A sheaf-theoretic perspective on remote scry

`datnut-pollen`

August 2, 2022

This is something of a continuation of the last note I wrote on `https://github.com/urbit/plan/blob/master/oort/infra-offsite/categorical_databases.pdf` categorical databases and Urbit. I ended on a bit of a cliffhanger, which was a rough and incomplete description of a mathematical model of how to think about data on the Urbit network as a whole as a "sheaf of databases". I'd like to expand upon this, and hopefully put Ted's idea of Urbit as a "categorical dual" of a blockchain on more solid ground, as I think it's closely related and when two clever people independently arrive at a similar notion, there's either something worth digging deeper into or a shared delusion.

Reading the previous note on categorical databases is helpful, but shouldn't be required to understand this. Since most people at Tlon do not have an extensive background in mathematics, I want to explain this idea in an informal, conversational tone that doesn't presume much beyond basic set theory and the definition of a graph.

To start, I'll summarize roughly what the idea is, and then expand the description from there.

We are building a sort of geometrical description of data held on the Urbit network. To simplify matters for now, we will freeze the network at a moment in time. For each Urbit node we draw a vertex (which we will call a 0-cell). To each vertex, we associate some data held by the node. This could be the current state of Arvo, the entire event log, the state of a Gall app, known values attached to paths in scry namespace, etc.

Next, we draw an edge between each pair of vertices. Associated to this edge (called a 1-cell) is the subset of the data held by each vertex that is agreed upon by both nodes. This could be the last known IP address of some node, the chat message history in a channel that both nodes are members of, values that both nodes know in the scry namespace, the `type` of `$type`, etc. If two nodes do not agree on any data, then we do not draw the edge between them.

Now we continue this pattern into higher dimensions. To every 3-tuple of nodes, we draw a triangle (called a 2-cell), with data associated to that is agreed upon by all three nodes, or equivalently, data agreed upon by all 3 edges of the triangle that join them. The examples of what this might be are the same as for 1-cells and 0-cells, since we're basically taking a kind of set-theoretic intersection of the data associated the the 1-cells that form the border of this 2-cell. As with

the 1-cell case, if the 3 nodes do not agree on any data then we do not draw a 2-cell bounding them.

And so on, through k-cells for larger integer values of k. Say our network has $N+1$ nodes. Then the highest possible dimension (the value of k) of a cell is N. Associated to this N-cell is the set of all data agreed upon by all nodes in the network. The most direct example of this would be data about the PKI held on the Ethereum blockchain. Another possible example would be the IP addresses of galaxies, since they are determined by DNS.

What I have described so far is basically one way to build a "sheaf of databases". I will say what sheaf means shortly.

This also displays one notion of a duality between a blockchain and an Urbit node. The name for the geometric structure we have built is a *cellular complex* (in my last note, I called this a simplicial complex, which is a special case of a cellular complex, but we actually want this more general notion, per the following). It is possible to turn any finite cellular complex "upside down and inside out", by replacing all 0-cells with N-cells, 1-cells with $N-1$-cells, and so on, replacing k-cells with $N-k$-cells. This is actually "degenerate", since we do not require, for instance, that the two vertices joined by an edge are distinct (this is what makes it not a simplicial complex, which requires two distinct vertices for each edge, three distinct edges for each triangle, and so on). So in this case, there would be a single 0-cell, whose associated data is data held by a blockchain, with many 1-cells joining the 0-cell to itself, and so on.

You can also think of this geometric construction as a kind of hypergraph. From a planar graph, one can construct what is known as the dual graph by putting a vertex inside of each face of the original graph, and connecting two vertices if and only if the corresponding faces shared an edge in the original graph. The dual of a cellular complex, or of a hypergraph, is a kind of generalization of the notion of a dual graph to higher dimensions.

Disclaimer: to avoid introducing too much at once, some of this is more about *presheaves*, which are a slightly more general creature that can be made into a sheaf via a "completion" process.

1 Sheaves and the scry namespace

So what is a sheaf then? Sheaves were invented to give a kind of universal way to think about the following problem: given a rule by which we can associate *data* to *regions* in some space (for an *extremely* general notion of space, including manifolds, (hyper)graphs, algebraic varieties, schemes, categories, and much more), when is it possible to *glue* data together whenever two regions *overlap*? The combination of "rule to associate data to a region" and "rule by which to glue together data associated to two overlapping regions to get data associated to the union of those regions" is what defines a given *sheaf*.

My go-to example to illustrate this is to think of a network of radars distributed across some region. Each radar can detect movement within a circle of some radius with the radar at its center. If our network of radars has over-

lapping circles, we would like to be able to glue together the movement data detected by two radars on the overlap of their two circles.

Here, the circles are the *regions*, the *data* is the movement data detected by the radars, and the process for joining data detected by two radars on the *overlap* of their respective regions is the *gluing rule*.

For Urbit, the quintessential example I have in mind is this: radars are ships, the space in which they are situated in is the scry namespace, and the gluing rule is something like "take the intersection", so the data associated to a given k-cell is the data agreed upon by the nodes associated to the $k-1$-cells that bound the k-cell. This derives from the fact that the remote scry namespace is static and functional. Every endpoint either doesn't have a value yet, or has been set once and for all by the node at the root of the remote scry path. More on this shortly.

Let's look at another example, closer to the idea that a sheaf of databases is a way to describe shared knowledge among a group of people, that makes it more explicit why we can't just stop at graphs and need to consider a higher-dimensional structure.

Let's model a person's knowledge as a database (interconnected tables of examples of things and relationships between them). If you have several people at a table having a conversation where everyone can hear everyone else, you could imagine this as being represented by a graph with four vertices, with every pair of vertices connected by an edge (the complete graph on four vertices). Whenever someone says something, each person updates their personal database to say "Soandso said suchandsuch".

Contrast this scenario with four people playing a game of telephone. If anyone can whisper to anyone else, again our graph would be the complete graph on four vertices. But the way in which knowledge is shared is totally different - telephone has very different dynamics from a conversation. So we see that one-dimensional graph theory in the way in which we are using it cannot distinguish between a four person conversation and six two-person conversations. If we instead think of cellular complexes, though, the cellular complex for the four-person conversation will have all possible 2-cells you can form between the four vertices, as well as the one 3-cell, all together forming a filled-in tetrahedron. The cellular complex for the game of telephone, though, will only consist of the the 0-cells and 1-cells, with no 2-cells or 3-cells - in other words, just the edges of a tetrahedron.

Let's return to Urbit. Again, Urbit nodes are the radars or persons in conversation, and the landscape which the radars survey is the scry namespace, with endpoints given by the following

```
[ship=@p rift=@ud life=@ud vane=@tas care=@tas =case =spur]
```

For us, the salient detail is that its a ship followed by a path. Our ship `our` knows the value located at every scry path starting with `ship=our`, and only some subset of values at paths starting with some other `@p`s. When another ship `her` scries for data at some path that starts with `our`, it gets an identical copy of the noun `our` held at that path. Since Arvo has nice properties like referential transparency, immutable data, and determinism, we know that the

noun at that scry path will never change. Thus any given ship either doesn't know the noun held at that path, or knows the same noun at that path that every other ship that knows that noun has.

So let's try to make the outline of Urbit as a sheaf of databases more concrete by considering this particular case of the global scry namespace.

We keep the same cellular complex as outlined earlier. Given a ship a, we associate a 0-cell Δ^0_a consisting of the set of 2-tuples $\mathcal{S}(\Delta^0_a) = \{(\texttt{path}, \texttt{noun})\}$ consisting of a scry path known by a as well as the associated noun. Associated to each 1-cell $\Delta^1_{a,b}$ (an edge joining ships a and b) is the intersection of the $(\texttt{path}, \texttt{noun})$ tuples held by each ship, i.e. $\mathcal{S}(\Delta^1_{a,b}) = \mathcal{S}(\Delta^0_a) \cap \mathcal{S}(\Delta^0_b)$. To a 2-cell $\Delta^2_{a,b,c}$, we have

$$\begin{aligned}\mathcal{S}(\Delta^2_{a,b,c}) &= \mathcal{S}(\Delta^1_{a,b}) \cap \mathcal{S}(\Delta^1_{a,c}) \cap \mathcal{S}(\Delta^1_{b,c}) \\ &= \mathcal{S}(\Delta^0_a) \cap \mathcal{S}(\Delta^0_b) \cap \mathcal{S}(\Delta^0_c)\end{aligned}$$

and so on. This is a very simple example of a sheaf, with an unsophisticated gluing rule: we only need to take the intersection of the data, due to determinism and immutability of data. You might object that, in analogy with the conversation vs game of telephone example, we should only be thinking of edges between nodes and no higher-dimensional cells, since Urbit nodes only ever talk directly to one another. The facts of referential transparency and immutability constitute a consistency constraint on what data may be held at each scry path on a given ship, and this constraint is analogous to the fact that everyone having a conversation at a table must hear the same thing, whereas with a game of telephone there are no such constraints. Of course, you could have Urbit nodes that lie and tell two nodes that different nouns are at one scry path. We're going to sweep this issue under the rug, but this behavior can actually be accounted for in the sheaf-theoretic context using something called cosheaves. We won't say any more on the matter this time.

More generally, going back to thinking of the entire network of Urbit nodes as a sheaf of databases, it ought to be possible to think of the "remote scry sheaf" as a subsheaf of the full "Urbit sheaf", and similarly for other kinds of shared data on the network.

2 What is it good for?

Our remote scry sheaf is pretty boring - every path either has a value or it doesn't, and if it does have a value, it will never change. We have a geometrical model that encodes facts like "the intersection of the remote scry data at ships a, b, and c equals the intersection of the remote scry data agreed upon by the pairs a, b, a, c, and b, c". But so far this doesn't seem to have bought us anything besides a tidy formalism with some weird geometric structure hanging out in the background.

Sheaves were invented to study what circumstances make it so that you *can't* glue together data. Specifically, they were first invented to address the fact that often you can solve a differential equation on some subset, and find another solution on a subset that overlaps the first, and these two solutions don't agree on the overlap. In other words, while it may be possible to solve the equation *locally*, there may not be a *global* solution that works for the entire space on which the equation is defined. Sheaves were designed to address this *local-to-global* problem, and isolate precisely what prevents gluing together local data to get global data (generally called *obstructions*).

In our Urbit example, you can always glue - there are no obstructions. But we can make things more interesting by considering virtual namespaces where disagreement is allowed.

This virtual namespace, where nodes can disagree on the value at a given path, could be encoded by putting pointers at some path in the real namespace that point to other paths that store the values at which some given node(s) say is the value at some point in a virtual namespace.

A simple example where disagreement might arise is with real-world measurements. One node says the mountain is 10,000 feet tall. Another node says the mountain is 3048 meters tall. Then when we try to glue together this data, we see a disagreement, but the gluing rule could tell us to simply convert between metric and imperial units.

This allows one to encode multiple "perspectives" on the same data, recognizing data as always coming from a particular perspective, rather than asserting one and only one "true" value for some data. This is much closer to how real life works - every English speaker has their own version of English in their head, and when you communicate with someone else you attempt to glue your version to theirs, and your effective communication is restricted to the intersection of your English and their English, subject to whatever gluing rules you are able to discern to account for disagreements on, e.g., the meaning of a word.

Another reason to think in these terms is because, historically, the introduction of sheaves to mathematics was revolutionary. While they originated in the study of differential equations, they were very soon after adopted by Grothendieck to study algebraic geometry (a topic which starts by considering solution sets of families of polynomials as geometric objects and progresses into more and more general notions that encompasses more and more of mathematics). You might not expect it a priori, but algebraic geometry is some of the deepest waters in mathematics, with tendrils that touch virtually every other area in profound ways. The most important engine for the growth of these tendrils was the introduction of sheaf theory.

Thus, given the historical importance of sheaf theory in the history of mathematics, I tend to think there is a good chance that there is something really interesting to be found by thinking about data on Urbit in terms of sheaf theory, and it could lead to many innovative ideas with remote scry and beyond. I couldn't begin to say what yet, though, so for now its just another perspective from which to think about things.

3 Some references

High level blog post aimed at a general audience: `https://johncarlosbaez.wordpress.com/2015/03/30/spivak-part-2/` A networked world - David Spivak

A couple of technical papers:

`https://math.mit.edu/~dspivak/informatics/SD.pdf` Simplicial databases - David Spivak

`https://arxiv.org/pdf/1603.01446.pdf` Sheaves are the canonical data structure for sensor integration - Michael Robinson