^I \to \text{Set}^J \quad \text{and} \quad [Q] : \text{Set}^I \to \text{Set}^K \]

we have, in Cat,

\[ \langle P, Q \rangle : \text{Set}^I \to \text{Set}^J \times \text{Set}^K \cong \text{Set}^{J \sqcup K} \]

and the constructions preserve polynomiality: in PolyFun,

\[ \langle P, Q \rangle : I \to (J \sqcup K) \]
For the closed structure, we can hope for the same: given, in PolyFun,

\[ P : I \sqcup J \to K \]

i.e., in Cat,

\[ P : \text{Set}^{I \sqcup J} \to \text{Set}^K \]

we have

\[ \text{Set}^{I \sqcup J} \to \text{Set}^K \]

which suggests defining the closure as

\[ \llbracket J, K \rrbracket = \text{Set}^J \times K \]

for LL-people: this looks like \( \not\).

[i4]
For the closed structure, we can hope for the same: given, in PolyFun,

\[ P : I \sqcup J \rightarrow K \]

i.e., in Cat,

\[ P : \text{Set}^{I \sqcup J} \rightarrow \text{Set}^K \]

we have

\[
\frac{
\text{Set}^{I \sqcup J} \rightarrow \text{Set}^K \\
\text{Set}^I \times \text{Set}^J \rightarrow \text{Set}^K 
}{
\text{Set}^I \rightarrow (\text{Set}^K)^{\text{Set}^J}
}
\]
For the closed structure, we can hope for the same: given, in \textbf{PolyFun},

\[ P : I \sqcup J \to K \]

i.e., in \textbf{Cat},

\[ P : \text{Set}^{I \sqcup J} \to \text{Set}^K \]

we have

\[
\begin{align*}
\text{Set}^{I \sqcup J} & \to \text{Set}^K \\
\text{Set}^I \times \text{Set}^J & \to \text{Set}^K \\
\text{Set}^I & \to (\text{Set}^K)^{\text{Set}^J}
\end{align*}
\]
For the closed structure, we can hope for the same: given, in PolyFun,

\[ P : I \sqcup J \to K \]

i.e., in Cat,

\[ P : \text{Set}^{I \sqcup J} \to \text{Set}^K \]

we have

\[
\begin{align*}
\text{Set}^{I \sqcup J} & \to \text{Set}^K \\
\text{Set}^I \times \text{Set}^J & \to \text{Set}^K \\
\text{Set}^I & \to (\text{Set}^K)^{\text{Set}^J} \\
\text{Set}^I & \to \text{Set}^{\text{Set}^J \times K}
\end{align*}
\]
Closed structure

For the closed structure, we can hope for the same: given, in **PolyFun**, 

\[ P : I \sqcup J \to K \]

i.e., in **Cat**, 

\[ P : \text{Set}^{I \sqcup J} \to \text{Set}^K \]

we have

\[
\begin{align*}
\text{Set}^{I \sqcup J} & \to \text{Set}^K \\
\text{Set}^I \times \text{Set}^J & \to \text{Set}^K \\
\text{Set}^I & \to (\text{Set}^K)^{\text{Set}^J} \\
\text{Set}^I & \to \text{Set}^{\text{Set}^J \times K}
\end{align*}
\]

which suggests defining the closure as

\[ [J, K] = \text{Set}^J \times K \]
For the closed structure, we can hope for the same: given, in $\text{PolyFun}$,

$$P : I \sqcup J \to K$$

i.e., in $\text{Cat}$,

$$P : \text{Set}^{I \sqcup J} \to \text{Set}^K$$

we have

$$\text{Set}^{I \sqcup J} \to \text{Set}^K$$

which suggests defining the closure as

$$[J, K] = \text{Set}^J \times K$$

for LL-people: this looks like $!J \otimes K$. 
In terms of operations, the intuition behind the bijection

$$\text{PolyFun}(I \sqcup J, K) \cong \text{PolyFun}(I, \text{Set}^J \times K)$$

is that we can formally transform operations as follows
In terms of operations, the intuition behind the bijection

\[ \text{PolyFun}(I \sqcup J, K) \cong \text{PolyFun}(I, \text{Set}/J \times K) \]

is that we can formally transform operations as follows
There are two problems with our closure. The first one is that 

$$[I, J] = \text{Set}/I \times J$$

is too large to be an object of our category.
There are two problems with our closure. The first one is that

\[ [I, J] = \text{Set}/I \times J \]

is too large to be an object of our category.

One can restrict to polynomial functors which are \textit{finitary}: we can then take

\[ [I, J] = \text{Set}_{\text{fin}}/I \times J \]
Closed structure

There are two problems with our closure. The first one is that

$$[I, J] = \textbf{Set}/I \times J$$

is too large to be an object of our category.

One can restrict to polynomial functors which are \textbf{finitary}: we can then take

$$[I, J] = \textbf{Set}_{\text{fin}}/I \times J$$

or rather

$$[I, J] = \mathbb{N}/I \times J$$
There are two problems with our closure. The first one is that
\[ [I, J] = \text{Set}/I \times J \]
is too large to be an object of our category.

One can restrict to polynomial functors which are \textbf{finitary}: we can then take
\[ [I, J] = \text{Set}_{\text{fin}}/I \times J \]
or rather
\[ [I, J] = \mathbb{N}/I \times J \]

Finitary polynomial functors are also known as \textbf{normal functors} (introduced by Girard).


**Theorem**

*The category PolyFun is cartesian closed.*
Theorem
The category PolyFun is cartesian closed.

Remark (Girard)
The 2-category PolyFun is not cartesian closed.
Failure of the cartesian closed structure

We would like to have an equivalence of categories

\[ \text{PolyFun}(I \sqcup J, K) \simeq \text{PolyFun}(I, \mathbb{N}/J \times K) \]
Failure of the cartesian closed structure

We would like to have an equivalence of categories

$$\text{PolyFun}(I \sqcup J, K) \simeq \text{PolyFun}(I, \mathbb{N}/J \times K)$$

but consider the polynomial functor

$$\lbrack P \rbrack(X) = X^2$$
We would like to have an equivalence of categories

$$\text{PolyFun}(I \sqcup J, K) \simeq \text{PolyFun}(I, \mathbb{N}/J \times K)$$

but consider the polynomial functor

$$\lbrack P \rbrack(X) = X^2$$

which is induced by the polynomial

$$1 \leftarrow 2 \rightarrow 1 \rightarrow 1$$
Failure of the cartesian closed structure

We would like to have an equivalence of categories

\[ \text{PolyFun}(I \sqcup J, K) \simeq \text{PolyFun}(I, \mathbb{N}/J \times K) \]

but consider the polynomial functor

\[ \{P\}(X) = X^2 \]

which has two automorphisms

\[
\begin{array}{c}
1 \leftrightarrow 2 \longrightarrow 1 \longrightarrow 1 \\
\tau \downarrow \text{id} \downarrow \\
1 \leftrightarrow 2 \longrightarrow 1 \longrightarrow 1
\end{array}
\]

(two elements on the left, one on the right because 0 is initial)
Failure of the cartesian closed structure

We would like to have an equivalence of categories

$$\text{PolyFun}(I \sqcup J, K) \simeq \text{PolyFun}(I, \mathbb{N}/J \times K)$$

but consider the polynomial functor

$$\llbracket P \rrbracket(X) = X^2$$

which has two automorphisms

The equivalence fails:

$$\text{PolyFun}(0 \sqcup 1, 1) \not\simeq \text{PolyFun}(0, \mathbb{N}/1 \times 1)$$

(two elements on the left, one on the right because 0 is initial)
Fixing the cartesian closed structure

The failure of the equivalence

\[ \text{PolyFun}(0 \sqcup 1, 1) \not\cong \text{PolyFun}(0, \mathbb{N}/1 \times 1) \]

can be interpreted as being due to the fact that \(2 \in \mathbb{N}/1\) has no non-trivial isomorphism.

This suggests moving to **groupoids**!
Fixing the cartesian closed structure

The failure of the equivalence

\[ \text{PolyFun}(0 \sqcup 1, 1) \not\cong \text{PolyFun}(0, \mathbb{N}/1 \times 1) \]

can be interpreted as being due to the fact that \(2 \in \mathbb{N}/1\) has no non-trivial isomorphism.

This suggests moving to groupoids!

More precisely, we should replace \(\mathbb{N}\) by the groupoid \(\mathbb{B}\) of all symmetric groups.
The notion of polynomial functor generalizes in any locally cartesian closed category.
The notion of polynomial functor generalizes in any locally cartesian closed category.

...but the category $\text{Gpd}$ is not cartesian closed!
The notion of polynomial functor generalizes in any locally cartesian closed category.

...but the category $\textbf{Gpd}$ is not cartesian closed!

Kock has identified that if we perform all the usual constructions up to homotopy (slice, pullbacks, etc.), we recover a suitable setting to define polynomial functors.
Polynomial functors in groupoids

The notion of polynomial functor generalizes in any locally cartesian closed category.

...but the category $\text{Gpd}$ is not cartesian closed!

Kock has identified that if we perform all the usual constructions up to homotopy (slice, pullbacks, etc.), we recover a suitable setting to define polynomial functors.

This requires properly defining and using all the usual constructions in a suitable 2-categorical sense.
Polynomial functors in groupoids

Given a polynomial $P$

$$E \xrightarrow{p} B$$

the induced polynomial functor

$$\left[ P \right] : \text{Gpd} \to \text{Gpd}$$

$$X \mapsto \int_{b \in B} E_b$$

where $E_b$ is the homotopy fiber of $p$ at $b$ and

$$\int_{b \in E} E_b = \sum_{b \in \pi_0(B)} X_b / \text{Aut}(b)$$

where the quotient is to be taken homotopically...
Part II

Formalization in Agda
There is a framework in which everything is constructed up to homotopy for free: homotopy type theory.

Let’s formally develop the theory of polynomials in this setting.
Some notations

Notations:

- **Type**: the type of all types
Some notations

Notations:

- **Type**: the type of all types
- **$t \equiv u$**: equality between terms $t$ and $u$
Some notations

Notations:

- **Type**: the type of all types
- **$t \equiv u$**: equality between terms $t$ and $u$
- **$A \simeq B$**: equivalence between types $A$ and $B$
Some notations

Notations:

- Type: the type of all types
- $t \equiv u$: equality between terms $t$ and $u$
- $A \simeq B$: equivalence between types $A$ and $B$

Axiom:

- univalence: $(A \equiv B) \simeq (A \simeq B)$
Notations:

- **Type**: the type of all types
- **$t \equiv u$**: equality between terms $t$ and $u$
- **$A \simeq B$**: equivalence between types $A$ and $B$

Axiom:

- **univalence**: $(A \equiv B) \simeq (A \simeq B)$

Homotopy levels (type = space):
Some notations

Notations:

- **Type**: the type of all types
- **t ≡ u**: equality between terms t and u
- **A ≃ B**: equivalence between types A and B

Axiom:

- **univalence**: \((A ≡ B) ≃ (A ≃ B)\)

Homotopy levels (type = space):

- **propositions**: \(\text{is-prop } A = (x, y : A) \to x ≡ y\)
Some notations

Notations:

- Type: the type of all types
- $t \equiv u$: equality between terms $t$ and $u$
- $A \simeq B$: equivalence between types $A$ and $B$

Axiom:

- univalence: $(A \equiv B) \simeq (A \simeq B)$

Homotopy levels (type = space):

- propositions: $\text{is-prop } A = (x \ y : A) \to x \equiv y$
- sets: $\text{is-set } A = (x \ y : A) \to \text{is-prop } (x \equiv y)$
Some notations

Notations:

- **Type**: the type of all types
- **$t \equiv u$**: equality between terms $t$ and $u$
- **$A \simeq B$**: equivalence between types $A$ and $B$

Axiom:

- **univalence**: $(A \equiv B) \simeq (A \simeq B)$

Homotopy levels (**type** = **space**):

- **propositions**: $\text{is-prop } A = (x \ y : A) \to x \equiv y$
- **sets**: $\text{is-set } A = (x \ y : A) \to \text{is-prop } (x \equiv y)$
- **groupoids**: $\text{is-groupoid } A = (x \ y : A) \to \text{is-set } (x \equiv y)$
A polynomial is

\[ I \leftrightarrow^s E \rightarrow^p B \rightarrow^t J \]

We are tempted to formalize it as

```haskell
record Poly (I J : Type) : Type₁ where
  field
  B : Type
  E : Type
  t : B \rightarrow J
  p : E \rightarrow B
  s : E \rightarrow I
```

but this is not very good because operations on those involve many handling of equalities.
Formalizing polynomials

A polynomial is

\[ I \leftarrow^s E \rightarrow^p B \rightarrow^t J \]

We formalize it as a container:

record Poly (I J : Type) : Type₁ where
  field
    Op : J → Type
    Pm : (i : I) → {j : J} → Op j → Type
A polynomial is

\[ I \xleftarrow{s} E \xrightarrow{p} B \xrightarrow{t} J \]

We formalize it as a **container**:

```
record Poly (I J : Type) : Type₁ where
  field
    Op : J → Type
    Pm : (i : I) → {j : J} → Op j → Type
```

The identity is

\[
\text{Id} : \text{Poly } I I
\]

\[
\text{Op } \text{Id } i = \top
\]

\[
\text{Pm } \text{Id } i \{j = j\} \text{ tt } = i \equiv j
\]
A polynomial is

\[ I \leftarrow^s E \xrightarrow{p} B \xrightarrow{t} J \]

We formalize it as a \textbf{container}:

\begin{verbatim}
record Poly (I J : Type) : Type₁ where
  field
    Op : J → Type
    Pm : (i : I) → \{j : J\} → Op j → Type
\end{verbatim}

We sometimes write

\[ I \rightsquigarrow J = \text{Poly} \ I \ J \]
The polynomial functor induced by a polynomial $P$ is

\[ [\_] : I \rightsquigarrow J \to (I \to \text{Type}) \to (J \to \text{Type}) \]

\[ [\_] P \times j = \Sigma (\text{Op } P j) (\lambda c \to (i : I) \to (p : Pm P i c) \to (X i)) \]
The polynomial functor induced by a polynomial \( P \) is

\[
\lfloor \_ \rfloor : I \leadsto J \to (I \to \text{Type}) \to (J \to \text{Type})
\]

\[
\lfloor \_ \rfloor \; P \; X \; j = \Sigma \; (\text{Op} \; P \; j) \; (\lambda \; c \to (i : I) \to (p : \text{Pm} \; P \; i \; c) \to (X \; i))
\]

The composite of two functors is

\[
\_ \cdot \_ : I \leadsto J \to J \leadsto K \to I \leadsto K
\]

\[
\text{Op} \; (P \cdot Q) = \lfloor Q \rfloor \; (\text{Op} \; P)
\]

\[
\text{Pm} \; (\_ \cdot \_ \; P \; Q) \; i \; (c, a) = \Sigma \; J \; (\lambda \; j \to \Sigma \; (\text{Pm} \; Q \; j \; c) \; (\lambda \; p \to \text{Pm} \; P \; i \; (a \; j \; p)))
\]
Morphisms of polynomials

The type of morphisms between two polynomials is

```lean
record Poly→ (P Q : Poly I J) : Type where
  field
  Op→ : {j : J} → Op P j → Op Q j
  Pm≃ : {i : I} {j : J} {c : Op P j} → Pm P i c ≃ Pm Q i (Op→ c)
```
**Theorem**

*We can build a pre-bicategory of types, polynomials and their morphisms.*
Theorem
We can build a pre-bicategory of types, polynomials and their morphisms.

Theorem
We can build a bicategory of groupoids, polynomials in groupoids and their morphisms.
Theorem

This bicategory is cartesian.
Theorem

This bicategory is cartesian.

The product is $\sqcap$ on objects, left projection is

$$\text{proj}_l : (I \sqcap J) \rightsquigarrow I$$

$\text{Op} \ \text{proj}_l \ i = \top$

$\text{Pm} \ \text{proj}_l \ (\text{inl} \ i) \ {i'} \ \text{tt} = i \equiv i'$

$\text{Pm} \ \text{proj}_l \ (\text{inr} \ j) \ {i'} \ \text{tt} = \bot$

and pairing is

$$\text{pair} : (I \rightsquigarrow J) \to (I \rightsquigarrow K) \to I \rightsquigarrow (J \sqcap K)$$

$\text{Op} \ (\text{pair} \ P \ Q) \ (\text{inl} \ j) = \text{Op} \ P \ j$

$\text{Op} \ (\text{pair} \ P \ Q) \ (\text{inr} \ k) = \text{Op} \ Q \ k$

$\text{Pm} \ (\text{pair} \ P \ Q) \ i \ {\text{inl} \ j} \ c = \text{Pm} \ P \ i \ c$

...
In order to define the 1-categorical closure, the plan was:

\[
\text{Set} \xrightarrow{\sim} \text{Set}_{\text{fin}} \xrightarrow{\sim} \mathbb{N}
\]
In order to define the 1-categorical closure, the plan was:

\[ \text{Set} \rightsquigarrow \text{Set}_{\text{fin}} \rightsquigarrow \mathbb{N} \]

For the 2-categorical closure the plan is

\[ \text{Gpd} \rightsquigarrow \text{Gpd}_{\text{fin}} \rightsquigarrow \mathbb{B} \]

Here, \( \mathbb{B} \) is the groupoid with \( n \in \mathbb{N} \) as objects and \( \Sigma_n \) as automorphisms on \( n \).
We write $\text{Fin } n$ for the canonical finite type with $n$ elements: its constructors are 0 to $n-1$. 
We write \( \text{Fin } n \) for the canonical finite type with \( n \) elements: its constructors are 0 to \( n-1 \).

\[
\text{data Fin : } \mathbb{N} \to \text{Set where}
\]
\[
\text{zero : } \{n : \mathbb{N}\} \to \text{Fin (suc n)} \\
\text{suc : } \{n : \mathbb{N}\} (i : \text{Fin n}) \to \text{Fin (suc n)}
\]
Finite types

The predicate of being finite is

\[
is\text{-finite} : \text{Type} \to \text{Type} \\
is\text{-finite} \ A = \Sigma \mathbb{N} (\lambda n \to \parallel A \simeq \text{Fin} \ n \parallel)
\]
Finite types

The predicate of being finite is

$$\text{is-finite} : \text{Type} \to \text{Type}$$
$$\text{is-finite } A = \Sigma \mathbb{N} (\lambda n \to \parallel A \simeq \text{Fin } n \parallel)$$

The type of finite types is

$$\text{FinType} : \text{Type}_1$$
$$\text{FinType} = \Sigma \text{Type} \text{ is-finite}$$
The predicate of being finite is

\[ \text{is-finite} : \text{Type} \to \text{Type} \]
\[ \text{is-finite} \ A = \Sigma \ N \ (\lambda \ n \to \ |\ A \simeq \text{Fin} \ n |) \]

The type of finite types is

\[ \text{FinType} : \text{Type}_1 \]
\[ \text{FinType} = \Sigma \ \text{Type} \ \text{is-finite} \]

(note that this is a large type)
A polynomial is **finitary** when, for each operation, the total space of its parameters is finite:

\[
\text{is-finitary} : (P : I \to J) \to \text{Type}
\]

\[
\text{is-finitary } P = \{j : J\} \ (c : \text{Op } P \ j) \to \text{is-finite } (\Sigma I (\lambda i \to \text{Pm } P \ i \ c))
\]
A small model for finite types

The type of integers is

data \( \mathbb{N} \) : Type where
  zero : \( \mathbb{N} \)
  suc : \( \mathbb{N} \to \mathbb{N} \)
The type \( \mathcal{B} \) is

\[
\text{data } \mathcal{B} : \text{Type where }
\begin{align*}
\text{obj} & : \mathbb{N} \to \mathcal{B} \\
\text{hom} & : \{m, n : \mathbb{N}\} \ (\alpha : \text{Fin } m \cong \text{Fin } n) \to \text{obj } m \equiv \text{obj } n \\
\text{id-coh} & : (n : \mathbb{N}) \to \text{hom } \{n = n\} \cong \text{refl} \equiv \text{refl} \\
\text{comp-coh} & : \{m, n, o : \mathbb{N}\} \ (\alpha : \text{Fin } m \cong \text{Fin } n) \ (\beta : \text{Fin } n \cong \text{Fin } o) \to \\
& \quad \text{hom } (\cong \text{-trans } \alpha \ \beta) \equiv \text{hom } \alpha \cdot \text{hom } \beta
\end{align*}
\]

(this is a small higher inductive type!)
The type $B$ is

\[
\text{data } B : \text{Type where }
\begin{align*}
\text{obj} & : \mathbb{N} \to B \\
\text{hom} & : \{m, n : \mathbb{N}\} (\alpha : \text{Fin } m \simeq \text{Fin } n) \to \text{obj } m \equiv \text{obj } n \\
\text{id-coh} & : (n : \mathbb{N}) \to \text{hom } \{n = n\} \simeq \text{refl } \equiv \text{refl} \\
\text{comp-coh} & : \{m, n, o : \mathbb{N}\} (\alpha : \text{Fin } m \simeq \text{Fin } n) (\beta : \text{Fin } n \simeq \text{Fin } o) \to \\
& \quad \text{hom } (\simeq \text{-trans } \alpha \beta) \equiv \text{hom } \alpha \cdot \text{hom } \beta
\end{align*}
\]

(this is a small higher inductive type!)

**Theorem**
\[
\text{FinType } \simeq B.
\]
The closure

We define

\[ \text{Exp} : \text{Type} \to \text{Type}_1 \]
\[ \text{Exp } I = I \to \text{Type} \]

**Theorem**

*Ignoring size issues, for polynomials we have*

\[ (I \sqcup J) \Rightarrow K \simeq I \Rightarrow (\text{Exp } J \times K) \]
The closure

We define

\[ \text{Exp} : \text{Type} \rightarrow \text{Type}_1 \]
\[ \text{Exp } I = \Sigma (I \rightarrow \text{Type}) (\lambda F \rightarrow \text{is-finite} (\Sigma I F)) \]

**Theorem**

*Ignoring size issues, for finitary polynomials we have*

\[ (I \sqcup J) \sim K \simeq I \sim (\text{Exp } J \times K) \]
The closure

We define

\[ \text{Exp} : \text{Type} \to \text{Type}_1 \]

\[ \text{Exp } I = \Sigma \text{FinType} (\lambda N \to \text{fst } N \to I) \]

**Theorem**

*Ignoring size issues, for finitary polynomials we have*

\[ (I \sqcap J) \rightsquigarrow K \simeq I \rightsquigarrow (\text{Exp } J \times K) \]
We define

$\text{Exp : Type } \rightarrow \text{Type}$

$\text{Exp } I = \Sigma B (\lambda b \rightarrow B\text{-to-Fin } b \rightarrow A)$

**Theorem**

*For finitary polynomials we have*

$$(I \sqcup J) \leadsto K \simeq I \leadsto (\text{Exp } J \times K)$$
Note that

\[ \text{Exp} : \text{Type} \to \text{Type} \]
\[ \text{Exp} \; I = \Sigma \; B \; (\lambda \; b \to \; B\text{-to-Fin} \; b \to \; A) \]

is the free pseudo-commutative monoid!