A Mathematical Formalism to Bridge Between Category Theory and Engineering

Henson Graves

January 21, 2025

Category theory is often thought to be 'abstract nonsense', but it can be very useful in engineering

Case Study Result From a Project called Algos

Goals

- A mathematical formalism that can be implemented within a proof assistant
- Used for formalizing theories in mathematics, science,engineering, and ...

Technical approach

- An axiomatic topos language (tierney1972sheaf) and sequent calculus deduction system (martin1975intuitionistic)
- Meta Algos is a category meta and is a specification for the proof assistant

Results

- language and deduction system correspond to methods of test and verification in physical environments
- Category constructions (limit cones, sheaves,...) to represent interacting physical systems in the Assistant

Project Trajectory

Stanford Computer Lab (smith1975computer)

- Worked on proof assistant for a formal set theory. Simple proofs were very difficult. Colleague suggested axiomatic topos theory instead of set theory

Algos Project at San Jose State U (graves1985algorith)

- Proof of Concept: Partial implementation of topos axioms
- Homemade deduction system worked
- Equal terms reduced to canonical form where possible

Aerospace Experience

- Enabled solving problem concerning topos axioms
- Enabled developing full mathematical formalism

Initial Implementation Partial

A first order language with **Map** *and* **Type** sorts with variables
Predicates

$$f = g, X = Y \tag{1}$$
$$f : A \to B \equiv Dom(f) = A \; and \; CODom(f) = B \tag{2}$$

Types (Objects)

$$\emptyset, One, X \times Y, X + Y, Y^X, Pow(X), \Omega \tag{3}$$

Some map constructions

- $true : One \to \Omega, \; false : One \to \Omega$
- $a : A = a : One \to A$
- $\rho 1 : X \to X + Y, \rho 2 : X \to X + Y$
- $f : X \to D, g : Y \to D, \Rightarrow f + g : X + Y \to D$
- $f : A \times B \to C \Rightarrow \lambda x.f : A \to C^B,$
- $g : A \to B^T \Rightarrow .g[t] : A \times T \to T$

Constructive Product Axioms

Problem is to find axioms that do not have existential quantifiers. Rule axioms were known for Products, Sums, Exponentials, For example, subtype classification

$$\textbf{tuple}$$

$$a : X \to A, b : X \to B \Rightarrow (a, b) : X \to A \times B \tag{4}$$

$$\textbf{projections}$$

$$f : C \to A, g : C \to B, \Rightarrow \pi 1(f, g) = f, \pi 2(f, g) = g \tag{5}$$

$$(t1, t2) = (r1, r2) \Rightarrow t1 = r1, t2 = r2 \tag{6}$$

$$(\pi 1_{X,Y}, \pi 2_{X,Y}) = id_{(X,Y)} \tag{7}$$

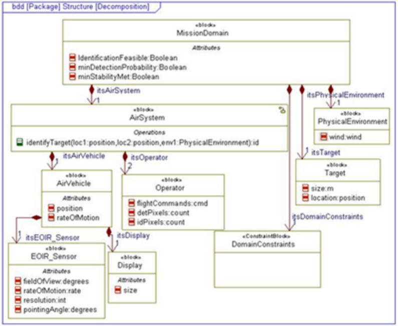
$$(f, g)(h) = (f(h), g(h)) \tag{8}$$

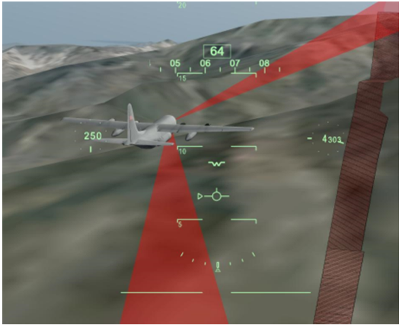
$$\pi : A \times B \to A \Rightarrow Epic(\pi) \tag{9}$$

Projection maps can be renamed.

$$(B \times C) = (\pi 1 : B + \pi 2 : C) = (x : B + y : C) \tag{10}$$

Constructed digital models and simulation and using them for analysis had a big impact on Algos



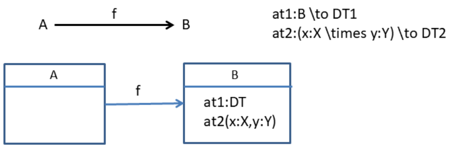


Impact on Algos

- Modeling Language constructions are similar to those of topos theory
- The graphically oriented language of SysML can be adapted for Algos in addition to linear syntax
- Experience enabled completing topos axioms
- Physical test and valuation uses constructive deduction as well as direct physical observation
- Engineering modeling language authoring systems are well developed, easy to use, and scale up
- Model development systems provide conception for their formalization

Adapted Graphically Oriented Syntax From Modeling Languages

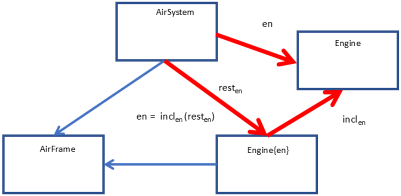
- maps = associations, types = blocks
- strongly overlapping constructions
- graphics can codify many additional assumptions



- an attribute such as $at1$ is a map $at1 : B \to DT$
- many other conventions make for ease of use

Completing Topos Axioms for Algos

First add inclusion subtypes with inclusion and restriction map and establish minimal factorization



Inclusion and Restriction Maps

$$f : A \to B \Rightarrow$$

$$incl_f : B\{f\} \to B, Monic(incl_f), rest_f : A \to B\{f\}, Epic(rest_f) \quad (11)$$

Takes more work to establish factorization is minimal

Empirical Test and Evaluation Uses Constructive Inference as Well as Direct Observation

Formalized deductions are labeled sequents $F : P(\hat{x}) \vdash Q(\hat{x})$. They form a category where the types are formulas and the maps are the labeled sequents. This is a Howard-Curry-Lambek category
The inference rules are presented in a numerator-denominator form where both numerator and denominator are deductions.

$$\frac{F : \Gamma \vdash P \quad G : \Gamma \vdash Q}{(F, G) : \Gamma \vdash P \wedge Q} \tag{12}$$

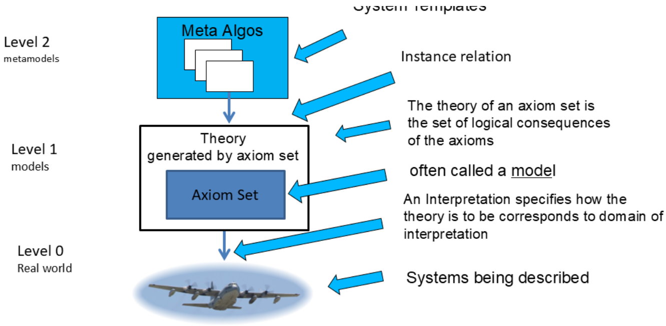
Deductions are equivalent to formulas, for example

$$\frac{F : \Gamma, P \vdash Q}{\lambda P.F : \Gamma \vdash P \Rightarrow Q}. \tag{13}$$

A theorem is a deduction of the form **One** $\vdash$ **F**

Engineering Authoring Assistants Suggest that an Algos Theory Could be the specification for an Implementable

Variations of this diagram are common in engineering and computer science.



- Level 0 is the interpretation domain for a theory
- Level 1 is a theory, i.e, the deductive closure of an axiom set with respect to the deduction system
- Level 3 is the meta theory with a meta types (template) for the theories

Meta Algos is a specification for a proof assistant for categories

- Meta Algos is an Algos theory with additional axioms used to represent axiom set theories, category theories, deductions, functors,...
- Since Meta Algos is a topos and is used to represent categories we distinguish

$$\mathbf{F} :: \mathbf{A} \to \mathbf{B} \ , \ \mathcal{D} :: \mathbf{A} \Rightarrow \mathbf{F}(\mathcal{D}) :: \mathbf{B} \tag{14}$$

- Types are templates, their instances are individual theories
- Meta Algos has additional axioms to make the types into 'sets'

$$\mathbf{\Omega} = \{\mathbf{True}, \mathbf{False}\} \tag{15}$$

$$\mathcal{A}, \mathcal{B} \in \mathbf{A}, \mathcal{B} \in \mathbf{A}\{\mathcal{B}\} \Rightarrow \mathcal{A} = \mathcal{B} \tag{16}$$

- Meta theory makes extensive use of membership, inclusion, and the Algos constructions, including recursion

Axiom Sets, Languages, theories are Recursive Types

Example meta types that have to be defined

- $\textbf{Signature} = \textbf{Const} + \textbf{Pred} + \textbf{Function}$
- $\textbf{Formula}(\textbf{Signature})$
- $\textbf{AxSet} \sqsubseteq \textbf{Formulas}(\textbf{Signature})$
- $\textbf{Deduction}(\textbf{Formulas}(\textbf{Signature}))$
- $(\textbf{F} : \textbf{One} \vdash \textbf{Q}) \in \textbf{Deduction} \Rightarrow \textbf{F} \in \textbf{Theorem}(\textbf{AxSet})$
- $\mathcal{C} = \{A, B, C, D, p1 : A \to B, p2 : A \to B, p3 : B \to C, p3(p2) : A \to C\}$ implies $\Rightarrow \mathcal{C} \in \textbf{Cat}$

Recursive Types in Meta Algos

List of a type in computer science is often written as:

$$\mathbf{List}(\mathbf{A}) = \mathbf{Nil} + \mathbf{Cons}(\mathbf{First}, \mathbf{Rest}) \tag{17}$$

with some axioms. We do the same by renaming of inclusions and projections maps in sums and products.

$$\mathbf{List}(\mathbf{A}) = \emptyset + (\mathbf{A} \times \mathbf{List}(\mathbf{A}) \tag{18}$$

$$\mathbf{List}(\mathbf{A}) = \mathbf{Nil} : \emptyset + \mathbf{Cons} : (\mathbf{A} \times \mathbf{List}(\mathbf{A})) \tag{19}$$

where nil and cons are the names of the inclusion maps.

$$\pi\mathbf{1} : \mathbf{A} \times \mathbf{List}(\mathbf{A}) = \mathbf{A} \ , \ \pi\mathbf{2} : \mathbf{A} \times \mathbf{List}(\mathbf{A}) = \mathbf{List}(\mathbf{A}) \tag{20}$$

imply

$$\mathbf{List}(\mathbf{A}) = \sigma\mathbf{1} : \emptyset + \sigma\mathbf{2} : (\pi\mathbf{1}, \pi\mathbf{2}) \tag{21}$$

or

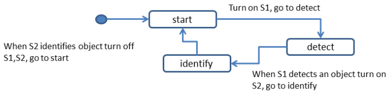
$$\mathbf{List}(\mathbf{A}) = \mathbf{Nil} + \mathbf{Cons}(\mathbf{First}, \mathbf{Rest}) \tag{22}$$

Behavior makes use of abstraction in Meta Algos

Behavior in engineering models is often represented by state machines. A state machine monitors sensors, when they change, depending on the current state, the action transitions to a new state and performs the action to modify the output attribute.

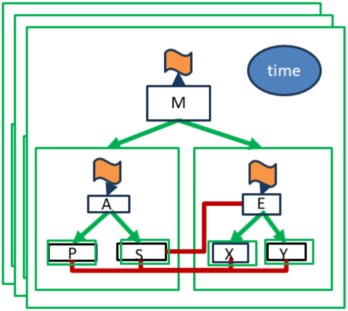
$$M = (In \times State \times Out) \tag{23}$$

and a type T finite linearly ordered time, and a state chart map $d : M \to M$ represents the diagram.



Behavior is represented as an exponential $\mathbf{M}^{\mathbf{Time}}$

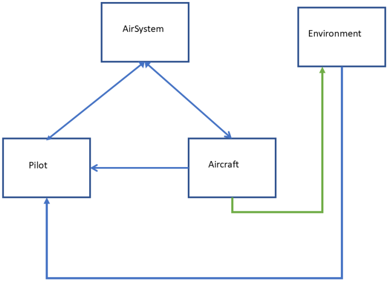
Interacting Systems within $\mathcal{A}$ in Meta Algos



- systems and their subsystems (green boxes)
- a system has a largest element, flags are attributes of largest element
- interaction between systems (red lines)

A System in Meta Algos

A system is a hierarchical structure of maps and types in $\mathcal{A}$ in Meta Algos



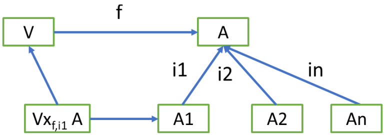
- the system S has a top element
- the system decomposition maps are bidirectional
- the components of S may have arrows to other maps in $\mathcal{A}$
- S is a bi-cone. This enables routing the interfaces of the system component through the greatest type. We write this as $S(A)$ where A is the greatest type of

A a collection of systems in $\mathcal{A}$ is a Covering Sieve one Shows, for example

The pullback of a map $f : V \to A_j$ along S(A) is

$$f^*(S) = \{V \times_{f,i} A_i \to V\} \in S(V) \tag{24}$$

for $A_i \in S(A)$ is a base change.



Showing that a covering by subsystem of $\mathcal{A}$ is a Grothendieck topology requires showing that $\mathcal{A}$ is a Grothendieck topos

Next Steps

- find an open source SysMl development platform
- implement Meta Algos as the proof assistant within the platform
- Build a business plan