A cartesian bicategory of polynomial functors in homotopy type theory

Samuel Mimram (joint with Eric Finster, Maxime Lucas and Thomas Seiller)
MFPS XXXVII – August 31, 2021
In a nutshell

The situation:

- the category of polynomial functors is cartesian closed

Our contributions:

- we have formalized polynomials in groupoids (or spaces) in HoTT/Agda
- we have shown that the resulting bicategory is cartesian closed
- we have provided a small axiomatization of the type $B$ of natural numbers and bijections
In a nutshell

The situation:

- the category of polynomial functors is cartesian closed
- the 2-category of polynomial functors is not cartesian closed [Girard]

Our contributions:
In a nutshell

The situation:

- the category of polynomial functors is cartesian closed
- the 2-category of polynomial functors is not cartesian closed [Girard]
- we should consider polynomial functors in groupoids [Kock]

Our contributions:
In a nutshell

The situation:

- the category of polynomial functors is cartesian closed
- the 2-category of polynomial functors is not cartesian closed [Girard]
- we should consider polynomial functors in groupoids [Kock]

Our contributions:

- we have formalized polynomials in groupoids (or spaces) in HoTT/Agda
In a nutshell

The situation:

- the category of polynomial functors is cartesian closed
- the 2-category of polynomial functors is not cartesian closed [Girard]
- we should consider polynomial functors in groupoids [Kock]

Our contributions:

- we have formalized polynomials in groupoids (or spaces) in HoTT/Agda
- we have shown that the resulting bicategory is cartesian closed
In a nutshell

The situation:

- the category of polynomial functors is cartesian closed
- the 2-category of polynomial functors is not cartesian closed [Girard]
- we should consider polynomial functors in groupoids [Kock]

Our contributions:

- we have formalized polynomials in groupoids (or spaces) in HoTT/Agda
- we have shown that the resulting bicategory is cartesian closed
- we have provided a small axiomatization of the type $\mathbb{B}$ of natural numbers and bijections
Part I

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.

\[ P(X) = \sum_{b \in B} X^{E_b} \]
Polynomial functors

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

$$E \xrightarrow{p} B$$

where

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

\[ X \quad X \quad X \quad X \quad \cdots \quad X \quad \downarrow \quad b \]
Polynomial functors

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 $b$.

It induces a polynomial functor

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

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

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

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

For instance, consider the polynomial corresponding to the function

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

The associated polynomial functor is

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

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

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

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

A polynomial is \textbf{finitary} when each monomial is a finite product.
We will more generally consider a “colored variant” of polynomials $P$

$$I \leftarrow^s E \overset{p}{\longrightarrow} B \overset{t}{\longrightarrow} J$$

this means that

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

![Diagram](attachment:image.png)
Polynomial functors: typed variant

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

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

this means that

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

It induces a polynomial functor

$$[[P]](X) : \textbf{Set}^I \to \textbf{Set}^J$$

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

**Proposition**

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

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

\[[Q] \circ [P] = [Q \circ P]\]

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

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

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

induces a polynomial functor

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

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

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

\[ I \xleftarrow{s'} E' \xrightarrow{p'} B' \xrightarrow{t'} J \]

We send monomials to monomials, preserving typing and arities:

\[ i_1 \ i_2 \ i_{n-1}i_n \]

\[ b \]

\[ \mapsto \]

\[ i_1 \ i_2 \ i_{n-1}i_n \]

\[ \beta(b) \]

\[ j \]
Morphisms between polynomials

A morphism between two polynomials is

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

We send monomials to monomials, preserving typing and arities:

\[
\begin{array}{c}
i_1 \quad i_2 \quad i_{n-1}i_n \\
\downarrow{\vdots} \\
b \\
\downarrow{j}
\end{array} \quad \mapsto \quad 
\begin{array}{c}
i_1 \quad i_2 \quad i_{n-1}i_n \\
\downarrow{\vdots} \\
\beta(b) \\
\downarrow{j}
\end{array}
\]

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

A morphism between two polynomials is

\[
\begin{align*}
I & \xrightarrow{s} E \xrightarrow{p} B \xrightarrow{t} J \\
I & \xleftarrow{s'} E' \xrightarrow{p'} B' \xrightarrow{t'} J
\end{align*}
\]

We send monomials to monomials, preserving typing and arities:

\[
\begin{align*}
\vdots & \mapsto \beta(b)
\end{align*}
\]

We can build a bicategory \textbf{Poly} of sets, polynomials and morphisms of polynomials. In the following, we will restrict to the case where 2-cells are equivalences.
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}. 

Morphisms between polynomial functors
The category $\text{PolyFun}$ is cartesian. Namely, given two polynomial functors in $\text{Poly}$

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

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

$$\lbrack P \rbrack : \text{Set}^I \to \text{Set}^J \quad \lbrack Q \rbrack : \text{Set}^I \to \text{Set}^K$$

we have, in $\text{Cat}$,

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

and the constructions preserve polynomiality: in $\text{PolyFun}$,

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

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

we have

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

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

de \textit{PolyFun},

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

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

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

we have

\[
\begin{array}{c}
\text{Set}^{I \sqcup J} \to \text{Set}^K \\
\text{Set}^I \times \text{Set}^J \to \text{Set}^K \\
\end{array}
\]
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

\[
\begin{array}{c}
\text{Set}^{I \sqcup J} \to \text{Set}^K \\
\text{Set}^I \times \text{Set}^J \to \text{Set}^K \\
\text{Set}^I \to (\text{Set}^K)^{\text{Set}^J}
\end{array}
\]

for LL-people: this looks like \(!J`K\).
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

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

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

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 \textbf{PolyFun},

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

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

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

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 LL-people: this looks like \( !J \Rightarrow K \).
Closed structure

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

\[ \begin{array}{c}
I \\
\Downarrow \\
J \\
\Downarrow \\
K
\end{array} \quad \cong \quad \begin{array}{c}
I \\
\Downarrow \\
\text{Set}/J \\
\Downarrow \\
J \\
\Downarrow \\
K
\end{array} \]

via

\[ \text{Set}^I \cong \text{Set}/J \]
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.
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
\[[I, J] = \text{Set}_{\text{fin}}/I \times J\]
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

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

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 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 normal functors [Girard].
Theorem

The category \textbf{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

\[ [P](X) = X^2 : \text{Set}^{0 \sqcup 1} \to \text{Set}^1 \]
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

$$\llbracket P \rrbracket (X) = X^2 : \text{Set}^{0\sqcup 1} \to \text{Set}^1$$

which is induced by the polynomial

$$1 \leftarrow \ 2 \longrightarrow 1 \longrightarrow 1$$
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

\[ \llbracket P \rrbracket(X) = X^2 : \text{Set}^{0 \sqcup 1} \to \text{Set}^1 \]

which has two automorphisms

\[
\begin{array}{cccccc}
1 & \xleftarrow{\text{id}} & 2 & \longrightarrow & 1 & \longrightarrow & 1 \\
\downarrow & & \downarrow & & \downarrow & & \downarrow \\
1 & \xleftarrow{\tau} & 2 & \longrightarrow & 1 & \longrightarrow & 1
\end{array}
\]
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

\[ \llbracket P \rrbracket (X) = X^2 : \text{Set}^{0\sqcup 1} \to \text{Set}^1 \]

whose exponential transpose is

\[
\begin{array}{ccc}
0 & \xleftarrow{} & 0 \\
\downarrow & & \downarrow \\
1 & \xrightarrow{\star \mapsto 2} & \mathbb{N}
\end{array}
\]

and has only one automorphism.
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

\[
\lbrack P \rbrack(X) = X^2 : \text{Set}^{0\sqcup1} \to \text{Set}^{1}
\]

whose exponential transpose is

\[
0 \leftarrow 0 \rightarrow 1 \xrightarrow{\star \mapsto 2} \mathbb{N}
\]

and has only one automorphism.

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)
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**!
The failure of the equivalence

$$\text{PolyFun}(0 \sqcup 1, 1) \ncong \text{PolyFun}(0, \mathbb{N}/1 \times 1)$$

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.
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.

This requires properly defining and using all the usual constructions in a suitable 2-categorical sense.
Given a polynomial $P$

$$E \xrightarrow{p} B$$

the induced polynomial functor

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

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

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

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

where the quotient is to be taken 2-categorically / homotopically...
Part II

Formalization in Agda
There is a framework in which everything is constructed \textit{up to homotopy} for free: \textbf{homotopy type theory}.

In particular, there is a well-known notion of groupoid in this setting: a type with no non-trivial equalities between equalities.

Let’s formally develop the theory of polynomials in this setting.
A polynomial

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

is a **container**:

record Poly (I J : Type) : Type₁ where

  field

  \( \text{Op} : J \to \text{Type} \)

  \( \text{Pm} : (i : I) \to \{j : J\} \to \text{Op} j \to \text{Type} \)

We sometimes write

\[ I \rightsquigarrow J = \text{Poly } I \ J \]
Composing polynomials

The polynomial functor induced by a polynomial \( P \) is

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

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

The polynomial functor induced by a polynomial $P$ is

\[
[-] : I \leadsto J \to (I \to \text{Type}) \to (J \to \text{Type})
\]

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

The composite of two polynomials is

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

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

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

**Theorem**

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

Note: by univalence, we can use propositional equality for 2-cells, which simplifies the definition.
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.*
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 with $\sqcup$ as coproduct.
Defining the exponential

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

\[
\text{Set} \quad \rightsquigarrow \quad \text{Set}_{\text{fin}} \quad \rightsquigarrow \quad \mathbb{N}
\]
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 \( n \).
Finite types

We write $\text{Fin } n$ for the canonical finite type with $n$ elements: its constructors are 0 to $n-1$. 

\[
\text{data Fin : } \mathbb{N} \rightarrow \text{Set where}
\begin{align*}
\text{zero : } & \{n : \mathbb{N}\} \rightarrow \text{Fin (suc n)} \\
\text{suc : } & \{n : \mathbb{N}\} (i : \text{Fin n}) \rightarrow \text{Fin (suc n)}
\end{align*}
\]
We write $\text{Fin } n$ for the canonical finite type with $n$ elements: its constructors are $0$ to $n-1$.

data Fin : $\mathbb{N} \rightarrow \text{Set}$ where

- $\text{zero} : \{n : \mathbb{N}\} \rightarrow \text{Fin } (\text{suc } n)$
- $\text{suc} : \{n : \mathbb{N}\} \ (i : \text{Fin } n) \rightarrow \text{Fin } (\text{suc } n)$
The predicate of being finite is

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

The predicate of being finite is

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

The type of finite types is

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

The predicate of being finite is

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

The type of finite types is

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

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

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

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

The type of \textbf{integers} is

\begin{verbatim}
data \textit{N} : Type where
    zero : \textit{N}
    suc  : \textit{N} \to \textit{N}
\end{verbatim}
The type $\mathbb{B}$ is

```haskell
data $\mathbb{B}$ : Type where
  obj : $\mathbb{N} \to \mathbb{B}$
  hom : $\{n : \mathbb{N}\} \to \text{obj} n \equiv \text{obj} n$
  id-coh : $(n : \mathbb{N}) \to \text{hom} \{n = n\} \equiv \text{refl}$
  comp-coh : $\{m n o : \mathbb{N}\} \to \text{hom} \{\sim\text{-trans} \alpha \beta\} \equiv \text{hom} \alpha \cdot \text{hom} \beta$
```

(this is a small higher inductive type!)
A small model for finite types

The type $\mathbb{B}$ is

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

(this is a small higher inductive type!)

**Theorem**

$\text{FinType } \simeq \mathbb{B}$. 
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*

\[ (I \sqcup J) \rightsquigarrow K \simeq I \rightsquigarrow (\text{Exp} \ J \times K) \]
We define
\[
\operatorname{Exp} : \text{Type} \to \text{Type}_1
\]
\[
\operatorname{Exp} I = \Sigma (I \to \text{Type}) \left( \lambda F \to \text{is-finite} \left( \Sigma I F \right) \right)
\]

**Theorem**

*Ignoring size issues, for finitary polynomials we have*

\[
(I \sqcup J) \leadsto K \simeq I \leadsto (\operatorname{Exp} J \times K)
\]
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*

\[ (I \sqcap J) \leadsto K \cong I \leadsto (\text{Exp } J \times K) \]
We define
\[ \text{Exp} : \text{Type} \to \text{Type} \]
\[ \text{Exp } I = \sum \mathbb{B} (\lambda b \to \mathbb{B}\text{-to-Fin} \ b \to A) \]

**Theorem**
*For finitary polynomials we have*

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

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!
Questions?