Teaching Domain Experts to Model Categorically

Lessons and Opportunities

Donald Thompson

Founder | President | CTO

Maana Q: Multi-paradigm Modeling and Simulation Platform

Business AI
Digital Transformation Team

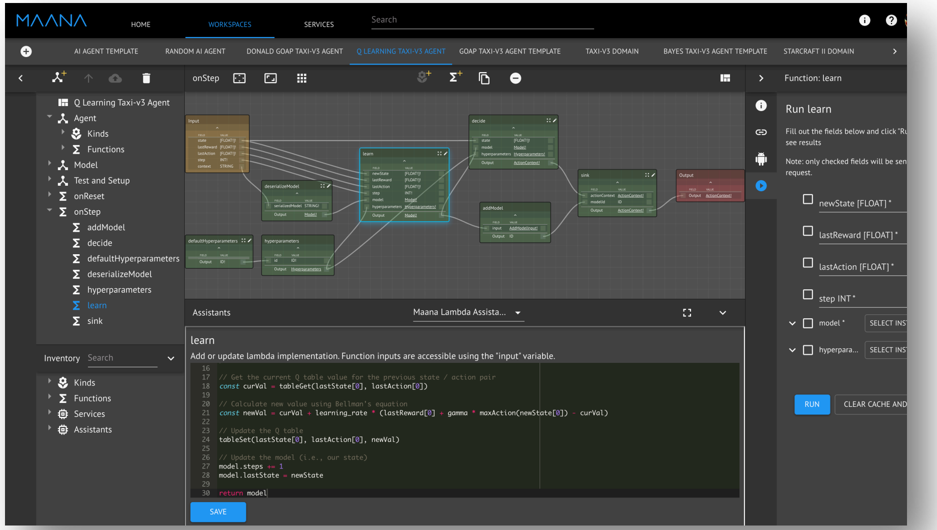
- Business Sponsor
- Subject Matter Expert
- Solution Architect
- Software Engineer
- Data Scientist
- Data Engineer

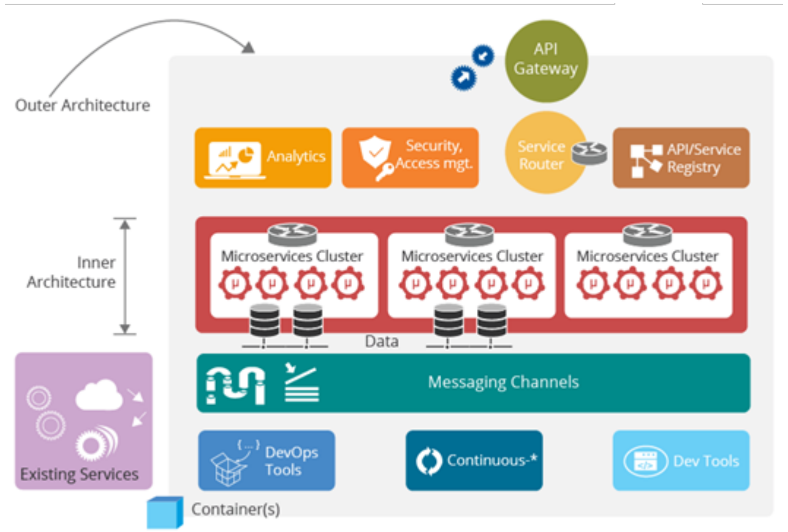
Knowledge Microservices
Query, Reasoning, Algorithms, and Data Access

- Trading Models
- Fleet Utilization
- Scheduling
- Safe Routing
- Maintenance Minimization
- Engineering Optimization
- Threat Detection
- Risk Mitigation
- Health, Safety, and Environment
- Supply Chain Demand Signal and Risk Minimization

User Experiences
Custom Knowledge Applications

- Web (e.g., React)
- Mobile (e.g., Flutter)
- Desktop (e.g., Electron)
- Mixed-Reality (e.g., Unity)
- PowerApp (e.g., Virtual Agent)

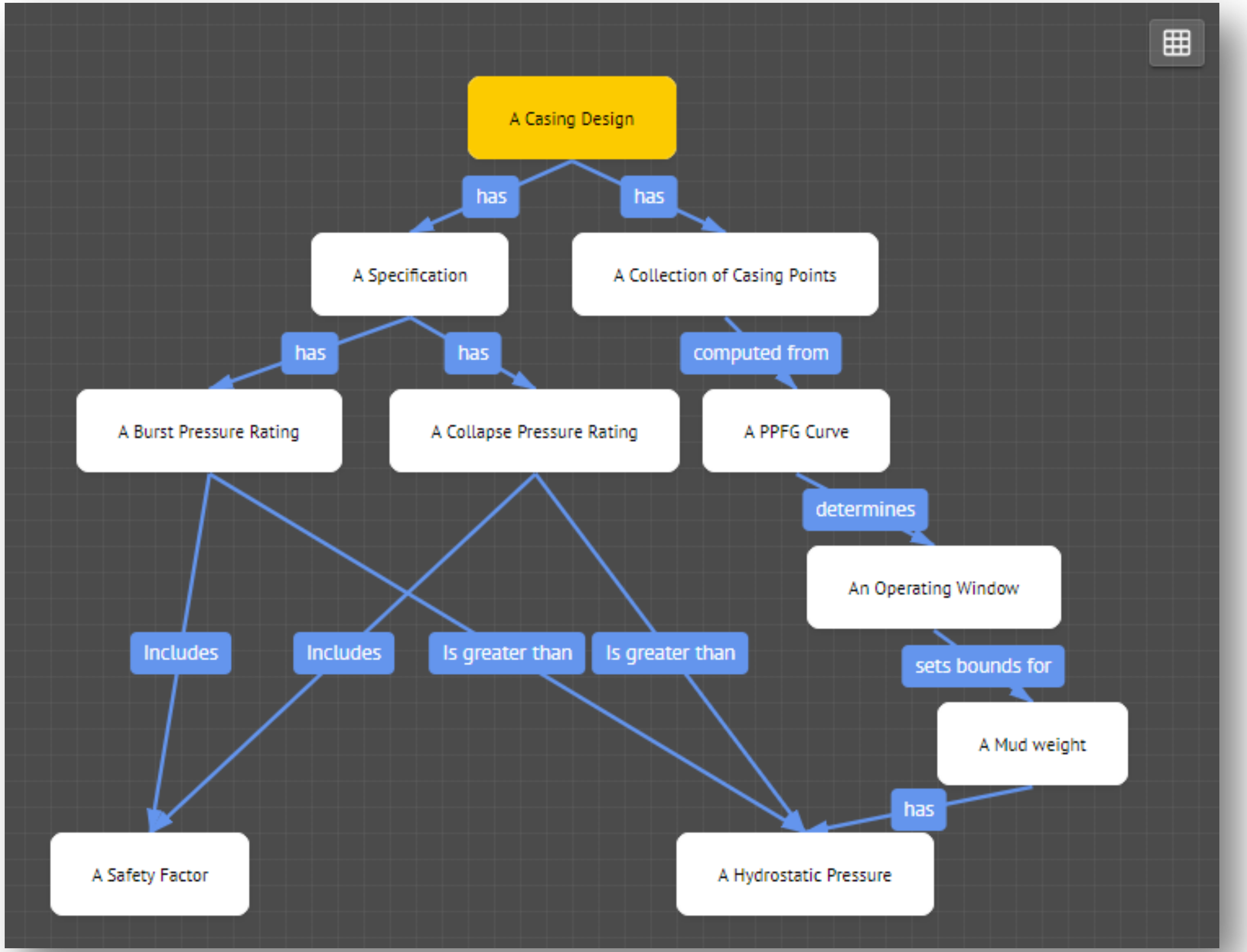




Maana Q Olog Assistant

- Added a new Olog Assistant based on David Spivak's work
    - Initial implementation to gauge interest and uptake by non-developers / mathematicians (i.e., "mortals")
- Categorical knowledge representation
- Separation of concerns
    - Specification: what is a casing design (i.e., requirements)?
    - Instance: a concrete casing design

Not a good olog model →



MAANA Q

Initial Training of SMEs

- Each SME read the first two sections of Spivak's paper
  - Types, Aspects, and Facts
    - No category theory, easily understood by non-mathematicians
- Mostly skipped rest
  - Instances
    - Defining a functor from model category into Set
- Watched Maana Q Olog Assistant training videos
  - Covers first three sections of paper
- Some (remote) hands-on instruction
- Left to model on their own for awhile
- Initial review of their work identified several challenges…

OLOGS: A CATEGORICAL FRAMEWORK FOR KNOWLEDGE REPRESENTATION

DAVID I. SPIVAK AND ROBERT E. KENT

Abstract. In this paper we introduce the olog, or ontology log, a category-theoretic model for knowledge representation (KR). Grounded in formal mathematics, ologs can be rigorously formulated and cross-compared in ways that other KR models (such as semantic networks) cannot. An olog is similar to a relational database schema; in fact an olog can serve as a data repository if desired. Unlike database schemas, which are generally difficult to create or modify, ologs are designed to be user-friendly enough that authoring or reconfiguring an olog is a matter of course rather than a difficult chore. It is hoped that learning to author ologs is much simpler than learning a database definition language, despite their similarity. We describe ologs carefully and illustrate with many examples. As an application we show that any primitive recursive function can be described by an olog. We also show that ologs can be aligned or connected together into a larger network using functors. The various methods of information flow and institutions can then be used to integrate local and global world-views. We finish by providing several different avenues for future research.

Contents

1.	Introduction	1
2.	Types, aspects, and facts	6
3.	Instances	14
4.	Communication between ologs	17
5.	More expressive ologs I	31
6.	More expressive ologs II	40
7.	Further directions	48
References		50

1. Introduction

Scientists have a pressing need to organize their experiments, their data, their results, and their conclusions into a framework such that this work is reusable, transferable, and comparable with the work of other scientists. In this paper, we will discuss the "ontology log" or *olog* as a possibility for such a framework. Ontology is the study of what something *is*, i.e the nature of a given subject, and ologs are designed to record the results of such a study. The structure of ologs is

This project was supported by Office of Naval Research grant: N000141010841 and a generous contribution by Clark Barwick, Jacob Lurie, and the Massachusetts Institute of Technology Department of Mathematics.

Challenges: Semantics

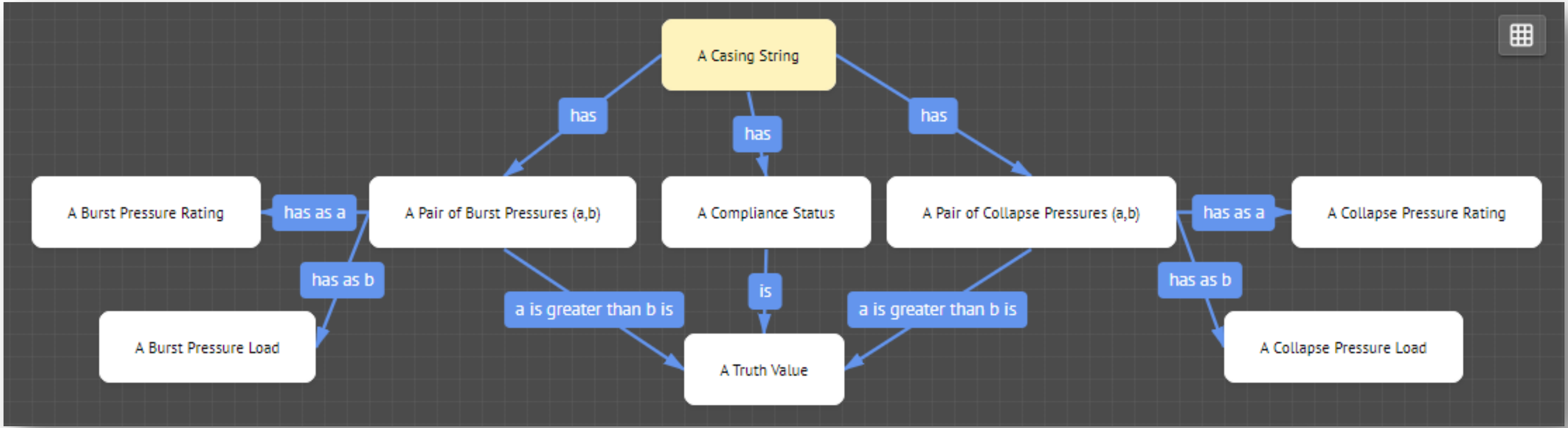
- Ologs are very light on semantics on their own
  - Dependent on interpretation of words used to label boxes
- SMEs wanted to capture more of the semantics in the diagrams
  - I.e., what does it mean to be a well-formed casing design
- Unenforced semantics
  - E.g., a type: `A Pair(Well, Casing String)` is not necessarily a product (i.e., `Well x Casing String`)
  - E.g., subtypes: `A Well that is a Subsea Well`
    - There should be an injections from `Subsea Well` into `Well`

Challenges: Rules of Best Practice

- SMEs treated ologs as a *sentence diagramming* tool
- Confusion when to use different abstractions
  - E.g., a process as Type (should be an Aspect)
  - E.g., a specification as Type (the entire olog is a specification)
- *"Every Type should be labeled with a singular noun"*
  - Used plurals for collections (e.g., a `Well` has a collection of `Casing Strings`)
  - Does not translate nicely to normalized database schemas

Challenges: Facts?

- SMEs treated ologs as *mind maps* or *data models*
- Used to thinking
  - about the properties of things
  - how things work
- They neglected to think about Facts
  - We wrote an example with natural language descriptions followed by more formal mathematical representations
  - E.g., “every well has a total vertical depth (TVD) and TVD > 0”
- Realization of the importance of Facts caused the “a ha!” moment
- Prompted remodeling in order to ensure paths commute



FACTS

DECLARE NEW FACT

OlogFact-e9af738492

Path A

A Casing String has A Compliance Status which is A Truth Value

Path B

A Casing String has A Pair of Burst Pressures (a,b) which a is greater than b is A Truth Value

DELETE

OlogFact-388a186f84

Path A

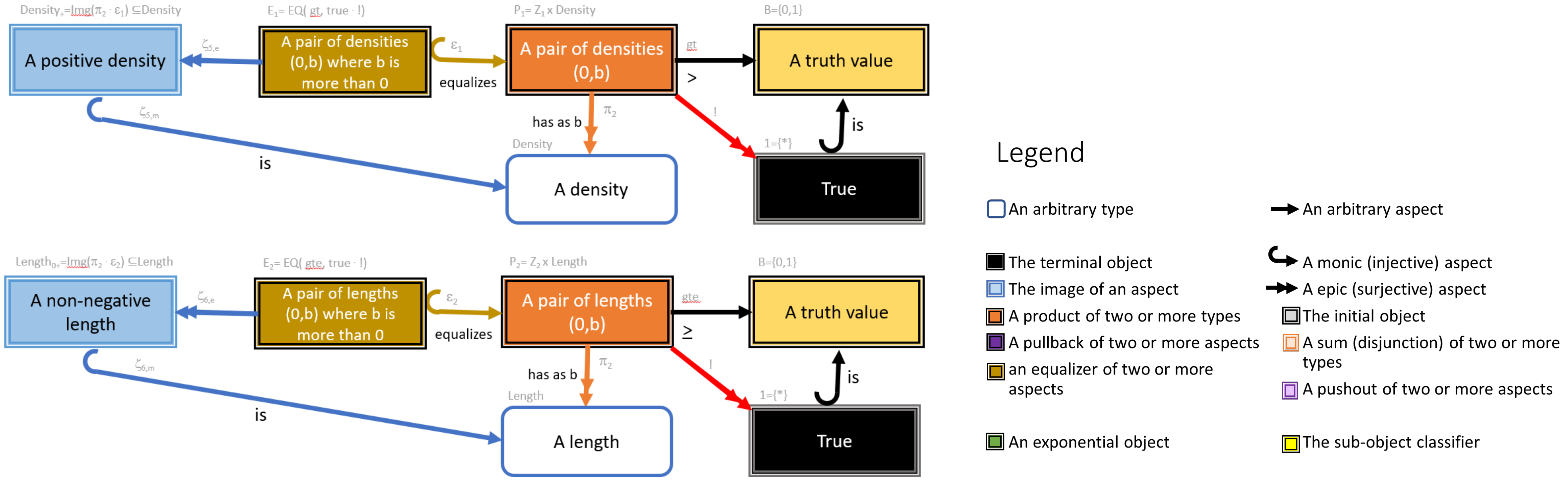
A Casing String has A Pair of Collapse Pressures (a,b) which a is greater than b is A Truth Value

Path B

A Casing String has A Compliance Status which is A Truth Value

DELETE

More Expressive Ologs



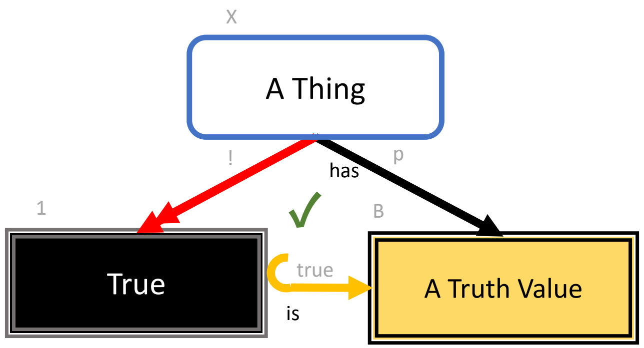
Library of Common Patterns

Pattern for Universally Quantified Predicate

- Given a statement of the form:

  [p is true for all values of x]

- Use the pattern on the right to encode this condition.
- The left-hand path factors through the terminal object, and always returns true.
- If the triangle commutes, then the predicate on the right-hand path must always be true.



Future / Vision

- Use design patterns to drive a DSL over the category theory
- Generalized Algebraic Theories (GATs)
  - Theories
  - Notations
  - Diagrams
  - DSLs