Using Mathematical Model Theory to Align Conceptual and Operational Ontologies in FIBO

Robert Nehmer[1], Mike Bennett[2]

[1]Oakland University, Rochester, Michigan USA
nehmer@oakland.edu
[2]EDM Council, UK
mbennett@edmcouncil.org

Abstract. This paper addresses the relationship between a conceptual ontology and an operational ontology. To date there has been little agreement about what the distinction between these and little consideration about how a conceptual ontology can be operationalized into an operational ontology and vice versa. Where it arises, the discussion is often about whether a particular ontology is a conceptual ontology or an operational ontology. While that discussion is interesting in itself, it masks a deeper discussion. That discussion occurs when a conceptual ontology is created which is intended to be operationalized in a variety of settings. In this paper, we consider the situation of the Financial Industry Business Ontology (FIBO) as promulgated by the Enterprise Data Management Council (EDMC). FIBO's intended use is as a conceptual model which would be operationalized in a variety of projects by various players in the financial industry. Those operationalizations lead to structures which are represented in some form of first order logic, such as OWL. We use model theory to formalize the relationships between the symbols of the operationalization and the interpretation of those symbols in the conceptual ontology.

Keywords: Conceptual ontology, operational ontology, mathematical model theory, interpretation function

1 Overview

In this research, we develop a method which uses concepts from mathematical model theory to inform the relationships between a conceptual ontology, an operational ontology and a software artifact. The research envisions a situation where there is an extant conceptual ontology which is intended to be operationalized in different contexts. It uses the Financial Industry Business Ontology (FIBO) as promulgated by the Enterprise Data Management Counsel (EDMC) in this capacity. The financial industry contains a large number of types of firms, such as retail banks, commercial banks, clearing houses, brokerage firms, etc. We suggest that if a single firm or a consortium of firms of a particular type wanted to use the FIBO conceptual ontology, they would not use the whole ontology but rather would build, let us say, their enterprise ontology as an operationalization of the FIBO concepts which have business value implications in their own firm. So, the operational ontology is for the entire firm. It, in turn, is used to create a variety of software projects which consistently use the operationalized concepts in their own business context. This research uses model theory to provide a general procedure to derive the operational ontology from the conceptual ontology and then derive the projects from the operational ontology.

In the FIBO lexicon, an 'operational ontology' is an RDF/OWL based ontology which is fine-tuned for a specific business purpose. This is distinct from the material in the FIBO OMG specifications or the overall conceptual ontology models for the following reasons:

- Business Conceptual Ontology defines the meanings of things in the domain of discourse. In financial securities, meanings are generally grounded in some legal or financial

concepts; operational ontologies need only define classes and properties for things that for the most part will correspond to instance data;

- Conceptual ontologies cover all the concepts that may be of relevance in some use case - that is, they cover more than one use case;
- The role of a conceptual model and the role of an application data store are very different - even when, as with semantic models, the architecture and model components may be the same. That is, a conceptual model (including an ontology of business terms), is a technology-neutral view of the business requirements, whereas an operational OWL ontology, having instance data, is a physical datastore. Even if they were the same, their roles are different and so are the constraints and requirements under which they are built.
- A conceptual model may use multiple inheritance (polyhierarchical taxonomies) to classify the subject matter in multiple ways. This is appropriate for a conceptual model but adds processing overhead for no real value in an operational application, be it an ontology or a conventional database application.
- The FIBO conceptual models make extensive use of partitioning of the upper parts of the model. This is essential in disambiguating similar terms. Consider for example the difference between a monetary amount as an abstract unit of measure, and a concrete amount of actual money. Such partitioning (in this example Concrete and Abstract), is also polyhierarchical, and so may add some processing overhead to applications, again for no benefit.

The EDM Council and OMG Finance Domain Task Force members are working on developing a series of operational ontologies, starting with derivatives and business entity concepts. In the course of doing this, we expect to more formally identify the differences between these and conceptual ontologies, and home in on the heuristics for extracting or deriving operational ontology content from the conceptual ontologies. This paper is an early attempt at such an effort. At the same time, the conceptual ontologies are developed in a strongly modular fashion, so that for many applications it may be possible to simply use a sub-set of the available modules (from http://www.hypercube.co.uk/edmcouncil/tech-oo.html).

The paper proceeds by first considering mathematical models. Next, it considers how to use those models to describe the relationship between conceptual and operational ontologies. Next, it specifies the general derivation of specific projects from the enterprise's operational ontology. Finally, it demonstrates a small proof of concept using a FIBO fragment. The diagram

below (Figure 1) shows the overall relationships between the conceptual ontology, the operational ontology and the implemented software as modeled in this paper.

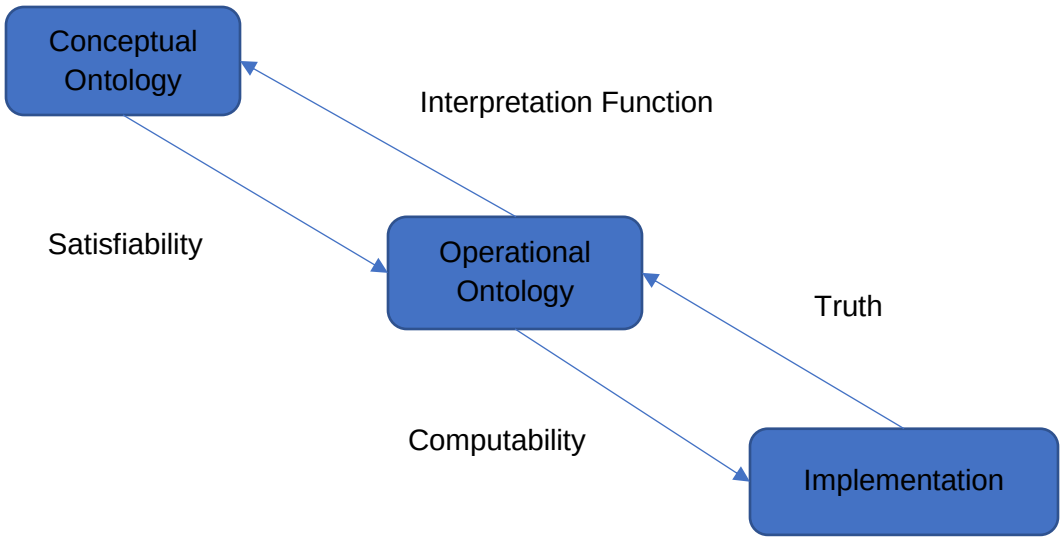

Figure 1. Relations among conceptual and operational ontologies and implementations

2 Mathematical models

The methods chosen to implement systems developed to run on digital devices are all types of first order logic or finite state grammars. These are logic representations used widely in abstract computer science and mathematics. First order logic differs from other logics in that it allows for quantification of function and predicate variables only. In some logics, such as sentential (propositional) logic, no quantifiers are allowed at all. In others, the so called higher order logics, quantification is allowed on the function and predicate symbols themselves, not just on their variables. In these logics, formulas such as $\exists f\ (f(x) \wedge P(x) \longrightarrow f(x) = P(x)$ would be well-formed. Inductive proofs proceed in much the same way in first order systems as they do in finite state grammars. The differences are that more than one axiom may serve as the base, that axioms can be introduced at any time and that the production rules are replaced by rules of inference. The inductive methods appeal to a continuous reapplication of rules is often added as an axiom to systems of formal logic. In axiomatic form, it appears as $(P(O) \wedge \forall n(P(n) \longrightarrow P(n+1))) \longrightarrow \forall x P(x)$. A question which must be addressed is how, in the derivation of formulas by repeatedly applying inference rules, the truth of the resulting formulas is preserved. This is answered by first considering the interpretation function again, then introducing models and, finally, discussing logical truth in models and derivations.

The interpretation function is a map between the symbols of the syntactic language, L, and the objects of a set, M, about which the language has meaningful things to say. Here the set M is the set of concepts in a conceptual ontology. The interpretation must map formulas of L into their meanings in M and in structures of M, the conceptual ontology. Let φ be the interpretation function and divide φ into primary mappings and secondary mappings. The primary mappings will provide the interpretation while the secondary mappings provide the inductive element for complex strings of L. The language is considered systematically through its subsets. The specification is adapted from Manin [1977] p. 24-26.

The specification of the interpretation function leads directly to the concept of a model. Any formula F is said to be φ-true if $\|F\|_{\varphi}(\bar{m}) = 1$ for all $\bar{m}$ in $\bar{M}$. For a set of formulas ∑, M is a model of that set when every formula of ∑ is φ-true. This is sometimes referred to as ∑ being satisfied by M. A subset of these formulas can be constructed as the axioms of a first order logic when the syntactic rules for formula building and derivation are correctly defined. The set of all elements and relations between elements in a model is called the universe of that model, denoted $|A|$, where A is the set of symbols of a model. The model will in most cases not be equivalent to the entire conceptual ontology. The usage of the term 'model' here is restricted to the definition above. Note that 'model' here should not be confused with mathematical models in general, such as linear or nonlinear systems. Nor should it be confused with mathematical modeling techniques. Rather, it is a set theoretic concept and is used as such throughout the discussion.

A model contains a subset of the universe of the language of the model, the conceptual ontology, by definition. Additionally, a model for a language consists of an interpretation function, φ, so that the complete representation of a model is a pair <A,φ> of symbols of the language and the interpretation function. At times the representation of A is broken down into its component sets, depending on the context in which the model is being presented. The interpretation function is the semantic component of the model. It maps the symbols themselves to the appropriate constants, functions, etc. That is to say, the interpretation creates the structure of the model, for instance, requiring that '<' as a symbol be interpreted as 'less than', '+' as 'addition', etc. For this reason, models are sometimes referred to as structures. Another important distinction to make here is the difference between the language and the metalanguage. The language in first order logic consists of the symbols and inferences, as above. The metalanguage in first order logic and model theory is a powerful concept which allows an analysis of the language itself. The distinction is between a sentence of the language such as ∀(x,y,z)(S(x) = S(y) + S(z)) whose interpretation is, say, 'for all asset, liability, and equity accounts, the sum of the asset account balances equals the sum of the liability account balances plus the sum of the asset account balances' and a sentence about sentences in the language.

3 Model theoretic constructs between conceptual and operational ontologies

The concept of a model can also be used to define the truth of formulas in a particular model. The idea of truth is closely connected to validity, completeness, and consistency as well as the method of proof employed in first order logic. The following discussion begins with the semantic, metalanguage of truth and proceeds to a discussion of the syntactic notion of deduction. It ends with a specification of how validity, completeness, and consistency are proved in first order logic.

The standard abbreviated notation for a model is to use German capital letters, usually $\mathfrak{A}$ and $\mathfrak{B}$. The string '$\mathfrak{A} \vDash \sigma$' means that σ is true in model $\mathfrak{A}$. This truth is defined precisely by the interpretation function developed above and is therefore at the semantic level of the language. The symbol $|\mathfrak{A}|$ is used to indicate the objects of $\mathfrak{A}$, such as accounts and ATs, of which the language speaks. These objects, of course, would not include the logical operators, predicates, etc., but only those objects which are terms, that is, constants, variables, and functions. Thus $|\mathfrak{A}| = M$ in the discussion of the interpretation function. Suppose ∑ is a set of formulas and σ is one particular formula, then $\Sigma \vDash \sigma$ means that ∑ logically implies σ. In terms of the model, the interpretation is as follows. For every $\bar{m}$ in which $\mathfrak{A}$ satisfies all formulas of ∑, $\bar{m}$ also satisfies σ. Notice that ∑ can be empty in which case σ is referred to as a valid formula or a tautology. These two concepts are explained in more detail below.

The symbol '$\vDash_{\mathfrak{A}} \sigma$' is used to mean that σ is logically implied in the model $\mathfrak{A}$. This notation is used when two or more different models are being considered. As such, elementary equivalence is alternatively defined as when any structure $\mathfrak{A}$ is a model for an arbitrary formula σ if and only if $\mathfrak{B}$ is also a model of σ, then $\mathfrak{A}$ and $\mathfrak{B}$ are elementary equivalent, that is $\mathfrak{A} \equiv \mathfrak{B}$. This

means that the two models cannot be distinguished in first order formulas. If, however, the weaker condition that $\vDash_{\mathfrak{A}} \sigma \Longrightarrow \vDash_{\mathfrak{B}} \sigma$ obtains among two models, where $\Longrightarrow$ again is the implication operator of the metalanguage, then $\mathfrak{A}$ is called a substructure of $\mathfrak{B}$.

We now proceed to make this more specific by constructing an operational ontology from a conceptual ontology. Doing this is equivalent to constructing $\vDash_{\mathfrak{A}}$ as follows. This requires that as the components of M, the conceptual ontology, are chosen to be designed into the operational ontology, and that this is done in a way which is definable in M. That is, constructing the model $\mathfrak{A}$ is constrained to models that are satisfiable. It is important to note that not all models of theories (sets of sentences) constructed from the conceptual ontology are guaranteed to be satisfiable. The design criteria, done correctly, limits the elements of M that can be designed into the model and their relationships to those which can be satisfied in the structure.

If we consider the case in the other direction, that is, defining the conceptual model from an operational model, situation is considerably simplified. In this case, the operation is equivalent to the designing of φ, the interpretation function. If we assume that the language in which the operational ontology was developed is satisfiable in the sense of a language being a set of true sentences, then we are guaranteed that there exists a model of the operational ontology. Note that in this case, $\mathfrak{A}$ will be equivalent to M unless more work is done to extend the concepts in M, the conceptual ontology.

4 Model theoretic constructs between operational ontologies and first order logic

Although of less importance to this paper, we consider the relationship between an operational ontology in the sense of this paper and its underlying language represented by strings in OWL or some other formulation. Again, we can consider both directions: the operational ontology to the formal language and its opposite, the language to the operational ontology. Consider ontology to language first. This requires that a computable language is designed which will satisfy truth in the operational ontology. It would not necessarily require that the entire ontology be satisfied but only the components necessary for a particular implementation of the ontology. In the other direction, from language to operational ontology, what is required is to build a true language. That is, the language would be constructed so that it is provable in some ontology. That ontology operationalizes the language and is its operational ontology. It would not, however, be necessarily meaningful in the sense of being interpretable in a conceptual ontology unless, of course, we continued the design process and created an interpretation function for it.

5 Proof of concept using FIBO

Consider the FIBO fragments below (Figures 2 and 3). The first is the REA claims concept diagram from the FIBO conceptual model. Here an REA claim (C) is defined on the part of an accounting reporting party (ARP). It occurs when there is a remaining imbalance in a commitment, such as in a contract which has not yet been completely fulfilled. It can be a debt claim (DC) or an ownership claim (OC). To build the interpretation function, really a set of functions, we map the symbols of the second fragment onto the semantics of the first fragment.

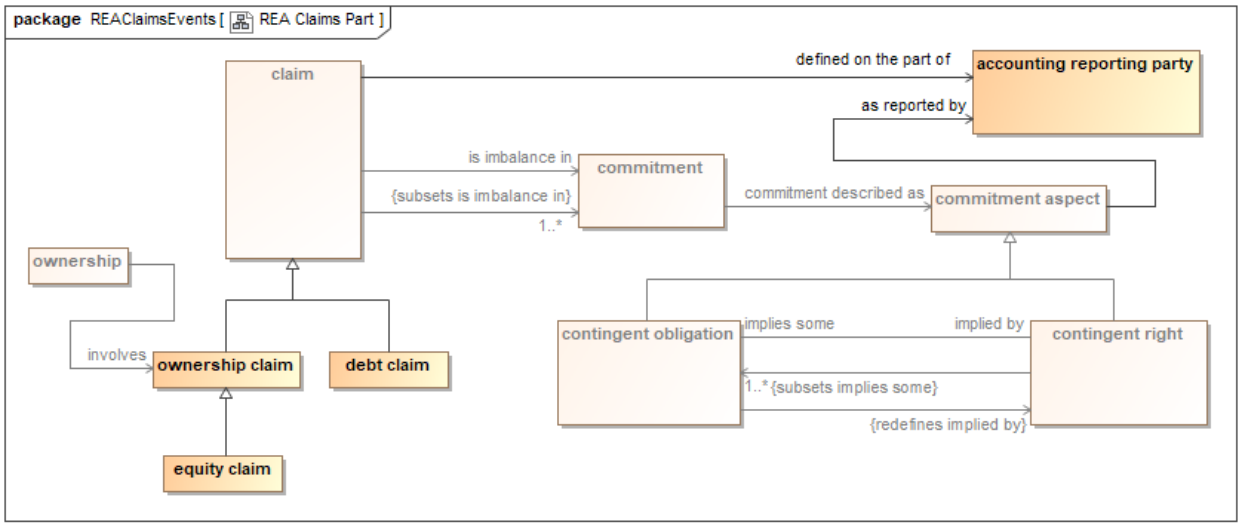

Figure 2. REA Claims in FIBO Conceptual Ontology

The second, related fragment has three terms and two relations on the buyer subtree. The terms are Buyer (B), Payment Obligation (PO) and Delivery Right (DR). The two relations are has Payment Obligation (hPO) and has Delivery Right (hDR). The interpretation function is { (B → ARP), ((PO, DR) → DC) } Notice that in FIBO as it exists, conceptually there is no distinction between a buyer and seller. This is because of the REA duality concept. Yet when the model is operationalized, the two aspects of buying and selling separate or unfold. Using Polish Notation, a satisfiable structure, again for the buyer alone, in this simple example is {hPO(B, PO), hDR(B, DR)}. This and the interpretation function is a model of the fragment. The implementation in first order logic can be represented simply as { hPO(B ∧ PO) ⊢ T; hDR(B ⋀ DR) ⊢ T } where T is “true.”

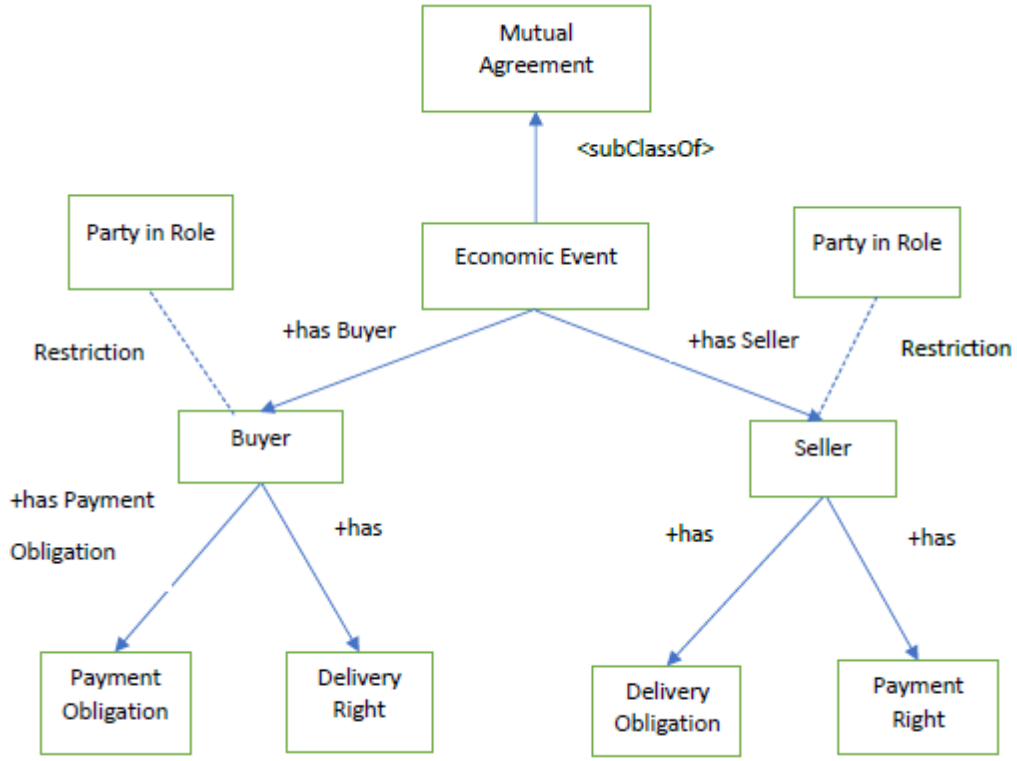

Figure 3. Payment and Delivery Obligations in Operational Ontology

References

1. Enderton, H. B. *A Mathematical Introduction to Logic*. Orlando: Academic Press, 1972.
2. Falbo, R. A.; Guizzardi, G.; Gangemi, A. and Presutti, V. (2011). Ontology Patterns: Clarifying Concepts and Terminology. Springer-Verlag Berlin Heidelberg.
3. http://www.hypercube.co.uk/edmcouncil/tech-oo.html
4. Lemaignan, Severin; Siadat, Ali; Dantan, Jean-Yves and Semenenko, Anatoli (2006). MASON: A Proposal For An Ontology Of Manufacturing Domain. Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06).
5. Long, F.; Zhang, L. and Wang, C. (2013). A Conceptual Logic-based Creation Method for Operational Plan Ontology Class Hierarchies. Fourth Global Congress on Intelligent Systems.
6. Manin, Y. I. *A Course in Mathematical Logic*. New York: Springer-Verlag, 1977.
7. Sivashanmugam, K., Verma, K., Sheth, A. P., & Miller, J. (2003). Adding Semantics to Web Services Standards. http://corescholar.libraries.wright.edu/knoesis/687