Cartesian Equipments as Databases

AtCat Seminar

Michael Lambert 18 October 2022 Question: What is "knowledge representation?"

Partial answer: OLOGs "ontology logs" [KS12]

OLOGs are schematic representations of concepts, their attributes, interconnections, and facts about them. Example: arginine



Identify:

- 1. objects = concepts
- 2. attributes = arrows
- 3. facts = commutative diagrams

 $\mathsf{OLOG} = \mathsf{finitely}\mathsf{-}\mathsf{presented} \ \mathsf{category}$

Certain type of database scheme (that is, a small category [Spi12])

Functors $\mathscr{C}\to Set$ thought of as "populating" a scheme \mathscr{C} with actual data in the form of tables

Example: let ${\mathscr C}$ denote the scheme



Example: data table on ${\mathscr C}$ given by

Employee				
ID	First	Last	Mgr	isln
101	David	Hilbert	103	q10
102	Bertrand	Russell	102	x02
103	Alan	Turing	103	q10

Each arrow out of Employee is a column; Employee names the table; spans end up being functional: David Hilbert has one manager and works in one department etc... But not every relationship is functional; think of "friend of" or "enemy of."

"Bicategorical or Relational OLOGs" [Pat17]

- a finitely-presented 'bicategory of relations' in the sense of [CW87]; in particular a relational OLOG is a cartesian bicategory;
- 2. arrows encode relations rather than functions;
- 3. more expressive than functional OLOGs: functional relationships encoded as "maps;"
- Drawback: cartesian bicategories not so intuitive; more equations; have to make choices when setting up an OLOG; equational reasoning hard

If only there were a way to combine or synthesize the theories of functional and relational OLOGs.

A **double category** $\mathbb D$ is a pseudo-category internal to categories.

- 1. $\mathbb{D}_0=\text{category}\ \text{of}\ \text{objects}\ \text{and}\ \text{arrows}$
- 2. $\mathbb{D}_1=\mbox{category}$ of proarrows and cells
- 3. cells look like



 ⊗ external composition associative up to coherent isomorphism; external identity functor y : D₀ → D₁ Many examples: sets and spans, sets and relations, categories and profunctors, rings and modules, metric spaces, posets, convex thingies, Markov categories

Any bicategory is a double category without ordinary arrows and cells with trivial external source and target.

Any 2-category is a double category without proarrows and only cells with trivial internal domain and codomain.

Likewise any double category has an underlying (2-)category (functional part) and an underlying bicategory (relational part).

Thus, just as a cartesian bicategory is the basis of a relational OLOG, we expect that a cartesian double category is a possible generalization incorporating the missing "functional aspects."

A double category \mathbb{D} is **cartesian** [Ale18] if $\mathbb{D} \to \mathbb{D} \times \mathbb{D}$ and $\mathbb{D} \to 1$ have right adjoints in the 2-category of double categories, pseudo double functors and (vertical) transformations.

A double category is an **equipment** if $\mathbb{D}_1 \to \mathbb{D}_0 \times \mathbb{D}_0$ is a (bi)fibration (equivalently if every ordinary arrow has a proarrow companion and proarrow conjoint).

Theorem ([Lam])

The bicategory underlying any cartesian equipment is a cartesian bicategory.

Define: a 'double category of relations' is a locally posetal cartesian equipment satisfying Carboni and Walter's Frobenius Axiom. A **double-categorical OLOG** is then a finitely-presented 'double category of relations'.

That \mathbb{D} is an equipment means that each ordinary arrow f has a companion and conjoint: proarrows f_i and f^* with special cells



satisfying some equations. In relations, companions and conjoints are graphs and opgraphs.

From companion and conjoints all "restriction" and "extension" cells can be constructed as external composites:



Restrictions are cartesian and extensions are opcartesian with respect to the source-target projection $\mathbb{D}_1 \to \mathbb{D}_0 \times \mathbb{D}_0$.

In the case of relations, restrictions are computed by pullback and extensions are images:



In general, restrictions and extensions allow us to formulate complex propositions in double-categorical OLOGs.

Example time: Stargate SG1



Figure 1: SG1

Summary: MacGyver leads USAF team through a wormhole gate to various (Canadian) planets in search of tech and alies to defend against the threat of the Goa'uld (basically ancient aliens).

Posit two individuals



constants of type "system lord" (rulers of the Goa'uld) and type "jaffa" (warrior-slaves of the system lords).



Figure 2: Apophis



Figure 3: Teal'c

Functional OLOG style of a certain fact: "Teal'c is (or rather was) the first prime of Apophis"



Just ask that the diagram commutes.

But perhaps we'd like to view "first prime" as a relation.

Start with a niche:



complete using a restriction:



Then impose the requirement that the top proarrow is the terminal object in $\mathbb{D}(1,1)$ (or just admits an arrow from the terminal).

Restrictions and extensions construct propositions from types and other propositions.

Tabulators construct types from propositions. Say we have a proposition

symbiote
$$\xrightarrow{\text{is goa'uld}} 1$$

expressing what is means for a symbiote to be a Goa'uld.



Figure 4: A larval symbiote

By taking the tabulator, we have a type to work with



Further conditions on tabulators mean we can go back and forth between propositions and types as we please.

So, how do we get from OLOGs to data manipulation? Innovation: model theory

Recall: a structure-preserving functor on a small category $M: \mathscr{C} \to \mathscr{D}$ is a model of \mathscr{C} , viewed as a theory (finite products, finite limits, a topos etc) in suitably structured \mathscr{D} (something like **Set** or some other topos)

A data instance is a cartesian pseudo double functor $\mathbb{D} \to \mathbb{R}el$, that is, a "model" of \mathbb{D} in relations. (Note: any pseudo double functor preserves the equipment structure, hence restrictions and extensions.)

Upshot: this reduces examples of data manipulation to functorial semantics and computations of pullbacks and images in \mathbb{R} **el**.

In [Spi12] get data manipulation (and "data migration generally") via adjoint functors. Given database scehmes \mathscr{C} and \mathscr{D} , together with a translation $F: \mathscr{C} \to \mathscr{D}$, there are adjoint functors, namely,

$$[\mathscr{D}, \mathbf{Set}] \xrightarrow[\Sigma_F]{} [\mathscr{C}, \mathbf{Set}] \qquad \Sigma_F \dashv F^* \dashv \Pi_F$$

Substitution behaves like a select query; the right adjoint provides an inner join; the left adjoint is a skolemized outer join.

Using double categories we can do these operations without translations of schemes, hence without adjoint functors, by simply setting up the OLOGs carefully.

Select queries, filtering, joins end up being computations of images and pullbacks in $\mathbb{R}el$.

Simple OLOG describing a "mission concept"

$$\fbox{date} \times \fbox{location} \xrightarrow{\texttt{mission}} \fbox{purpose} \times \fbox{team}$$

instanced by a table

mission			
date	location	purpose	team
2-6-98	P41-771	search & rescue	SG3
7-31-98	Cimmeria	assist Cimmerians	SG1
1-2-99	P3R-272	investigate inscriptions	SG1
10-22-99	Ne'tu	search & rescue	SG1
8-6-2004	Tegalus	negotiation	SG9

Perhaps we only care about when our teams were off-world but not about either the date or purpose. Extend along projections:



The instance $M \colon \mathbb{D} \to \mathbb{R}\mathbf{el}$ preserves the extension.

Compute the image in relations.

date and team		
date	team	
2-6-98	SG3	
7-31-98	SG1	
1-2-99	SG1	
10-22-99	SG1	
8-6-2004	SG9	

This would be a "select" operation in SQL.

We might also filter by those missions of a certain team, say, SG1. Start by asking for an individual constant:

$$1 \xrightarrow{SG1} \text{team}$$

and form the restriction cell



In this case, the returned data would be a subtable consisting only of certain rows of the original **mission** table, that is,

SG1 missions			
date	location	purpose	
7-31-98	Cimmeria	assist Cimmerians	
1-2-99	P3R-272	investigate inscriptions	
10-22-99	Ne'tu	search & rescue	

that is, "filter" for rows with a certain value in SQL.

Indeed there are other examples of joins using a construction like local products.

We can also do (inner) joins of tables with a column in common. In general, this looks like:



Think of this as giving two tables, namely, p and q with entries from sets A, B, C and D. The tables have entries in one column in common, namely, those from the set B.

A simple OLOG:



Data:

expertise		
person	skill	
Hammond	command	
Kovacek	law	
Maybourne	chicanery	
Morrison	combat	
O'Neill	command	
Rothman	archaeology	
Simmons	sociopathy	
Warren	combat	

membership		
person	team	
Kovacek	SG9	
Morrison	SG3	
O'Neill	SG1	
Rothman	SG11	
Warren	SG3	

Form the restriction cell:



In relations this is just computed as a pullback of the product relation along the diagonal crossed with the identity. That is, form the cartesian product of all pairs from the two relations and match the person argument:

expertise 🖂 membership			
person	skill	team	
Kovacek	law	SG9	
Morrison	combat	SG3	
O'Neill	command	SG1	
Rothman	archaeology	SG11	
Warren	combat	SG3	

which is the join of the two along the common column.

So, it appears we can replicate SQL-type data operations without recourse to adjoint functors. This is "data munging."



Figure 5: Ologging Cimmerians

But why does all of this work? Because we can program it.

Julia Language (2009 Bezanson et. al. https://julialang.org/)

- 1. high-level, fast, used in scientific computing, numerical analysis, alternative to R and Python
- 2. macros like LISP
- 3. key feature: multiple dispatch (can overload a function symbol as long as you're clear what types arguments have)
- Algebraic Julia and CatLab (https://github.com/AlgebraicJulia/Catlab.jl): category theory enhancing capabilities of scientific computing
- machinery of GATs (generalized algebraic theories): categories, monoidal categories, bicategories, (now) double categories, cartesian double categories, and equipments

```
♣ category.jl > ...
@theory ThCategory{Ob,Hom} begin
  (dop begin
    (→) := Hom
     (.) := compose
  """ Object in a category """
  Ob::TYPE
   """ Morphism in a category """
  Hom(dom::Ob,codom::Ob)::TYPE
  compose(f::(A \rightarrow B), g::(B \rightarrow C))::(A \rightarrow C) \rightarrow (A::Ob, B::Ob, C::Ob)
  ((f \cdot g) \cdot h == f \cdot (g \cdot h)
    \dashv (A::0b, B::0b, C::0b, D::0b, f::(A \rightarrow B), g::(B \rightarrow C), h::(C \rightarrow D)))
   f \cdot id(B) == f - (A::Ob, B::Ob, f::(A \rightarrow B))
   id(A) \cdot f == f \dashv (A::Ob, B::Ob, f::(A \rightarrow B))
```

Here's the pattern:

- 1. load theory
- 2. define required syntax and any extra equations
- 3. generate instance that behaves according to the rules of the GAT
- 4. perform desired computations
- in particular, the GAT will build products, restrictions, extensions, tabulators, etc given basic types, arrows, proarrows, and cells.

Remaining questions:

- 1. outer joins?
- 2. program relation-valued double functors
- 3. query language?
- 4. translations of double-categorical schemes for full data migration; adjoints between virtual double categories of (lax) double functors?

For more you can see my blog post:

https://topos.site/blog/2022/09/data-operations-are-functorial-semantics/

THANK YOU!



Figure 6: Golf time

References

Evangelia Aleiferi.

Cartesian Double Categories with an Emphasis on Characterizing Spans. PhD thesis, Dalhousie University, 2018.

A. Carboni and R.C.F. Walters.

Cartesian bicategories I.

Journal of Pure and Applied Algebra, 49:11–32, 1987.

Robert E. Kent and David I. Spivak.

Ologs: A categorical framework for knowledge representation.

PLoS ONE, 7(1), 2012.

Michael Lambert.

'Double Categories of Relations'.

Theory and Applications of Categories (to appear).

🔋 Evan Patterson.

Knowledge representation in bicategories of relations. 2017.

https://arxiv.org/abs/1706.00526.



Functorial data migration.

Information and Computation, 217:31-51, 2012.