Some Takeaways from Poly at Work 2024

Harrison Grodin

In this document, I describe some takeaways from the Poly at Work 2024 workshop. Developments were heavily impacted by discussions with other participants. The ideas presented here are joint work with Reed Mullanix.

1 Indexed Inductive Types via Endobicomodules

In programming languages, it is common to have inductive types. For example, the following type describes syntax trees:

```
type t =
  | Bool of bool
  | String of string
  | And of t * t
  | Concat of t * t
  | IsEmpty of t
  | If of t * t * t
```

This type can be understood as the initial algebra of the polynomial $p = 2 + S + y^2 + y + y^3$, where S is the set of finite-length strings. Via initiality, we can write a map out of this type to evaluate syntax trees. However, we are presented with a problem: what should we do with invalid states, like `Concat (Bool true, String "a")`? To remedy this, we may break `t` into two mutually-inductive types: one for boolean expressions and one for string expressions.

```
type t1 =
  | Bool of bool
  | And of t1 * t1
  | IsEmpty of t2

and t2 =
  | String of string
  | Concat of t2 * t2
  | If of t1 * t2 * t2
```

Now, an element of `t1` is a well-formed boolean expression, and an element of `t2` is a well-formed string expression. Invalid states are no longer representable: `Concat (Bool true, String "a")` is not an element of either `t1` or `t2`. We can express types `t1` and `t2` simultaneously as the

initial algebra of a polynomial in two variables:

$$p_1 = 2 + y_1^2 + y_2$$
$$p_2 = S + y_2^2 + y_1 y_2^2$$

Such a polynomial in two variables can be understood as the data of an endobicomodule

$$2y \xleftarrow{p} \triangleleft\ 2y,$$

using the polynomial p from the earlier type. This consists of two maps, λ and ρ, subject to some coherence conditions.

- A left-module $\lambda : p \to 2p$ consists of a map $p(1) \to 2$ that assigns to each position in p whether it belongs to p_1 or p_2.
- A right-module $\rho : p \to p \triangleleft 2y$ consists of a map $p[i] \to 2$ for each $i : p(1)$ that assigns to each direction whether it should go to y_1 or y_2.

To support N mutually-inductive types, one can use a bicomodule $Ny \xleftarrow{} \triangleleft\ Ny$. More generally, to support indexed inductive types with a parameter type A, one can use a bicomodule $Ay \xleftarrow{} \triangleleft\ Ay$.

This development gives a new perspective on "modes", splitting a type (like `t`) into multiple types (like `t1` and `t2`) where each has different characteristics.

Question 1. What is the type-theoretic interpretation of endobicomodules when the domain is not a linear Ay, but rather an arbitrary polynomial q?

2 Foundations of Poly

What is the essence of **Poly**?

2.1 A Zoo of Definitions

There are many equivalent definitions of **Poly**. One could say that **Poly** is:

1. the category of dependent lenses;
2. the coproduct completion of the product completion of the terminal category;
3. $\mathbf{Fam}(\mathbf{Set}^{\mathrm{op}})$;
4. the category of Grothendieck lenses;
5. the category of **Set**-indexed coproducts of representable functors;
6. a diagram in a locally cartesian closed category;
7. the category of connected-limit-preserving functors.

We observe that these definitions seem to be naturally grouped as follows:

I. Perspectives 1. and 2. describe **Poly** by freely adding limits and then colimits to a category; these look like $\mathbf{Fam}(\mathbf{Lim}(\mathcal{C})^{\mathrm{op}})$.

II. Perspectives 3. to 5. describe **Poly** by freely adding colimits to a category that already has limits; these look like $\mathbf{Fam}(\mathcal{C}^{\mathrm{op}})$.

III. Perspectives 6. and 7. describe **Poly** in terms of structure that already exists in a category.

What structure of **Poly** remains as we vary the parameters on each construction? Group I. is a special case of Group II., but preliminary discussions suggest that freely-added limits are not essential; we therefore focus on Group II.. Here, we always have coproducts, since they are freely added. Moreover, we get Day convolution structures for monoidal structures available in $\mathcal{C}$. Recall that the objects of $\mathbf{Fam}(\mathcal{C}^{\mathrm{op}})$ are pairs $(A : \mathbf{Set}, B : A \to \mathcal{C})$ of positions and directions, where the domain of B is the set A treated as a discrete category. For any monoidal structure $(\cdot, I)$ on $\mathcal{C}$, we get the monoidal structure

$$(A, B) \odot (A, B') = (A \times A', (a, a') \mapsto B(a) \cdot B'(a'))$$

on $\mathbf{Fam}(\mathcal{C}^{\mathrm{op}})$. We can also define the composition product $\triangleleft$ as in **Poly**, using external hom $\mathcal{C}(-,=)$ in place of the function space $\to$.

Question 2. Defined this way, what properties does $\triangleleft$ retain?

Perspective 6. generally behaves like **Poly** regardless of the base category and has been studied extensively in the literature. In his lightning talk, Kevin discussed that Perspective 7. can be altered to *finite*-connected-limit-preserving functors to produce a **Poly**-like category.

Question 3. What other properties on functors describe categories resembling **Poly**?

2.2 Poly as a Friendly Normal Duoidal Category

From another viewpoint, one could argue that **Poly** is fundamentally about the $\otimes$ and $\triangleleft$ monoidal structures. Most constructions in **Poly** are defined in terms of adjoints to these structures. Moreover, these monoidal structures $(y, \otimes)$ and $(y, \triangleleft)$ are normal duoidal. Preliminary discussions suggest that generalizations of **Poly** may arise through normal duoidal structures, but more work is needed to develop this idea further.