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

Categorifying polynomials

A **polynomial** is a sum of monomials

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

(no coefficients, but repetitions allowed)

Categorifying polynomials

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.

$$P(X) = \sum_{b \in B} X^{E_b}$$

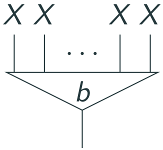
Polynomial functors

This data can be encoded as a **polynomial** P, which is a diagram in **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.



Polynomial functors

This data can be encoded as a **polynomial** P, which is a diagram in **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.

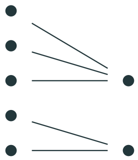
It induces a **polynomial functor**

$$\begin{aligned} [\![P]\!] : \mathbf{Set} &\to \mathbf{Set} \\ X &\mapsto \sum_{b \in B} X^{E_b} \end{aligned}$$

Polynomial functors

For instance, consider the polynomial corresponding to the function

$$E \xrightarrow{p} B$$



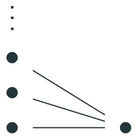
The associated polynomial functor is

$$\begin{aligned} [\![P]\!](X) &: \textbf{Set} \to \textbf{Set} \\ & X \mapsto X \times X \sqcup X \times X \times X \end{aligned}$$

Polynomial functors

For instance, consider the polynomial corresponding to the function

$$\mathbb{N} \xrightarrow{p} 1$$



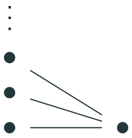
The associated polynomial functor is

$$\begin{aligned} [\![P]\!](X) &: \mathbf{Set} \to \mathbf{Set} \\ X &\mapsto X \times X \times X \times \dots \end{aligned}$$

Polynomial functors

For instance, consider the polynomial corresponding to the function

$$\mathbb{N} \xrightarrow{p} 1$$



The associated polynomial functor is

$$\begin{aligned} [\![P]\!](X) : \textbf{Set} &\to \textbf{Set} \\ X &\mapsto X \times X \times X \times \ldots \end{aligned}$$

A polynomial is **finitary** when each monomial is a finite product.

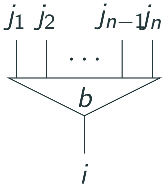
Polynomial functors: typed variant

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 $s(e) \in I$.



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**

$$\begin{aligned} [\![P]\!](X) : \mathbf{Set}^I &\to \mathbf{Set}^J \\ (X_i)_{i \in I} &\mapsto \left(\sum_{b \in t^{-1}(j)} \prod_{e \in p^{-1}(b)} X_{s(e)} \right)_{j \in J} \end{aligned}$$

The category of polynomial functors

Given a set I, we have an "identity" polynomial functor:

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

Given a set I, we have an "identity" polynomial functor:

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

Proposition
The composite of two polynomial functors is again polynomial:

$$\begin{tikzcd} \mathbf{Set}^I \arrow[r, "[\![P]\!]"] \arrow[rr, bend right, "[\![Q]\!]\circ[\![P]\!]"'] & \mathbf{Set}^J \arrow[r, "[\![Q]\!]"] & \mathbf{Set}^K \end{tikzcd}$$

The category of polynomial functors

Given a set I, we have an "identity" polynomial functor:

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

Proposition
The composite of two polynomial functors is again polynomial:

$$\begin{tikzcd} \mathbf{Set}^I \arrow[r, "[\![P]\!]"] \arrow[rr, bend right, "[\![Q]\!]\circ[\![P]\!]"'] & \mathbf{Set}^J \arrow[r, "[\![Q]\!]"] & \mathbf{Set}^K \end{tikzcd}$$

Proof.
Basically the usual one:

$$\begin{aligned} [\![Q]\!] \circ [\![P]\!](X_i) &= \sum \prod \sum \prod X_i \\ &\cong \sum \sum \prod \prod X_i \\ &\cong \sum \prod X_i \end{aligned}$$

The category of polynomial functors

We can thus build a category **PolyFun** of sets and polynomial functors:

- an object is a set I,
- a morphism

  $$F : I \to J$$

  is a polynomial functor

  $$[\![P]\!] : \mathbf{Set}^I \to \mathbf{Set}^J$$

Polynomial vs polynomial functors

A *polynomial* P

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

induces a *polynomial functor*

$$[\![P]\!] : \mathbf{Set}^I \to \mathbf{Set}^J$$

We have mentioned that composition is defined for polynomials. However, on polynomials, it is not strictly associative: we can build a *bicategory* **Poly** of sets an polynomial functors.

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

A morphism between two polynomials is

$$\begin{tikzcd}
I \arrow[d, equal] & E \arrow[l, "s"'] \arrow[r, "p"] \arrow[d, "\varepsilon"'] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[r, "t"] \arrow[d, "\beta"] & J \arrow[d, equal] \\
I & E' \arrow[l, "s'"] \arrow[r, "p'"'] & B' \arrow[r, "t'"'] & J
\end{tikzcd}$$

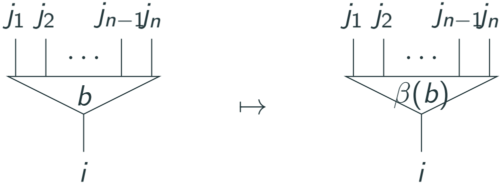
We send operations to operators, preserving typing and arities:

Morphisms between polynomials

A morphism between two polynomials is

$$
\begin{tikzcd}
I \arrow[d, equal] & E \arrow[l, "s"'] \arrow[r, "p"] \arrow[d, "\varepsilon"'] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[r, "t"] \arrow[d, "\beta"] & J \arrow[d, equal] \\
I & E' \arrow[l, "s'"] \arrow[r, "p'"'] & B' \arrow[r, "t'"'] & J
\end{tikzcd}
$$

We send operations to operators, preserving typing and arities:



We can build a bicategory **Poly** of sets, polynomials and morphisms of polynomials.

Morphisms between polynomial functors

A morphism between polynomial functors

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

is a "suitable" natural transformation, and we can build a 2-category **PolyFun**.

Cartesian structure

The category **PolyFun** is cartesian. Namely, given two polynomial functors in **Poly**

$$P : I \to J \qquad\qquad Q : I \to K$$

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

$$[\![P]\!] : \mathbf{Set}^I \to \mathbf{Set}^J \qquad\qquad [\![Q]\!] : \mathbf{Set}^I \to \mathbf{Set}^K$$

we have, in **Cat**,

$$\langle P, Q \rangle : \mathbf{Set}^I \to \mathbf{Set}^J \times \mathbf{Set}^K \cong \mathbf{Set}^{J \sqcup K}$$

and the constructions preserve polynomiality: in **PolyFun**,

$$\langle P, Q \rangle : I \to (J \sqcup K)$$

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**,

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

we have

$$\frac{\mathbf{Set}^{I \sqcup J} \to \mathbf{Set}^K}{}$$

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**,

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

we have

$$\dfrac{\dfrac{\mathbf{Set}^{I \sqcup J} \to \mathbf{Set}^K}{\mathbf{Set}^I \times \mathbf{Set}^J \to \mathbf{Set}^K}}{}$$

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**,

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

we have

$$\frac{\dfrac{\mathbf{Set}^{I \sqcup J} \to \mathbf{Set}^K}{\mathbf{Set}^I \times \mathbf{Set}^J \to \mathbf{Set}^K}}{\mathbf{Set}^I \to (\mathbf{Set}^K)^{\mathbf{Set}^J}}$$

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**,

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

we have

$$\frac{\dfrac{\dfrac{\mathbf{Set}^{I \sqcup J} \to \mathbf{Set}^K}{\mathbf{Set}^I \times \mathbf{Set}^J \to \mathbf{Set}^K}}{\mathbf{Set}^I \to (\mathbf{Set}^K)^{\mathbf{Set}^J}}}{\mathbf{Set}^I \to \mathbf{Set}^{\mathbf{Set}^J \times K}}$$

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**,

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

we have

$$\frac{\dfrac{\dfrac{\mathbf{Set}^{I \sqcup J} \to \mathbf{Set}^K}{\mathbf{Set}^I \times \mathbf{Set}^J \to \mathbf{Set}^K}}{\mathbf{Set}^I \to (\mathbf{Set}^K)^{\mathbf{Set}^J}}}{\mathbf{Set}^I \to \mathbf{Set}^{\mathbf{Set}^J \times K}}$$

which suggests defining the closure as

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

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**,

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

we have

$$\frac{\frac{\frac{\mathbf{Set}^{I \sqcup J} \to \mathbf{Set}^K}{\mathbf{Set}^I \times \mathbf{Set}^J \to \mathbf{Set}^K}}{\mathbf{Set}^I \to (\mathbf{Set}^K)^{\mathbf{Set}^J}}}{\mathbf{Set}^I \to \mathbf{Set}^{\mathbf{Set}^J \times K}}$$

which suggests defining the closure as

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

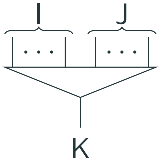
for LL-people: this looks like $!J \parr K$.

Closed structure

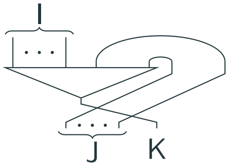
In terms of operations, the intuition behind the bijection

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

is that we can formally transform operations as follows



$\rightsquigarrow$

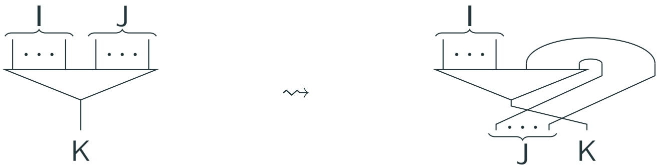


Closed structure

In terms of operations, the intuition behind the bijection

$$\mathbf{PolyFun}(I \sqcup J, K) \cong \mathbf{PolyFun}(I, \mathbf{Set}/J \times K)$$

is that we can formally transform operations as follows



Closed structure

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

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

is too large to be an object of our category.

Closed structure

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

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

is too large to be an object of our category.

One can restrict to polynomial functors which are **finitary**: we can then take

$$[I, J] = \mathbf{Set}_{\mathrm{fin}}/I \times J$$

Closed structure

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

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

is too large to be an object of our category.

One can restrict to polynomial functors which are **finitary**: we can then take

$$[I, J] = \mathbf{Set}_{\mathrm{fin}}/I \times J$$

or rather

$$[I, J] = \mathbb{N}/I \times J$$

Closed structure

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

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

is too large to be an object of our category.

One can restrict to polynomial functors which are **finitary**: we can then take

$$[I, J] = \mathbf{Set}_{\mathrm{fin}}/I \times J$$

or rather

$$[I, J] = \mathbb{N}/I \times J$$

Finitary polynomial functors are also known as **normal functors** (introduced by Girard).

Cartesian closed structure

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

Cartesian closed structure

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

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

Failure of the cartesian closed structure

We would like to have an equivalence of categories

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

but consider the polynomial functor

$$[\![P]\!](X) = X^2$$

Failure of the cartesian closed structure

We would like to have an equivalence of categories

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

but consider the polynomial functor

$$[\![P]\!](X) = X^2$$

which is induced by the polynomial

$$1 \longleftarrow 2 \longrightarrow 1 \longrightarrow 1$$

Failure of the cartesian closed structure

We would like to have an equivalence of categories

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

but consider the polynomial functor

$$[\![P]\!](X) = X^2$$

which has two automorphisms

$$\begin{tikzcd}
1 \arrow[d, equal] & 2 \arrow[l] \arrow[r] \arrow[d, "\tau"', "\mathrm{id}"] \arrow[dr, phantom, "\lrcorner", very near start] & 1 \arrow[r] \arrow[d] & 1 \arrow[d, equal] \\
1 & 2 \arrow[l] \arrow[r] & 1 \arrow[r] & 1
\end{tikzcd}$$

Failure of the cartesian closed structure

We would like to have an equivalence of categories

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

but consider the polynomial functor

$$[\![P]\!](X) = X^2$$

which has two automorphisms

$$\begin{tikzcd}
1 \arrow[d, equal] & 2 \arrow[l] \arrow[r] \arrow[d, shift right, "\tau"'] \arrow[d, shift left, "\mathrm{id}"] \arrow[dr, phantom, very near start, "\lrcorner"] & 1 \arrow[r] \arrow[d] & 1 \arrow[d, equal] \\
1 & 2 \arrow[l] \arrow[r] & 1 \arrow[r] & 1
\end{tikzcd}$$

The equivalence fails:

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

(two elements on the left, one on the right because 0 is initial)

Fixing the cartesian closed structure

The failure of the equivalence

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

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

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

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.

Polynomial functors in groupoids

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

Polynomial functors in groupoids

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

...but the category **Gpd** is not cartesian closed!

Polynomial functors in groupoids

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

...but the category **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 **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

$$\begin{aligned} [\![P]\!] : \mathbf{Gpd} &\to \mathbf{Gpd} \\ X &\mapsto \int^{b \in B} E_b \end{aligned}$$

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

$$\int^{b \in E} E_b = \sum_{b \in \pi_0(B)} X_b / \mathrm{Aut}(b)$$

where the quotient is to be taken homotopically...

Part II

Formalization in Agda

Homotopy type theory

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
- $\texttt{t} \equiv \texttt{u}$: equality between terms `t` and `u`
- $\texttt{A} \simeq \texttt{B}$: equivalence between types `A` and `B`

Some notations

Notations:

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

Axiom:

- univalence: $(\texttt{A} \equiv \texttt{B}) \simeq (\texttt{A} \simeq \texttt{B})$

Some notations

Notations:

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

Axiom:

- univalence: $(\texttt{A} \equiv \texttt{B}) \simeq (\texttt{A} \simeq \texttt{B})$

Homotopy levels (type = space):

Some notations

Notations:

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

Axiom:

- univalence: $\texttt{(A} \equiv \texttt{B)} \simeq \texttt{(A} \simeq \texttt{B)}$

Homotopy levels (type = space):

- propositions: $\texttt{is-prop A = (x y : A)} \to \texttt{x} \equiv \texttt{y}$

Some notations

Notations:

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

Axiom:

- univalence: $(\mathtt{A} \equiv \mathtt{B}) \simeq (\mathtt{A} \simeq \mathtt{B})$

Homotopy levels (type $=$ space):

- propositions: $\texttt{is-prop A = (x y : A)} \to \mathtt{x} \equiv \mathtt{y}$
- sets: $\texttt{is-set A = (x y : A)} \to \texttt{is-prop (x} \equiv \texttt{y)}$

Some notations

Notations:

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

Axiom:

- univalence: $\texttt{(A} \equiv \texttt{B)} \simeq \texttt{(A} \simeq \texttt{B)}$

Homotopy levels (type = space):

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

Formalizing polynomials

A polynomial is

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

We are tempted to formalize it as

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

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

Formalizing polynomials

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
```

Formalizing polynomials

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

```
Id : Poly I I
Op Id i = ⊤
Pm Id i {j = j} tt = i ≡ j
```

Formalizing polynomials

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
```

We sometimes write

```
I ⇝ J = Poly I J
```

Composing polynomials

The polynomial functor induced by a polynomial `P` is

```
⟦_⟧ : I ⇝ J → (I → Type) → (J → Type)
⟦_⟧ P X j = Σ (Op P j) (λ c → (i : I) → (p : Pm P i c) → (X i))
```

Composing polynomials

The polynomial functor induced by a polynomial `P` is

```
⟦_⟧ : I ⇝ J → (I → Type) → (J → Type)
⟦_⟧ P X j = Σ (Op P j) (λ c → (i : I) → (p : Pm P i c) → (X i))
```

The composite of two functors is

```
_·_ : I ⇝ J → J ⇝ K → I ⇝ K
Op (P · Q) = ⟦ Q ⟧ (Op P)
Pm (_·_ P Q) i (c , a) = Σ J (λ j → Σ (Pm Q j c) (λ p → Pm P i (a j p)))
```

Morphisms of polynomials

The type of morphisms between two polynomials is

```
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)
```

A bicategory

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

A bicategory

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.

Products

Theorem
This bicategory is cartesian.

Products

Theorem
This bicategory is cartesian.

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

```
projl : (I ⊔ J) ⇝ I
Op projl i = ⊤
Pm projl (inl i) {i'} tt = i ≡ i'
Pm projl (inr j) {i'} tt = ⊥
```

and pairing is

```
pair : (I ⇝ J) → (I ⇝ K) → I ⇝ (J ⊔ K)
Op (pair P Q) (inl j) = Op P j
Op (pair P Q) (inr k) = Op Q k
Pm (pair P Q) i {inl j} c = Pm P i c
```

Defining the exponential

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

$$\mathbf{Set} \qquad \rightsquigarrow \qquad \mathbf{Set}_{\mathrm{fin}} \qquad \rightsquigarrow \qquad \mathbb{N}$$

Defining the exponential

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

$$\mathbf{Set} \quad \rightsquigarrow \quad \mathbf{Set}_{\mathrm{fin}} \quad \rightsquigarrow \quad \mathbb{N}$$

For the 2-categorical closure the plan is

$$\mathbf{Gpd} \quad \rightsquigarrow \quad \mathbf{Gpd}_{\mathrm{fin}} \quad \rightsquigarrow \quad \mathbb{B}$$

Here, $\mathbb{B}$ is the groupoid with $n \in \mathbb{N}$ as objects and Σ_n as automorphisms on n.

Finite types

We write `Fin n` for the canonical finite type with `n` elements:
its constructors are `0` to `n-1`.

Finite types

We write `Fin n` for the canonical finite type with `n` elements:
its constructors are 0 to `n-1`.

```
data Fin : ℕ → Set where
  zero : {n : ℕ}              → Fin (suc n)
  suc  : {n : ℕ} (i : Fin n) → Fin (suc n)
```

Finite types

The predicate of being **finite** is

```
is-finite : Type → Type
is-finite A = Σ ℕ (λ n → ∥ A ≃ Fin n ∥)
```

Finite types

The predicate of being **finite** is

```
is-finite : Type → Type
is-finite A = Σ ℕ (λ n → ∥ A ≃ Fin n ∥)
```

The type of finite types is

```
FinType : Type₁
FinType = Σ Type is-finite
```

Finite types

The predicate of being **finite** is

```
is-finite : Type → Type
is-finite A = Σ ℕ (λ n → ∥ A ≃ Fin n ∥)
```

The type of finite types is

```
FinType : Type₁
FinType = Σ Type is-finite
```

(note that this is a *large* type)

Finitary polynomials

A polynomial is **finitary** when, for each operation, the total space of its parameters is finite:

```
is-finitary : (P : I ⇝ J) → Type
is-finitary P = {j : J} (c : Op P j) → is-finite (Σ I (λ i → Pm P i c))
```

A small model for finite types

The type of **integers** is

```
data ℕ : Type where
  zero : ℕ
  suc  : ℕ → ℕ
```

A small model for finite types

The type $\mathbb{B}$ is

```
data 𝔹 : Type where
  obj      : ℕ → 𝔹
  hom      : {m n : ℕ} (α : Fin m ≃ Fin n) → obj m ≡ obj n
  id-coh   : (n : ℕ) → hom {n = n} ≃-refl ≡ refl
  comp-coh : {m n o : ℕ} (α : Fin m ≃ Fin n) (β : Fin n ≃ Fin o) →
             hom (≃-trans α β) ≡ hom α · hom β
```

(this is a small higher inductive type!)

A small model for finite types

The type $\mathbb{B}$ is

```
data 𝔹 : Type where
  obj      : ℕ → 𝔹
  hom      : {m n : ℕ} (α : Fin m ≃ Fin n) → obj m ≡ obj n
  id-coh   : (n : ℕ) → hom {n = n} ≃-refl ≡ refl
  comp-coh : {m n o : ℕ} (α : Fin m ≃ Fin n) (β : Fin n ≃ Fin o) →
             hom (≃-trans α β) ≡ hom α · hom β
```

(this is a small higher inductive type!)

Theorem
`FinType` $\simeq \mathbb{B}$.

The closure

We define

```
Exp : Type → Type₁
Exp I = I → Type
```

Theorem
Ignoring size issues, for polynomials we have

```
(I ⊔ J) ⇝ K  ≃  I ⇝ (Exp J × K)
```

The closure

We define

```
Exp : Type → Type₁
Exp I = Σ (I → Type) (λ F → is-finite (Σ I F))
```

Theorem
Ignoring size issues, for finitary polynomials we have

```
(I ⊔ J) ⇝ K  ≃  I ⇝ (Exp J × K)
```

The closure

We define

```
Exp : Type → Type₁
Exp I = Σ FinType (λ N → fst N → I)
```

Theorem
Ignoring size issues, for finitary polynomials we have

$$(I \sqcup J) \rightsquigarrow K \quad \simeq \quad I \rightsquigarrow (\mathrm{Exp}\ J \times K)$$

The closure

We define

```
Exp : Type → Type
Exp I = Σ 𝔹 (λ b → 𝔹-to-Fin b → A)
```

Theorem
For finitary polynomials we have

```
(I ⊔ J) ⇝ K  ≃  I ⇝ (Exp J × K)
```

The exponential

Note that

```
Exp : Type → Type
Exp I = Σ 𝔹 (λ b → 𝔹-to-Fin b → A)
```

is the free pseudo-commutative monoid!