Geolog Lecture 3

Owen Lynch

December 18, 2025

1 Recap

Last week, we talked about:

1. Syntactic presentation of regular theories
2. Finite models of regular theories
3. Semantic account of regular theories

This time, we are going to further develop the semantic account of regular theories. We are then going to talk about "derived signatures." The idea of the derived signature is that it splits a theory into a signature, which gives the data model for the theory, and a bunch of axioms, which give the "well-formedness conditions" for the theory.

2 Regular theories

Rule 2.1 (Sigma theories)**.**

$$\frac{\Gamma \vdash A \text{ theory} \qquad \Gamma, x : A \vdash B \text{ theory}}{\Gamma \vdash \Sigma\, A\, B \text{ theory}}$$

Recall:

- $\Gamma : \mathsf{Reg} \to \mathsf{Cat}$
- $A : \int \Gamma \to \mathsf{Cat}$
- $B : \int (\Gamma.A) \to \mathsf{Cat}$

This is fairly straightforward:

$$(\Sigma\, A\, B)(\mathcal{E}, M) = (M_A : A(\mathcal{E}, M)) \times B(\mathcal{E}, (M, M_A))$$

As for introduction/elimination/beta/eta, this follows from the obvious isomorphism $\Gamma.A.B \cong \Gamma.(\Sigma\, A\, B)$.

This is representable because we can compose the display maps $S[\Gamma] \to S[\Gamma.A] \to S[\Gamma.A.B]$ to get $S[\Gamma] \to S[\Gamma.A.B] \cong S[\Gamma.(\Sigma\, A\, B)]$.

Rule 2.2 (Pi theories)**.**

$$\frac{\Gamma \vdash A : \mathsf{Theory} \qquad \Gamma, x : \mathsf{Elt}\, A \vdash B \text{ theory}}{\Gamma \vdash (x : A) \to B \text{ theory}}$$

Recall:

- $\Gamma : \mathsf{Reg} \to \mathsf{Cat}$
- $A : ((\mathcal{E}, M) : \int \Gamma) \to \mathsf{Type}(\mathcal{E}, M) = \mathcal{E}$
- $B : \int (\Gamma.\mathsf{Elt}\, A) \to \mathsf{Cat}$

Then we have

$$((x : A) \to B) : \int \Gamma \to \mathsf{Cat}$$

We define it in the following way. Suppose that $(\mathcal{E}, M) \in \int \Gamma$. Then let $A_{\mathcal{E}} = A(\mathcal{E}, M) \in \mathcal{E}$. There is a functor $\mathsf{wkn}_A : \mathcal{E} \to \mathcal{E}/A_{\mathcal{E}}$ defined by $X \mapsto (X \times A_{\mathcal{E}} \xrightarrow{\pi_2} A_{\mathcal{E}})$. Thus, we have $\mathsf{wkn}_A(M) \in \Gamma(\mathcal{E}/A_{\mathcal{E}})$.

Now, by functoriality, $A(\mathcal{E}/A_{\mathcal{E}}, \mathsf{wkn}_A(M)) = (A_{\mathcal{E}} \times A_{\mathcal{E}} \to A_{\mathcal{E}})$. This has a section given by $\Delta_{A_{\mathcal{E}}} : A_{\mathcal{E}} \to A_{\mathcal{E}} \times A_{\mathcal{E}}$; that is $\Delta_{A_{\mathcal{E}}} \in (\mathsf{Elt}\, A)(\mathcal{E}/A_{\mathcal{E}}, \mathsf{wkn}_A(M))$. Thus, we have $(\mathsf{wkn}_A(M), \Delta_{A_{\mathcal{E}}}) \in (\Gamma.\mathsf{Elt}\, A)(\mathcal{E}/A_{\mathcal{E}})$. This all makes the following definition well-typed.

$$((x : A) \to B)(\mathcal{E}, M) = B(\mathcal{E}/A_{\mathcal{E}}, (\mathsf{wkn}_A(M), \Delta_{A_{\mathcal{E}}}))$$

We now interpret the introduction and elimination rules. The introduction rule is:

$$\frac{\Gamma, x : \mathsf{Elt}\, A \vdash b : B}{\Gamma \vdash \lambda x.b : (x : A) \to B}$$

First, recall the type signature for b

$$b : \left((\mathcal{E}, M) : \int (\Gamma.\mathsf{Elt}\, A)\right) \to B(\mathcal{E}, M)$$

From this, we can directly define

$$(\lambda x.b)(\mathcal{E}, M) = b(\mathcal{E}/A_{\mathcal{E}}, (\mathsf{wkn}_A(M), \Delta_{A_{\mathcal{E}}})) \in B(\mathcal{E}/A_{\mathcal{E}}, (\mathsf{wkn}_A M, \Delta_{A_{\mathcal{E}}})) = ((x : A) \to B)(\mathcal{E}, M)$$

For the elimination rule

$$\frac{\Gamma \vdash a : \mathsf{Elt}\, A \qquad \Gamma \vdash f : (x : A) \to B}{\Gamma \vdash f\ a : B[x \setminus a]}$$

we recall the type signature of a

$$a : ((\mathcal{E}, M) : \int \Gamma) \to (\mathsf{Elt}\, A)(\mathcal{E}, M) = \mathcal{E}(1, A(\mathcal{E}, M))$$

This gives us a functor $a^*\colon \mathcal{E}/A_{\mathcal{E}} \to \mathcal{E}/1 \cong \mathcal{E}$ given by sending $X \to A$ to

$$\begin{array}{ccc} a^*(X) & \longrightarrow & X \\ \downarrow & & \downarrow \\ 1 & \xrightarrow{a} & A \end{array}$$

We apply this functor to $f\colon B(\mathcal{E}/A_{\mathcal{E}}, (\mathsf{wkn}_A(M), \Delta_{A_{\mathcal{E}}}))$ and we get $a^*(f)\colon B(\mathcal{E}, (M, a))$, as required. We leave verification of the beta/eta laws to the reader.

Now, we must also show that $(x\colon A) \to B$ is representable. This is actually not possible with our current definition of theory because it requires an induction argument on how the theory is built up; we will return to this when we have a more refined definition of theory.

Rule 2.3 (Propositional equality)**.**

$$\frac{\Gamma \vdash a\ a' \colon \mathsf{Elt}\, A}{\Gamma \vdash a = a' \colon \mathsf{Prop}}$$

For now, we only treat equality of elements of types, rather than equality of models of general theories. This is because semantically speaking, two models are objects in a category, and it doesn't make sense to compare two objects for equality.

The semantic interpretation of $(a = a')(\mathcal{E}, M)$ is the equalizer

$$a = a' \longrightarrow 1 \underset{a'}{\overset{a}{\rightrightarrows}} A$$

This is a subobject of 1, and hence an object of $\mathsf{Prop}(\mathcal{E}, M)$ as required.

Rule 2.4 (Propositional truncation)**.**

$$\frac{\Gamma \vdash A \colon \mathsf{Type}}{\Gamma \vdash [\![A]\!] : \mathsf{Prop}}$$

For a fixed $(\mathcal{E}, M) \in \int \Gamma$, the semantic interpretation of $[\![A]\!]$ is the regular-epi/mono factorization of the unique map $A \to 1$.

$$\begin{array}{ccc} A & \longrightarrow & 1 \\ & \searrow \quad \nearrow & \\ & [\![A]\!] & \end{array}$$

The introduction rule for $[\![A]\!]$ is easy:

$$\frac{\Gamma \vdash a \colon \mathsf{Elt}\, A}{\Gamma \vdash [\![a]\!] : \mathsf{Proof}\ [\![A]\!]}$$

If I have a section of $A \to 1$, then I also have a section of $[\![A]\!] \to 1$.

The elimination rule is more complicated

$$\frac{\Gamma \vdash p : \mathsf{Proof}\ [\![A]\!] \qquad \Gamma, x : A \vdash b : B \qquad \Gamma, x : A, x' : A \vdash q : \mathsf{Proof}(b[x] = b[x'])}{\Gamma \vdash (\mathsf{let}\ x \leftarrow p\ \mathsf{via}\ (x\ x'.q)\ \mathsf{in}\ b) : B}$$

This says that if from an element of A I can produce an element of B in such a way that the specific choice of element of A doesn't matter, then I can produce an element of B from $[\![A]\!]$.

This is a stronger elimination rule than the alternative

$$\frac{\Gamma \vdash a : \mathsf{Proof}\ [\![A]\!] \qquad \Gamma, x : A \vdash p : \mathsf{Proof}\ P}{\Gamma \vdash (\mathsf{let}\ x \leftarrow a\ \mathsf{in}\ p) : \mathsf{Proof}\ P}$$

This alternative would correspond to an epi-mono factorization; the stronger elimination rule takes advantage of the fact that the epimorphism is *regular*. Specifically, the regular elimination rule corresponds to the definition of $[\![A]\!]$ as a higher inductive type

$$\begin{aligned}&\mathsf{data}\ [\![A]\!] : \mathsf{Type}\ \mathsf{where}\\ &\quad \mathsf{squash} : A \to [\![A]\!]\\ &\quad \mathsf{squashed} : (x\ x' : A) \to \mathsf{squash}\ x = \mathsf{squash}\ x'\end{aligned}$$

This reflects the fact that regular categories are finite limit categories with a particular special class of colimits. We are going to see a lot of other examples of this general pattern.

Remark 2.5. Propositional truncation and sigma types allow us to define existential quantification via

$$\frac{\Gamma \vdash A : \mathsf{Type} \qquad \Gamma, x : \mathsf{Elt}\ A \vdash P : \mathsf{Prop}}{\Gamma \vdash (\exists (x : A).P) :\equiv [\![\Sigma\ A\ P]\!] : \mathsf{Prop}}$$

3 Regular theories with derived signatures

There's a problem with general regular theories: a finitely *presented* model of a regular theory is not necessarily a *finite* model.

To understand what this means, let us temporarily work back in finite limit theories (which have all of the rules from above except for propositional truncation).

If Γ is a context, and $\Gamma \vdash A : \mathsf{Type}$, then A presents a model of Γ in the following sense. The category $(\Gamma.A)(\mathsf{Set})$ has an initial object, and we may apply the forgetful functor $(\Gamma.A)(\mathsf{Set}) \to \Gamma(\mathsf{Set})$ to this initial object to get a model of Γ in Set.

This is because the category of models of a finite limit theory in Set has all colimits (this is classically known).

However, for most finite limit theories, a model presented by a type A will not restrict to a model in FinSet. For instance, this is the case in Monoid, which might be defined by

```
theory Monoid
  car : Type
  unit : car
  mul : car -> car -> car

  mul/runit : (x : car) -> mul x unit = x
  mul/lunit : (x : car) -> mul unit x = x
  mul/assoc : (x y z : car) -> (mul x (mul y z)) = (mul (mul x y) z)
end
```

The type `[x y : car, comm : mul x y = y x]` presents the monoid $\mathbb{N}^2$, which is clearly not finite.

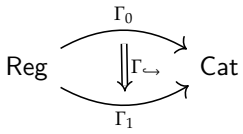
The geolog operation log stores a finitely presented model; what we want is for that model to be finite. This is so that, for instance, we can take advantage of the finite Herbrand universe for datalog queries; in general many operations are only possible when the model is actually finite.

The answer is that we give a semantics to our type theory which actually produces *two* related theories from a context. The first theory is the "actual" theory (the theory from the previous section), while the second theory is the "signature" theory. The idea is that just like before when we had a "signature" and "axioms" and a model of the signature might or might not satisfy the axioms; here the "signature" theory tells us what might or might not satisfy

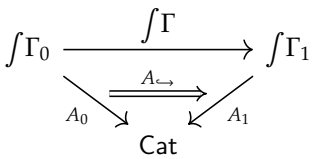
To be more specific, consider the 2-category $\mathsf{CAT}^{\hookrightarrow}$ where an object is a fully-faithful functor $\iota_C : C_0 \hookrightarrow C_1$.

Then we take the following semantics for our type theory.

- A context Γ is a 2-functor $\mathsf{Reg} \to \mathsf{Cat}^{\hookrightarrow}$. We sometimes refer to Γ in components as



- A substitution is a natural transformation.
- A type $\Gamma \vdash A$ type is interpreted as a diagram



- A term $\Gamma \vdash a : A$ is interpreted as

- an object $a_0(C, M) \in A_0(C, M)$ for all $(C, M) \in \int \Gamma_0$
- an object $a_1(C, N) \in A_1(C, N)$ for all $(C, N) \in \int \Gamma_1$
- a morphism $a_{\hookrightarrow} \colon A_{\hookrightarrow}(C, M)(a_0(C, M)) \to a_1(C, \Gamma(M))$ in $A_1(C, \Gamma(M))$ for all $(C, M) \in \int \Gamma_0$.

We must then carefully re-examine the semantics for each of the type theory rules. The rules for the actual component will remain the same; the problem will be to reinterpret the constructions for the data component.

Rule 3.1.

$$\vdash \mathsf{Type}\ \text{theory}$$

This is the same as before; $\mathsf{Type}_0(C) = \mathsf{Type}_1(C) = C$.

Rule 3.2.

$$\frac{\Gamma \vdash A \colon \mathsf{Type}}{\Gamma \vdash \mathsf{Elt}\, A\ \text{theory}}$$

We define $(\mathsf{Elt}\, A)_1(C, N) = \mathsf{Sub}(A(C, N))$, the poset of subobjects of $A(C, N)$ in C. The map $(\mathsf{Elt}\, A)_{\hookrightarrow} \colon C(1, A(C, M)) \to \mathsf{Sub}(A(C, M))$ is the inclusion: note that any morphism out of 1 is automatically a monomorphism.

Rule 3.2 is the main departure for the data component. One way of thinking about it is that it is a systematic way of replacing all function symbols with relation symbols. It is helpful to look at an example for where this comes into play

```
theory Magma
  car : Type
  unit : car
  mul : car -> car -> car
end

theory MagmaData
  car : Type
  is-unit : car -> Prop
  is-mul : car -> car -> car -> Prop
end

instance DataFromActual (M : Magma) : MagmaData
  car = M.car
  is-unit x = x == M.unit
  is-mul x y z = M.mul x y == z
end
```

4 Query IR