Ontology Validation as Dialogue

Michael GRÜNINGER [a]

[a] *Department of Mechanical & Industrial Engineering, University of Toronto, Canada*

Abstract. Ontology verification is concerned with the relationship between the intended models of an ontology and the models of the axiomatization of the ontology. An even more difficult challenge is ontology validation – are the intended models of the ontology indeed the correct models for the ontology? The identification and specification of the intended models of an ontology is inherently a dialogue between the user of the ontology (who implicitly knows the intended models) and the ontology designer (who is attempting to axiomatize the class of intended models). In this paper we explore techniques for eliciting intended models from the user and evaluating these models with respect to user responses. We consider how this procedure can be applied to several first-order ontologies.

Keywords. ontology design, verification, validation, intended models

1. Introduction

The design of high quality ontologies is an overarching challenge for the applied ontology community. Following Ontology Summit 2015, we consider ontologies to be artefacts that are designed with respect to a set of requirements, analogous to software engineering. In particular, these requirements may be characterized by the class of structures which capture the intended semantics of the signature of the ontology. If we carry the analogy of software engineering further, we see that ontology evaluation also focuses on verification and validation. Ontology verification is concerned with the relationship between the intended models of an ontology and the models of the axiomatization of the ontology. An even more difficult challenge is ontology validation – are the intended models of the ontology indeed the correct models for the ontology?

In this paper, we take the position that the identification and specification of the intended models of an ontology is inherently a dialogue between the user of the ontology (who implicitly knows the intended models) and the ontology designer (who is attempting to axiomatize the class of intended models). After an overview of ontology verification and validation, we motivate the need for an interactive approach by considering three large first-order ontologies. We then propose a procedure for eliciting intended models from the user and evaluating these models with respect to user responses.

2. Ontology Evaluation

2.1. Ontology Verification

When selecting or developing an axiomatization of an ontology[1] for an application domain, the knowledge engineer typically has some requirements in mind. These requirements for the ontology are specified with respect to the intended semantics of the terminology; from a mathematical perspective the requirements may be characterized by the class of structures which capture the intended semantics, and such structures can be referred to as the required structures for the ontology. Previous work in this approach [1,2] has focused on the case in which the required models and the axioms of the ontology have the same signature. In this case, the correctness of the ontology is defined with respect to the relationship between the required structures for the ontology (which we will denote by $\mathfrak{M}^{req}$) and the models of its axiomatization T_{ont} (see Figure 1). If the ontology is too weak, then there exist models which are spurious (that is, they are not required structures):

$$\mathscr{M} \in Mod(T_{ont}) \text{ and } \mathscr{M} \notin \mathfrak{M}^{req}$$

If the ontology is too strong, then there exist required models which are omitted:

$$\mathscr{M} \in \mathfrak{M}^{req} \text{ and } \mathscr{M} \notin Mod(T_{ont})$$

In other words, an axiomatization is correct if and only if it does not include any spurious models, *and* it does not omit any required models:

$$\mathscr{M} \in Mod(T_{ont}) \text{ iff } \mathscr{M} \in \mathfrak{M}^{req} \qquad \textbf{(C)}$$

Verification is therefore concerned with the relationship between the intended models of an ontology and the models of the axiomatization of the ontology. In particular, we want to characterize the models of an ontology up to isomorphism and determine whether or not these models are equivalent to the intended models of the ontology. This relationship between the intended models and the models of the axiomatization plays a key role in the application of ontologies in areas such as semantic integration and decision support.

We can say that two software systems are semantically integrated if their sets of intended models are equivalent. However, systems cannot exchange the models themselves – they can only exchange sentences in the formal language that they use to represent their knowledge. We must be able to guarantee that the inferences made with sentences exchanged in this way are equivalent to the inferences made with respect to the system's intended models – given some input the application uses these intended models to infer the correct output.

In the area of decision support, the verification of an ontology allows us to make the claim that any inferences drawn by a reasoning engine using the ontology are ac-

[1] By an ontology, we mean an axiomatic theory in the language of first-order logic. We, therefore, use the words 'ontology' and 'theory' interchangeably. We consider a theory to be a set of first-order sentences closed under logical entailment, and a subtheory to be a subset of the corresponding theory. For a theory T, $Mod(T)$ denotes the class of all models of T.

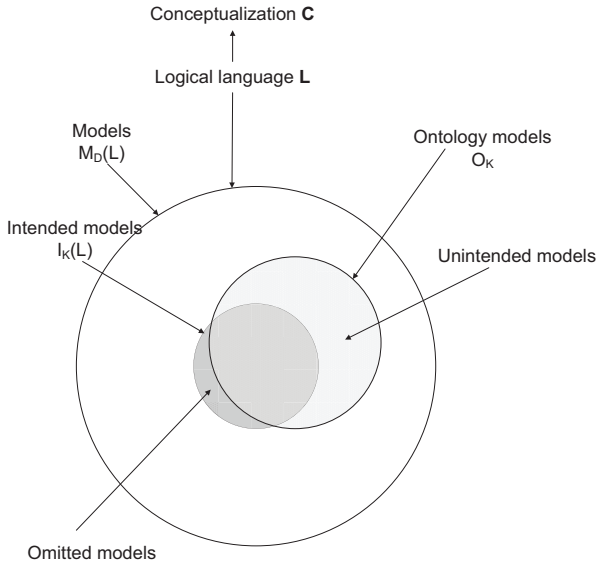

Figure 1. The possible relationships between the models of an ontology and the required structures (modified after [2]). Note that Guarino's notions of intended and unintended models corresponds to our notions of required and spurious structures.

tually entailed by the ontology's intended models. If an ontology's axiomatization has unintended models, then it is possible to find sentences that are entailed by the intended models, but which are not provable from the axioms of the ontology.

Unfortunately, it can be quite difficult to characterize the models of an ontology up to isomorphism. Ideally, since the classes of structures that are isomorphic to an ontology's models often have their own axiomatizations, we should be able to reuse the characterizations of these other structures.

2.2. Ontology Validation

If ontology verification is concerned with the relationship between the models of the axioms of the ontology and the intended models for the ontology, ontology validation addresses the problem of whether the intended models are actually the right ones for the user. Following the Ontology Application Framework developed during Ontology Summit 2013, there are three major kinds of applications that motivate the design of ontologies – search, decision support, and semantic integration. Ontology validation seeks to evaluate the ontology with respect to its application.

When integrating software applications, we are faced with the additional challenge that almost no application has an explicitly axiomatized ontology. We can model a software application as if it were an inference system with a formal ontology, and use this ontology to predict the set of sentences that the inference system decides to be entailed or satisfiable. This approach, known as the Ontological Stance ([3]), is an operational characterization of the set of intended models for the application's terminology.

The Ontological Stance also addresses the relationship between the intuitions and the intended models; this is, of course, informal, but we can consider the domain intu-

itions as providing a physical interpretation of the intended models. In this sense, we can adopt an experimental or empirical approach to the evaluation of the class of intended models in which we attempt to falsify these models. If we can find some objects or behaviours within the domain that do not correspond to an intended model, then we have provided a counterexample to the class of model. In response, we can either redefine the scope of the class of models (i.e. we do not include the behaviour within the characterization of the models) or we can modify the definition of the class of models so that they capture the new behaviour.

If a software application already has an ontology, then the Ontological Stance can be used to evaluate the ontology with respect to the application's intended models. This is distinct from ontology verification – even if we have a complete characterization of all models of the ontology's axiomatization, the models of the ontology may not be intended models. Using the Ontological Stance, if the models of the application ontology are not equivalent to the intended models, then there are two possibilities:

1. there exists a model $\mathscr{M}$ of the ontology's axioms T that is not an intended model;
2. the intended models entail some sentence Φ that is not entailed by the axioms T.

These two possibilities provide the fundamental distinctions that we need to implement any procedure for ontology validation.

3. Case Studies

The challenges of ontology verification and validation can be illustrated with several large first-order ontologies.

3.1. MoSt

The field of medicinal chemistry involves the design, synthesis, and development of new drugs that can be further enhanced with the application of ontologies. The work of [4] introduces a molecular structure ontology MoSt to aid in the task of drug discovery. MoSt combines conventional graph theory and ontological approaches to describe the shape of molecules. This ontology allows us to consider molecules from the shape perspective by identifying basic functional groups of the ring and chain types, and to use the axioms of the ontology to combine these functional groups together.

The verification of MoSt proves that models of MoSt correspond to the class of tripartite incidence structures for cyclic and path subgraphs of a graph. Intuitively, the graph is isomorphic to the structure axiomatized by ontologies for conventional chemical graph theory, in which atoms are the vertices and bonds are the edges. Primitive functional groups are either chains (which are path subgraphs of the molecular graph) or rings (which are cyclic subgraphs of the molecular graph).

Intended models of MoSt correspond to physically realizable molecules, so that the validation problem is composed of two parts:

1. Every molecule is a model of the ontology.
2. Every model of the ontology is a molecule.

Molecules are Models of MoSt Molecules that currently exist and are chemically feasible are named in accordance with the IUPAC naming rules from which an unambiguous structural formula can be created, which in turn can be used to build the underlying chemical graph. The MoSt Ontology axiomatizes the chemical graphs by taking the graphs and decomposing them into their primitive functional groups, and then re-composing them again via an attachment theorem. Furthermore, MoSt has the property that two models of the ontology can be combined together to form another model of the ontology. Consequently, we can say that all IUPAC-named molecules that have underlying chemical graphs are therefore models of the ontology.

Models of MoSt are (Chemically Feasible) Molecules Conversely, a model of MoSt is essentially a chemical graph: we can give this graph to a chemist and ask her to determine whether it is possible to synthesize the molecule. Whether or not a chemical graph can be physically realized is dependent on the difficulty of the synthesis process, along with the number of steps required. The motivation for MoSt is that a model can result in a potential molecule that has not yet been discovered and physically realized. Drug discovery becomes the process of building models by answering queries with the ontology and constructing models of the ontology using additional constraints and extensions.

To prove that the models of MoSt are indeed molecules therefore requires a dialogue with users of the ontology. Models of MoSt are generated and then presented to medicinal chemists. If they reject a model (i.e. it is not a feasible molecule), we would need to generate new axioms to eliminate the model. In other words, user interaction is needed to identify and eliminate unintended models. To guarantee that there are no omitted models (i.e. physically feasible molecules that are not models of MoSt), we can search through existing molecule knowledge bases (e.g. ChemSpider) to verify that each molecule is a model of MoSt. We first generate molecular descriptions from IUPAC names and then construct models of the molecular descriptions; in such a case, no user interaction is required.

3.2. BoxWorld

The BoxWorld Ontology $T_{boxworld}$ ([5]) is a modular first-order ontology of shape that supports intelligent manufacturing and commonsense reasoning about cutting, joining, and folding sheets of materials such as cardboard and metal. The verification of $T_{boxworld}$ demonstrates that models of $T_{boxworld}$ are equivalent to mathematical structures that are generalizations of polyhedra and abstract polytopes. The question remains, though, as to exactly which shapes are models of $T_{boxworld}$. In particular, are all intuitive shapes also models of the axioms? Are there models of the axioms which do not correspond to intuitive shapes? $T_{boxworld}$ has no omitted models iff all physically possible shapes are models of $T_{boxworld}$. $T_{boxworld}$ has no unintended models iff all models of $T_{boxworld}$ are physically possible shapes. To some extent, ontology verification can address this by providing representation theorems that characterize the models up to isomorphism; however, by itself, this does not give any indication of whether or not the models are intended.

Who judges the feasibility of a shape? An additional difficulty in this regard is that the answer might indeed vary across different users. An ontology for shapes composed of surfaces also has applications in sheet metal fabrication, in which products are man-

ufactured by folding, cutting, and joining pieces of sheet metal. A shape ontology for sheet metal processes would be able to answer questions such as:

- *What objects can we make from a single sheet of metal, either by cutting or folding?*
- *What objects can we make from joining multiple surfaces?*

On the other hand, an ontology of shape for furniture might well have a different set of intended models. As a result, there is no single correct answer to the problem of specifying intended models. Instead, we need a flexible way of allowing different users to interact with the ontology and specify those models that are intended from their perspectives.

3.3. CardWorld Vision

The CardWorld Vision Ontology $T_{cardworld\_vision}$ provides an axiomatization of images and depiction for 2D polygonal surfaces in scenes with occlusion and noise (errors in edge detection). It consists of eight modules that axiomatize intuitions about scenes, images, depiction, and occlusion for 2D polygonal surfaces. Strictly speaking, we only need to show that a model exists in order to demonstrate that a theory is satisfiable. However, for ontology validation, we need a complete characterization of the possible models. For example, since we are considering the domain of computer vision, to show that a theory is satisfiable, we need only specify an image and scene which together with the axioms are satisfied by some structure. The problem with this approach is that we run the risk of having demonstrated satisfiability only for some restricted class of images, scenes, or surfaces. For example, a theory of scenes and images may be shown to be consistent by constructing a satisfying interpretation, but the interpretation may require that there is no occlusion in the scene; although such a model may be adequate for such scenes, it would in no way be general enough for our purposes. $T_{cardworld\_vision}$ is supposed to support a comprehensive theory of 2D image interpretation, so we need to explicitly characterize the classes of images, scenes, surfaces, and other assumptions which are guaranteed to be satisfied by the specified structures. For example, within CardWorld, we are only considering 2D surfaces which are opaque, which are not self-occluding, which cannot interpenetrate each other. In addition, we are not considering scene features such as colour, surfaces markings, or texture. Since these properties are not considered within the scope of the ontology, they are not formalized within any structures. Of course, if someone wished to represent these properties, then they would need to extend the class of intended structures appropriately.

$T_{cardworld\_vision}$ has no omitted models iff all interpretations of the image are models of $T_{cardworld\_vision}$. In other words, an omitted model exists if there exists a possible interpretation of the image that is not a model. On the other hand, $T_{cardworld\_vision}$ has no unintended models iff all models of $T_{cardworld\_vision}$ are possible interpretation. Unintended models are not a problem for $T_{cardworld\_vision}$, but omitted models are – how do we have a sense of all possible interpretations of an image, particularly in cases where there is heavy noise? One approach is to use annotation by users – given an image, the user specifies which scene objects are depicted by which image objects, which image objects are noise, and which objects are undepicted (either by occlusion or because of noise). However, this typically identifies only one possible interpretation of the image. People

abandon such a preferred interpretation only when presented with refuting evidence (in which case the interpretation is not actually a model).

The relationship between the intuitions and the structures is, of course, informal, but we can consider the domain intuitions as providing a physical interpretation of the structures. In this sense, we can adopt an experimental or empirical approach to the evaluation of the class of intended structures in which we attempt to falsify these structures. If we can find some objects or behaviour within the domain which do not correspond to an intended structure, then we have provided a counterexample to the class of structures. In response, we can either redefine the scope of the class of structures (i.e. we do not include the behaviour within the characterization of the structures) or we can modify the definition of the class of structures so that they capture the new behaviour.

3.4. Upper Ontologies

Upper ontologies characterize the semantics of general concepts that underlay every knowledge representation enterprise. Since upper ontologies are expected to be broadly reused, verifying that they do not have unintended models and that they are not missing any intended models are of paramount interest for the knowledge representation community. Work has been initiated on the verification of upper ontologies. For example, in [6] and [7], a number of novel classes of mathematical structures were axiomatized in order to adequately represent the models of DOLCE.

A much more difficult challenge is the validation of upper ontologies. How do we discover the models that the designers of upper ontologies intended? For example, the work of [8], a model of SUMO-Time was discovered in which there was not a linear ordering over the set of timepoints. This is unusual insofar as most other time ontologies enforce linearity, but absent any interaction with the ontology designer it is not clear whether this should in fact be allowed as an intended model.

4. Finding the Right Models

In each of the ontologies discussed in the preceding section, we noted the role that the ontology user plays in identifying which models are unintended and which intended models are omitted. We now consider procedures that can be used to support the validation of an ontology as a dialogue between the ontology curator and the ontology user/designer.

At the heart of any such procedure is the identification of unintended and omitted models, followed by decisions about how to modify the ontology in light of the existence of any such models. For example, with an early version of $T_{boxworld}$, a proof could not be found for the competency question stating that an edge cannot meet another edge at two distinct points:

$$(\forall e_1, e_2, e_3, p_1, p_2) meet(e_1, e_2, p_1) \wedge meet(e_1, e_3, p_2)$$
$$\wedge \neg(p_1 = p_2) \supset \neg(e_2 = e_3)$$

In this case, no proof existed because there existed a model of the axioms that did not satisfy this sentence, that is, there existed a surface with a hole consisting of only two

edges. But is this model indeed unintended? If so, we need to extend the axioms. If not, then we need to relax the specification of the class of intended models.

On the other hand, intended models of an ontology are omitted if the axioms of the ontology are not satisfied by the intended models, that is, the ontology is too strong. In this case, a proof can be found for a sentence that contradicts the intended semantics of the ontology. Given this possibility, a thorough inspection of all proofs found must be performed; if this case is detected, it is indicative of at least one of two possible errors with the ontology. First, an examination of the proof may lead to the identification of some axiom in the ontology which is not entailed by the intended models, and the problem can be resolved by removing the axiom. A second possible error arises when examination of the proof leads to the detection of some error in the definition of the requirements, that is, in the specification of the intended models. It is important to devote considerable attention to the detection of this possibility so that the ontology is not revised to satisfy incorrect requirements.

For example, the following sentence was entailed from an early version of $T_{boxworld}$: Every edge meets at most two distinct edges. Upon inspection, however, it was discovered this was actually a proof of:

$$T_{boxworld} \models (\neg\exists p)\mathit{point}(p)$$

Examination of the proof showed that the axiom:

$$(\forall e, p_1, p_2, p_3)\mathit{edge}(e) \wedge \mathit{point}(p_1) \wedge \mathit{point}(p_2) \wedge \mathit{point}(p_3) \wedge \mathit{part}(p_1, e)$$
$$\wedge \mathit{part}(p_2, e) \wedge \mathit{part}(p_3, e) \supset (p_1 = p_3) \vee (p_2 = p_3)$$

was incorrect, and needed to be weakened. We will refer to this as the existence of an *unintended proof*, since it is not a sentence that should be provable from the ontology.

It is interesting to see the relationship between this case and the failure of verification for the ontology. Part of the methodology for verification shows that a sentence that is entailed by axioms is also entailed by the set of intended models. Hidden consequences such as the above are counterexamples to this part of the verification, since it is a sentence that is provable from the axioms, yet it is not entailed by the intended models. One can either weaken the axioms (so that the sentence is no longer provable), or one can strengthen the requirements by restricting the class of intended models (so that the sentence is entailed by all intended models).

4.1. Hashemi Procedure

[9] proposed an interactive procedure for using an ontology repository to find the axiomatization of an ontology. The Hashemi Procedure finds the best match between a theory in a given ontology repository and the set of intended and unintended models as identified by an ontology designer. Since the class of intended models $\mathfrak{M}^{intended}$ is not always explicitly specified, the system elicits a subset $\mathfrak{N}$ from the user. It is not enough to jump from $\mathfrak{N}$ to a theory$T$. Further interaction is required, in which the system provides models of existing ontologies and the user identifies them as either intended or not.

The Hashemi Procedure presumes that there is an existing set of axiomatized theories with the same signature (known as a hierarchy in [10]):

Definition 1. *A hierarchy $\mathbb{H} = \langle \mathscr{H}, \leq \rangle$ is a partially ordered, finite set of theories $\mathscr{H} = T_1, ..., T_n$ such that*

1. $\Sigma(T_i) = \Sigma(T_j)$, *for all* i, j;
2. $T_1 \leq T_2$ *iff* T_2 *is an extension of* T_1;
3. $T_1 < T_2$ *iff* T_2 *is a non-conservative extension of* T_1.

The Procedure consists of two parts – elicitation of user models and the proposal of models for existing ontologies. The first component locates the user somewhere in the hierarchy, providing "bounds" for theories which characterize the user's intended models. In the second part, models of existing ontologies, coupled with user responses, tighten this bound, resulting in selecting the strongest (if any) theories from the hierarchy which capture the user's intuition.

4.2. The Dialogue of Validation

The Hashemi Procedure was actually motivated by the problem of ontology design rather than ontology verification and validation. It was assumed that the user knows the class of intended models but does not know how to axiomatize them. In the current paper, we are considering a different problem – we have the axioms of the ontology, but we are unsure whether or not the models of the axioms are equivalent to the intended models, and we require interaction with the user to validate the ontology.

We start with a verified ontology T and crawl through the hierarchy by generating unintended proofs and unintended models. The first *InsertTheory* Procedure adds T to the hierarchy; this is a modification of the *FindTheory* Procedure introduced by [10]. Beginning with the root theory in a core hierarchy $\mathbb{C}$, the procedure searches[2] through $\mathbb{C}$ to find the maximal theories that are interpretable by T. If there are multiple maximal theories, the procedure returns the union of their sets of axioms. The result completely specifies which theories in the hierarchy are weaker than T and which theories are stronger than T.

Within the *EliminateUnintende* Procedure, the user distinguishes intended and unintended models of T that are generated from the ontology [3] Once an unintended model has been identified, we find the weakest extension of T that eliminate these models. Since extending T might lead to a theory that is too strong and omits models, the *WeakenProof* Procedure allows the user to select which axioms to eliminate from potential unintended proofs generated from the ontology.

This dialogue with the user, through alternating requests for unintended models and unintended proofs, leads to a theory whose models is the closest approximation to the class of intended models.

Definition 2. *A near-miss for a class of intended structures $\mathfrak{M}^{intended}$ in a hierarchy $\mathbb{H}$ is a theory T^{miss} that has no omitted models with respect to $\mathfrak{M}^{intended}$, and all extensions of T^{miss} in $\mathbb{H}$ have either unintended models or omitted models.*

[2] We assume the existence of three other algorithms related to partial orderings. The first is $ChainDecomp(P)$ which outputs the set of chains for a poset P, and the second is $PosetSort(X,P)$, which constructs a poset P from an unordered set X. The third algorithm is $NextTheory(T,P)$ which returns the elements in the poset P that covers the element T.

[3] In the procedures introduced in this section, we restrict our attention to theories that have finite models, a property that holds for all of the case study ontologies considered in Section 3.

Procedure 1 $InsertTheory(\mathbb{H}, T, \mathbb{H}')$

Require: Hierarchy $\mathbb{H} = \langle \mathscr{T}, \sqsubseteq \rangle$, theory $T \notin \mathscr{T}$.
Ensure: Hierarchy $\mathbb{H}' = \langle \mathscr{T} \cup \{T\}, \sqsubseteq \rangle$.

$Chains_i \leftarrow ChainDecomp(\mathbb{C}_i)$
$Collect_i \leftarrow \emptyset$
for all $G_{ij} \in Chains_i$ **do**
  $Candidate_{ij} \leftarrow \emptyset$
  $T_{current} \leftarrow$ minimal theory in G_{ij}
  $T_{max} \leftarrow$ maximal theory in G_{ij}
  while $T_{current} \neq T_{max}$ **do**
    if $T \cup \Delta_i \models T_{current}$ and $T \cup \Delta_i$ is a conservative extension of $T_{current}$ **then**
      $Candidate_{ij} \leftarrow T_{current}$
      $T_{current} \leftarrow NextTheory(G_{ij}, T_{current})$
    else
      $T_{current} \leftarrow T_{max}$
    end if
  end while
end for
$Collect_i \leftarrow \bigcup_j \{Candidate_{ij}\}$
$PosetSort(Collect_i, P)$
$T_i \leftarrow$ set of axioms in the union of maximal theories in P
$\mathscr{T} \leftarrow \mathscr{T} \cup \{T_i\}$

Procedure 2 $EliminateUnintended(T, \mathbb{H}, T')$

Require: Theory T, Hierarchy $\mathbb{H} = \langle \mathscr{T}, \sqsubseteq \rangle$.
Ensure: Theory T' with no unintended models.

Generate $\mathfrak{M} \subseteq Mod(T)$
User selects subset $\mathfrak{N} \subseteq \mathfrak{M}$ of unintended models.
$T' \leftarrow$ weakest extension of T in $\mathbb{H}$ such that $\mathfrak{N} \cap Mod(T') = \emptyset$

Procedure 3 $WeakenProof(T, \mathbb{H}, T')$

Require: Theory T, Hierarchy $\mathbb{H} = \langle \mathscr{T}, \sqsubseteq \rangle$.
Ensure: Theory T' with no omitted models.

$\Theta(T) \leftarrow \{\theta : \theta \in S < T$ such that θ is not an axiom in $T\}$
for all $\theta \in \Theta(T)$ **do**
  Generate proof of θ
  User selects subset $\Sigma \subset T$ of sentences that should be eliminated
  $EliminateUnintended(T \setminus \Sigma, T')$
  if $T' < T$ **then**
    $WeakenProof(T', \mathbb{H}, T^*)$
  end if
end for

Procedure 4 $Validate(T, \mathbb{H}, T^*)$

Require: Theory T, Hierarchy $\mathbb{H} = \langle \mathscr{T}, \sqsubseteq \rangle$.
Ensure: Near-miss theory T'.
$InsertTheory(\mathbb{H}, T, \mathbb{H}')$
$EliminateUnintended(T, \mathbb{H}, T')$
$WeakenProof(T, \mathbb{H}', T^*)$

It is easy to see from this definition that a near miss with no unintended models has the property that $Mod(T^{miss})$ is equivalent to $\mathfrak{M}^{intended}$.

Theorem 1. *Given a theory T and hierarchy $\mathbb{H}$, if the $Validate(T, T^*)$ Procedure terminates, then T^* is a near-miss theory for the class of intended structures provided by the user.*

5. Summary

We have presented a procedure which incorporates interactions with an ontology user to arrive at an ontology whose models are the closest approximation to the class of intended models. The resulting ontology can be considered to be validated with respect to the user's selections of unintended and omitted models.

One limitation with the Procedure is that we can only guarantee a near-miss theory as the output, since it can only add or remove axioms that already appear within some other theory in the same hierarchy. In general, we must be able to weaken an axiom rather than remove axioms (as illustrated by the example in Section 4).

A major challenge in the implementation of the validation procedures is the generation of models for the user to select. In the case of the BoxWorld Ontology, Mace4 was able to construct models of the CardWorld and BoxWorld axioms that correspond to the common shapes. Unfortunately, Mace4 has been inadequate for exploring the set of all models of the CardWorld and BoxWorld Ontologies, in the search for potentially unintended models. Even if we restrict our attention to small domains (e.g. a box with five surfaces, twelve edges, and eight vertices), Mace4 is unable to find models. We will be pursuing the use of other model constructors such as Paradox to see whether they are able to construct models on different domains.

References

[1] Megan Katsumi and Michael Gruninger. Theorem Proving in the Ontology Lifecycle. In *Knowledge Engineering and Ontology Design (KEOD 2010)*, pages 74–86, 2010.

[2] Nicola Guarino, Daniel Oberle, and Steffen Staab. *What is an Ontology?*, pages 1–17. Springer-Verlag, Berlin, 2 edition, 2009.

[3] Michael Grüninger. The ontological stance for a manufacturing scenario. *Journal of Cases in Information Systems*, 11:1–25, 2009.

[4] C. Chui and M. Gruninger. A molecular structure ontology for medicinal chemistry. In *Proc. of the 10th Int. Conference on Formal Ontologies in Information Systems (FOIS2016)*, pages 317–330. IOS Press, 2016.

[5] M. Gruninger and S. Bouafoud. Thinking outside (and inside) the box. In *Proceedings of SHAPES 1.0: The Shape of Things. Workshop at CONTEXT-11*, volume 812. CEUR-WS, 2011.

[6] Carmen Chui and Michael Grüninger. Mathematical Foundations for Participation Ontologies. In *Formal Ontology in Information Systems - Proceedings of the Eighth International Conference, FOIS 2014, September, 22-25, 2014, Rio de Janeiro, Brazil*, pages 105–118, 2014.

[7] C. Chui and M. Grüninger. Verification and modularization of the dolce upper ontology. In *FOUST: The Foundational Stance, Joint Ontology Workshop 2017, Bolzano, Italy*, 2017.

[8] Lydia Silva Muñoz and Michael Grüninger. Mapping and verification of the time ontology in SUMO. In *Formal Ontology in Information Systems - Proceedings of the 9th International Conference, FOIS 2016, Annecy, France, July 6-9, 2016*, pages 109–122, 2016.

[9] Ali B. Hashemi and Michael Grüninger. Ontology design through modular repositories. In *KEOD*, 2009.

[10] Michael Grüninger, Torsten Hahmann, Ali Hashemi, Darren Ong, and Atalay Özgövde. Modular first-order ontologies via repositories. *Applied Ontology*, 7(2):169–209, 2012.