

# Exploiting Modularity for Ontology Verification

Michael GRÜNINGER<sup>a</sup>, Torsten HAHMANN<sup>b</sup>, Megan KATSUMI<sup>a</sup>

<sup>a</sup> *Department of Mechanical and Industrial Engineering, University of Toronto,  
Toronto, Ontario, Canada M5S 3G8*

<sup>b</sup> *Department of Computer Science, University of Toronto, Toronto, Ontario,  
Canada M5S 3G8*

**Abstract.** Within knowledge representation, ontologies are logical theories that support software integration and decision support systems. Ontology verification is concerned with the relationship between the intended structures for an ontology and the models of the axiomatization of the ontology. To verify a particular ontology, we ideally characterize all the models of the ontology up to elementary equivalence and prove that these models are equivalent to the intended structures for the ontology. In this paper, we investigate the use of automated theorem provers and model finders to assist in the interactive verification of first-order ontologies. We identify the reasoning tasks that are associated with different aspects of ontology verification and discuss challenges for the application of automated reasoning systems to support these tasks.

**Keywords.** ontology repository, ontology evaluation, first-order logic, representation theorems

## 1. Introduction

An ontology is a logical theory that axiomatizes the concepts in some domain, which can either be commonsense knowledge representation (such as time, process, and shape) or the representation of knowledge in more technical domains (such as biology and engineering). In current ontology research, the languages for formal ontologies (such as RDFS, OWL, and Common Logic) are fragments of first-order logic, and many applications of ontologies, such as decision support and the semantic integration of software systems, rely on automated theorem proving or model generation. Within these applications, we need to make the claim that any inferences drawn by an automated reasoner using the ontology are actually entailed by the ontology's intended structures. 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. Ontology verification is concerned with proving that the intended structures for an ontology are equivalent to the models of the ontology's axiomatization. In Section 3 we show how this is typically done in the metatheory, using a specification of the models with respect to different classes of mathematical structures. As a

result of this characterization, we can replace a proof about the relationships between two classes of models with automated reasoning tasks that prove theorems about the relationship between two first-order theories.

Many ontologies that require verification are in an early development stage and thus lack a complete understanding of their models [6]. The relatively large size of ontologies (possibly containing many redundant axioms) when compared to traditional mathematical theories further complicates automated reasoning with ontologies. Traditional theorem provers are designed to reason with relatively small theories [7]. Proving particular properties in an ontology often requires only a small subset of its axioms, resulting in a great potential for inefficiencies. Since our theories are less structured than the knowledge bases examined in [1], we cannot easily use partition-based reasoning.

However, ontologies have an advantage that many mathematical theories do not have: they can be modularized. Modularization is a well-known technique to deal with large artefacts such as software or theories. For mathematical theories modularization may be unnecessary or even impossible because of the strong interaction amongst all axioms of a particular theory. But ontologies can often be modularized into sets of axioms that are closely related (coherent), e.g. by restricting a certain predicate or function. At the same time, modularization tries to minimize dependencies between modules (coupling).

In Section 4 we explore techniques for automated ontology verification in which the relationships between the modules within an ontology repository play a key role. Although several approaches [4] to ontology modularity rely on conservative extensions, we can also use a repository that consists of sets of modules that are ordered by entailment, allowing for nonconservative extensions [5]. When verifying a particular ontology we consider the modules that axiomatize subtheories of the ontology; the reasoning problems for verification of the ontology are solved by restricting them to the subtheories. In one approach, we search for the weakest modules in the repository that are required to find a proof; the hypothesis is that such modules do not contain axioms that are intuitively unnecessary for the proof from the original ontology. In another approach, a set of modules can be used to develop potentially useful lemmas, which can then be reused by any other modules that are extensions. Conversely, a set of lemmas can be used to characterize new modules which contain the minimal set of axioms required to prove the lemmas. A key objective of this paper is to present the pragmatic challenges for automated ontology verification and to show how modularity can, in principle, be leveraged to address those challenges. Despite their simplicity, the proposed verification techniques that exploit modularity are often quite effective. We hope that our work here promotes the development of more sophisticated modularity-based techniques to support ontology verification.

## 2. Modularity

Modularization of ontologies is still a young area of research looking for best practices on how to modularize an ontology and there are many – often conflicting – criteria for good modularizations. Our focus here is not so much on how to

modularize ontologies but how to use a given modularization to help with the task of ontology verification. Thus, we assume that a modularization into fairly coherent modules with little coupling between modules is already given.

A module consists of a set of axioms and a set of imports which in turn define a transitive import relation  $<$  – similar to modules in Common Logic [2]:

**Definition 1** A module  $M = (\mathcal{S}_M, \mathcal{I}_M)$  is a set of axioms  $\mathcal{S}_M$  together with a set of imported modules  $\mathcal{I}_M$ .

We say module  $M = (\mathcal{S}_M, \mathcal{I}_M)$  imports module  $N = (\mathcal{S}_N, \mathcal{I}_N)$  and write  $N < M$  if there is a chain of modules  $M_0, M_1, \dots, M_k$  with  $k \geq 1$ ,  $M = M_0$ , and  $N = M_k$  so that  $M_i \in \mathcal{I}_{M_{i-1}}$  for all  $1 \leq i \leq k$ .

If a module has an acyclic transitive import closure, it is a modular ontology:

**Definition 2** Let  $\mathbf{M}$  be the set of all modules reachable from a module  $M$ , that is,  $N \in \mathbf{M}$  iff  $N = M$  or  $N < M$ .

We call the structure  $(\mathbf{M}, <)$  a modular ontology iff  $\mathbf{M}$  is a finite set and  $<$  is irreflexive, i.e.,  $N \not< N$  for all  $N \in \mathbf{M}$ . We say the module  $M$  defines the ontology  $(\mathbf{M}, <)$ . The theory  $T_M = \bigcup_{N \in \mathbf{M}} \mathcal{S}_N$  axiomatizes  $(\mathbf{M}, <)$ .

Each module in a modular ontology defines itself a modular ontology. Theories of two modules in a modular ontology are related in the following way:

**Lemma 1** Let  $K, N \in \mathbf{M}$  be modules of a modular ontology  $(\mathbf{M}, <)$ .

If  $N < K$  then  $T_N \models \varphi \Rightarrow T_K \models \varphi$  for all sentences  $\varphi$ .

Two axiomatizations with equivalent languages and models are not distinguished but considered alternative axiomatizations of one module or ontology.

When designing ontologies, definitions are commonly used to capture intuitive concepts in the intended structures of the ontology. Theories of modules only containing definitions are known as definitional extensions; here, we do not treat them any different than theories defined by other modules.

Intuitively, modules closely resemble traditional mathematical theories; they are rather small, coherent axiomatizations of a single concept. While many ontology verification tasks might work reasonably well for individual modules, they often fail for a complete ontology. However, the modular structure can be exploited in various ways to facilitate the automation of ontology verification tasks. Though it is not guaranteed that the proposed procedures result in definite answers to specific verification tasks, they are at least able to confine a possible problem to a small set of modules requiring further manual inspection.

### 3. Reasoning Tasks for Ontology Verification

Our methodology for ontology verification revolves around the application of model-theoretic notions to the design and analysis of ontologies. The semantics of the ontology’s terminology can be characterized by a set of structures, which we refer to as the set of intended structures for the ontology. Intended structures are specified with respect to the models of well-understood mathematical theories

(such as partial orderings, lattices, incidence structures, geometries, and algebra). The extensions of the relations in an intended structure are then specified with respect to properties of these models.

Verification is concerned with the relationship between the intended structures for an ontology and the models of the ontology’s axiomatization. In particular, we want to characterize the models of an ontology up to isomorphism and determine whether or not these models are elementarily equivalent to the intended structures of the ontology. From a mathematical perspective this is formalized by the notion of representation theorems. Representation theorems are proven in two parts – we first prove every intended structure is a model of the ontology and then prove that every model of the ontology is elementary equivalent to some intended structure. Classes of structures for theories within an ontology are therefore axiomatized up to elementary equivalence – the theories are satisfied by any structure in the class, and any model of the theories is elementarily equivalent to a structure in the class.

This approach ontology verification rests upon the specification of two classes of structures – the intended structures and the models of the ontology’s axiomatization. The primary challenge for proving representation theorems is that it can be quite difficult to characterize the models of an ontology up to elementary equivalence. Ideally, since the classes of structures that are elementary equivalent to an ontology’s models often have their own axiomatizations, we should be able to reuse the characterizations of those other structures. Therefore, we replace a theorem about the relationships between two classes of models with automated reasoning tasks that focus on the relationship between two theories.

### 3.1. Representation Theorems

We now show how a theorem about the relationship between the class of the ontology’s models and the class of intended structures can be replaced by a theorem about the relationship between the ontology (a theory) and the theory axiomatizing the intended structures (assuming that such axiomatization is known). We use automated reasoners to prove the latter relationship and thus verify an ontology in a (semi-)automated way.

The mapping  $\pi$  is an interpretation of a theory  $T_A$  with language  $L_A$  into a theory  $T_B$  with language  $L_B$  if it preserves the theorems of  $T_A$ . We say that the two theories  $T_A$  and  $T_B$  are definably equivalent iff they are mutually interpretable, i.e.  $T_A$  is interpretable in  $T_B$  and vice versa. Translation definitions for the interpretation are sentences in the language  $L_A \cup L_B$  of the form

$$(\forall \bar{x}) p_i(\bar{x}) \equiv \varphi(\bar{x})$$

where  $p_i(\bar{x})$  is a relation symbol in  $L_A$  and  $\varphi(\bar{x})$  is a formula in  $L_B$  with no variables occurring free except for those in  $\bar{x}$ .

The key to using theorem proving and model finding to support ontology verification is the following theorem from [5]:

**Theorem 1** *A theory  $T$  is definably equivalent with a set of theories  $T_1, \dots, T_n$  iff the class of models  $Mod(T)$  can be represented by  $Mod(T_1) \cap \dots \cap Mod(T_n)$ .*

The necessary direction of a representation theorem (i.e. if a structure is intended, then it is a model of the ontology’s axiomatization) can be stated as

$$\mathcal{M} \in \mathfrak{M}^{intended} \Rightarrow \mathcal{M} \in Mod(T_{onto})$$

If we suppose that the theory that axiomatizes  $\mathfrak{M}^{intended}$  is the union of some previously known theories  $T_1, \dots, T_n$ , then by Theorem 1 we need to show that  $T_{onto}$  interprets  $T_1 \cup \dots \cup T_n$ . If  $\Delta$  is the set of translation definitions for this relative interpretation, then the necessary direction of the representation theorem is equivalent to the following reasoning task:

$$T_{onto} \cup \Delta \models T_1 \cup \dots \cup T_n \quad (\mathbf{Rep-1})$$

The sufficient direction of a representation theorem (any model of the ontology’s axiomatization is also an intended structure) can be stated as

$$\mathcal{M} \in Mod(T_{onto}) \Rightarrow \mathcal{M} \in \mathfrak{M}^{intended}$$

In this case, we need to show that  $T_1 \cup \dots \cup T_n$  interprets  $T_{onto}$ . If  $\Pi$  is the set of translation definitions for this relative interpretation, the sufficient direction of the representation theorem is equivalent to the following reasoning task:

$$T_1 \cup \dots \cup T_n \cup \Pi \models T_{onto} \quad (\mathbf{Rep-2})$$

By Theorem 1,  $Mod(T_{onto})$  is representable by  $\mathfrak{M}^{intended}$  iff  $T_1 \cup \dots \cup T_n$  is definably equivalent to  $T_{onto}$ , which we can show by proving both of the above reasoning tasks. All subsequently discussed reasoning tasks for ontology verification are related to the entailment problems **Rep-1** and **Rep-2**.

#### 4. Techniques for Automated Ontology Verification

Although we have specified a set of reasoning tasks that can be used to prove the representation theorems for an ontology, we still face challenges to automation arising from intractability and the semidecidability of first-order logic in combination with little confidence in the ontology’s axiomatization. If a theorem prover does not find a proof, does a proof exist or is the theory in fact too weak to prove the goal? If a model finder does not construct a model, does a model exist or is the theory in fact inconsistent? In the context of ontology verification, we are seeking techniques that can help us to solve the reasoning tasks **Rep-1** and **Rep-2** when the theorem prover fails to find a proof. In this section, we discuss various techniques that can be used to address these issues based on the notion of ontology tuning, which alters the axiomatization of the theory so that it remains logically unchanged but is successful in proving a certain sentence. The modularity of an ontology is key to these techniques. With the technique of theory weakening, we are searching for the weakest subtheory that is required to prove the goal. Another technique provides guidance – in particular by exploiting relative interpretations – for the generation and reuse of lemmas that can assist a theorem prover.

#### 4.1. Theory Weakening

If the theorem prover fails to find a proof, theory weakening can be used to remove axioms which are intuitively unnecessary for the proof and which might misguide the prover. Although this is a common approach in general, we can again use the modularity of the ontology to guide us in the identification of the appropriate set of axioms. Suppose  $(\mathbf{M}, <)$  is an ontology and that with the theory  $T_N$  we are not able to prove a sentence  $\varphi$ . We can hypothesize that the theory  $T_K$  defined by a module  $K < N$  might still be able to prove  $\varphi$ , so that we have  $T_N \models \varphi$  by the following relationship:

**Lemma 2** *Let  $(\mathbf{M}, <)$  be a modular ontology with modules  $K, N \in \mathbf{M}$  that  $K < N$ . If  $T_K \models \varphi$ , then  $T_N \models \varphi$ .*

The challenge, of course, is to select good candidate modules,  $K$ , whose theory is likely strong enough to prove the property but also small enough (length of the axiomatization) to prove the property automatically.

How to weaken the ontology's theory  $T_{onto}$  depends on the specific reasoning task; it can be based on the relationship between  $T_{onto}$  and  $T_1 \cup \dots \cup T_n$ . We can approach it as follows. For a specific partial axiomatization  $T_i$  of the intended structures, find the modules that axiomatize the subtheory  $S_i \subseteq T_{onto}$  such that

$$T_i \cup \Pi \models S_i$$

Each such  $S_i$  is equivalent to the axioms in  $T_{onto}$  that are interpretable by the theory  $T_i$ , so that we only need to use  $T_i$  rather than all of the theories  $T_1, \dots, T_n$  when attempting to prove sentences of  $T_{onto}$  whose translations are provable in a particular subtheory  $S_i$ .

We may further conjecture that  $T_i$  and  $S_i$  are definably equivalent, that is

$$S_i \cup \Delta_i \models T_i$$

If for some  $i$  a counterexample to this can be generated, we can find the subtheory  $T' \subset T_{onto}$  whose axioms are in the set difference of  $T_{onto}$  and  $\bigcup_i S_i$ . We can then finish the reducibility proof by showing

$$T_1 \cup \dots \cup T_n \models T'$$

The intuition here is that each of the theories  $T_i$  is definably equivalent to some subtheory of  $T_{onto}$ ; we only need to use all of the theories  $T_1, \dots, T_n$  when attempting to prove sentences of  $T_{onto}$  that are not in any of the associated subtheories.

If the subtheories  $S_i$  cannot be extracted from the modular structure, we can still prove lemmas using successively stronger subtheories of the ontology by starting with the atomic (weakest) modules and including more and more modules until a proof is found or until no more counterexamples can be generated.

Theory weakening equally applies to showing inconsistency of an ontology: it suffices to show inconsistency of some weaker theory. Likewise, goals interrelating several defined relations, such as a sentence stating that a set of relations is jointly exhaustive and pairwise disjoint (JEPD), can often be split into subgoals of which many may be provable in from a subset of the axioms and definitions.

#### 4.2. Lemma Generation and Reuse

The use of lemmas is a commonly accepted technique to improve theorem prover performance as it may reduce the number of steps required to obtain a proof. Previous work [6] has shown that lemmas can potentially assist with ontology verification. As discussed earlier, ontology verification requires finding proofs for both tasks **Rep-1** and **Rep-2**. We will now explore how these tasks can be leveraged to develop potentially useful lemmas.

Consider an ontology,  $T_{onto}$  that we are attempting to verify by proving that it is definably equivalent to a set of well-understood theories  $T_1, \dots, T_n$ . Suppose that some  $T_i \models \varphi$ , and that  $\varphi_{onto}$  is the sentence expressed in the language of  $T_{onto}$ , using the translation definitions. If  $T_{onto} \models \varphi_{onto}$  we can now store and use  $\varphi_{onto}$  as a potentially useful lemma.

With a similar intuition, we can consider weaker theories as sources for potential lemmas; any additional results of theories weaker than some  $T_i$  can be translated into lemmas expressed in the language of the ontology. More generally, if we can prove or already know that the ontology interprets some theory  $T_A$ , we can assist in proving that the ontology also interprets a theory  $T_B$  stronger than  $T_A$  by translating all axioms of  $T_A$  as well as any other results thereof into the language of the ontology to be used as lemmas. A similar approach of reusing lemmas has been implemented by the Interactive Mathematical Proof System (IMPS) [3]. For example, if we prove

$$T_{onto} \cup \Delta \models T_A$$

then the translations of the axioms and lemmas in  $T_A$  into the language  $L_{onto}$  of  $T_{onto}$  become lemmas to be used to show

$$T_{onto} \cup \Delta \models T_B$$

This lends itself to an approach that begins with showing that  $T_{onto}$  interprets a very weak theory and then retaining the translated axioms as lemmas for proving interpretations of successively stronger theories. First, proofs of basic properties of relations should be attempted. These include, e.g., reflexivity, symmetry, anti-symmetry, or transitivity for binary relations or symmetry or anti-symmetry between certain places of a relation, acyclicity, transitivity, or orderability for ternary relations.

Independent of the source of a lemma, for every lemma of an ontology there is some minimal module of the ontology whose theory proves the lemma:

**Lemma 3** *Let  $(\mathbf{M}, <)$  be a modular ontology. For any sentence  $T_{onto} \models \varphi$ , there is a weakest module  $W \in \mathbf{M}$  so that  $T_W \models \varphi$  and  $T_N \not\models \varphi$  for all  $N < W$ .*

The lemma  $\varphi$  then characterizes the module  $W$ . Storing lemmas with their weakest module allows us to reuse the lemmas every time a module is used. Though different axiomatizations of such a weakest module exists, they have equivalent theories and thus we do not distinguish them.

## 5. Summary

By specifying a set of theorem proving tasks that can be used to prove representation theorems for an ontology, we have shown how, in principle, to employ automated reasoners for rigorous ontology verification. We identified the pragmatic issues associated with this method of verification, and presented basic techniques geared towards making it work in practice. This leads to various open challenges for the further development of techniques to assist ontology verification by exploiting the modular structure of ontologies:

- How to design heuristics to be implemented in automated reasoners that maximally exploit the ontology tuning techniques and the modular structure of an ontology? How can ontology tuning be partly automated?
- When the intended structures are not axiomatized, how to select good candidate modules whose theory is likely strong enough to prove a property of interest but small enough (length of the axiomatization) to prove the property automatically?
- Can the set of lemmas entailed by a particular set of modules itself form a module? If so, could this be used as the basis for an approach to ontology modularization that is focused on entailed subtheories rather than subsets of axioms?

At the moment, ontology verification is primarily done manually – a user selects the ontology and tries to find a model or an inconsistency proof using a reasoner. This is extremely tedious for large ontologies; verification of first-order ontologies often requires skillful use of theorem provers with extensive of manual tweaking of the input and options (parameters). The long-term goal is to support interactive ontology verification with simple procedures that make extensive use of automated theorem provers and model finders. To this end, we conclude with a challenge to develop automated reasoning tools specifically for ontology verification, i.e. that account for the special properties of modularized ontologies and their reasoning tasks and techniques, as presented here.

## References

- [1] Amir, E., McIlraith, S.: Partition-based logical reasoning for first-order and propositional theories. *Artificial Intelligence* 162(1–2): 49–88 (2005)
- [2] Delugach, H. (ed.): *Common Logic – a framework for a family of logic-based languages*. ISO/IEC WD 24707 (Information Technology). [standards.iso.org/ittf/PubliclyAvailableStandards/c039175\\_ISO\\_IEC\\_24707\\_2007\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c039175_ISO_IEC_24707_2007(E).zip). (2007)
- [3] Farmer, W.M.: An infrastructure for intertheory reasoning. In: *Proc. of Conf. of Automated Deduction (CADE-17)*. pp. 115–131. LNCS 1831, Springer (2000)
- [4] B. Cuenca Grau, I. Horrocks, Y. Kazakov, U. Sattler: Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research* 31: 273–318 (2008)
- [5] Grüninger, M., Hahmann, T., Hashemi, A., Ong, D. Ontology verification with repositories. In: *Conf. on Formal Ontology in Inf. Systems (FOIS-10)*. pp. 317–33. IOS Press (2010)
- [6] Katsumi, M., Grüninger, M.: Theorem proving in the ontology lifecycle. In: *Conf. on Knowledge Engineering and Ontology Design (KEOD)*. (2010)
- [7] Pease, A., Sutcliffe, G.: First order reasoning on a large ontology. In: *CADE-21 WS on Empirically Successful Automated Reasoning on Large Theories (ESARLT)*. (2007)