INCONSISTENCY IDENTIFICATION IN DYNAMIC ONTOLOGIES BASED ON MODEL CHECKING

Mahdi Gueffaz, Perrine Pittet, Sylvain Rampacek, Christophe Cruz, Christophe Nicolle

INCONSISTENCY IDENTIFICATION IN DYNAMIC ONTOLOGIES BASED ON MODEL CHECKING

Mahdi Gueffaz, Perrine Pittet, Sylvain Rampacek, Christophe Cruz and Christophe Nicolle
LE2I Laboratory, UMR CNRS 5158, BP 47870, 21078 Dijon Cedex, France
{mahdi.gueffaz, perrine.pittet, sylvain.rampacek, christophe.cruz, cnicolle}@u-bourgogne.fr

Keywords: Model Checking, NuSMV Model Checker, Ontology Evolution, Ontology Design Pattern, Temporal Logic.

Abstract: The increasing use of ontologies and the cost of changes support the need to manage the evolution of ontologies. A common kind of error in ontology evolution is the logical contradiction declined as incoherences and inconsistences. In this paper, we propose a new approach to predict and identify the incoherences and inconsistences in the evolution of ontologies based on temporal logic and ontology design patterns. We implement the proposed approach using the NuSMV model checker. Based on these patterns, we propose an automated process to guide and monitor the implementation of change while ensuring the consistency of the evolved ontology.

1 INTRODUCTION

The semantic Web aims at organizing and structuring the huge quantity of information present on the Web. It consists of a semi-structured language based on XML (Bray and al, 2006). The W3C suggests the representation of the semantic Web in layers. Each of them is built upon the previous layers. Ontologies are the most important layer in the success of the semantic Web; they are often used in dynamic, multi-user and distributed environments.

The ontology construction goes through several stages. Among them, the evolution step change consists in turning the ontology more accurate and appropriate to the domain. Ontology evolution is a critical task because the new implementation can lead to the apparition of incoherences and inconsistences. Ontology inconsistency can occur for several reasons such as: modeling errors when correcting or adapting the ontology domain, conceptualization or specification. An incoherence corresponds to the existence of an unsatisfiable concept in the ontology intension. An inconsistence occurs when an individual exists for this unsatisfiable concept in the ontology extension. In the rest of the paper, we will use the term ontology inconsistency to define the set of incoherences and inconsistences occurring in the ontology. Ontology evolution corresponds to the application of a succession of change operations on the intension or the extension of the ontology.

Leading the implementation of the changes while maintaining the consistency of the ontology is a crucial task and a huge cost in terms of time and complexity. This task associated with ontology versioning purposes is called ontology change management. Furthermore, ontology change management, if led by a human editor, needs to be helped with an automatic or semi-automatic process. Actually, it is illusive to believe that a human could understand enough the entire conceptualization of the ontology to be able to predict all the consequences of the application of the changes and avoid inconsistences or incoherences. In the literature, one of the key objectives of the ontology change management is to bring an automated process to drive the application of a change while ensuring the consistency of the evolved ontology and its related versions.

In our contribution, we focus on incoherence issues during the change management of the ontology. We have defined a semi-automated methodology based on Model Checking (Baier and Katoen, 2008) helped by Ontology Design Patterns. The combination of the two techniques provides the inconsistent axiom succession patterns to retrieve in the ontology graph in temporal logics.

This paper is about ontology inconsistency identification in the evolution process. We first explain our use of the term ontology and formally define what we mean by ontology inconsistency, before introducing our methodology. We propose a

new way to predict and identify in the evolution of the ontologies by using the model checking technique. Model checking can handle complex problems with large amounts of information, stored as a graph, in order to verify critical systems. It will be a good opportunity to use the model checking on the ontology as it is represented by graphs. We use the NuSMV model checker (Cimatti and al, 2000) for this purpose.

The paper is organized as follows: the second section presents the work done in the detection and the resolution of ontology inconsistency. The third section presents the model checking technique and our contribution to identify the incoherences and inconsistences in the evolution of the ontology. The section four brings definitions on the ontology inconsistency in description logics. These definitions are used in section five, in which we apply our approach in order to remove the inconsistency on a formalized example. Finally, we end with a conclusion and future works.

2 RELATED WORKS

Several works have been proposed to maintain the evolution of ontologies coherent by trying to detect and delete the occurred contradictions. In the literature, maintaining consistency is based on the principles of resolution presented in (Haase and Stojanovic, 2005). When the axioms of change applied lead to an inconsistent ontology, the inconsistency is localized and the axiom having the lowest degree of confidence is identified and deleted.

The Pellet reasoner (Sirin, 2004) is the most used for the analysis and detection of inconsistency. Pellet is more or less accurate in its analysis based on the types of inconsistency and does not always give enough detail. Indeed, some logical inconsistency types, especially those relating to property, are not detected by Pellet. However, combined with the change patterns defined in (Djedidi, 2009), they are potentially supported. In order to bind this lack, in (Djedidi, 2009), the defined change management methodology called Onto-Evoal (Ontology Evolution and Evaluation) is based on modeling using these patterns.

In our proposal, we used a pattern-based methodology in a different way. We use a formal method, especially the model checking technique using temporal logics to handle the identification of logical contradictions from OWL DL logical constraints patterns. There are few works using temporal logics with ontology evolution. In (Plessers and De Troyer, 2006), temporal logic is used to represent ontology changes but the purpose is not the inconsistency detection.

3 MODEL CHECKING

In this section, we firstly present an overview of the model checking technique and the temporal logic. Secondly, we present our approach using this technique to identify ontological inconsistency.
Formal methods (Baier and Katoen, 2008) offer great potential for an early inclusion of verification in the design process, providing technical audit more efficiently, and reduce the verification time. Formal methods are highly recommended techniques for the software development. We use the method based on models that is the model checking method.

Model checking is a powerful tool for system verification, as it can reveal errors that were not discovered by other formal methods such as testing or simulation. It uses the temporal logic to describe the properties checking the system model. The concepts of temporal logic are used for the first time by Pnueli (Pnueli, 1977) in the specification of formal properties are fairly easy to use. The operators are very close in terms of natural language. The formalization in temporal logic is simple enough, although this apparent simplicity requires significant expertise. The temporal logic allows representing and reasoning about certain properties of the system, so it is well-suited for the system verification.

The model checking method examines all relevant system states in order to check whether they satisfy the desired property. The model checker gives a counter example that indicates how the model can violate the property. With the help of a simulator, the user can locate the error and adapt the model or the property to prevent the property violation.

4 INCONSISTENCY IDENTIFICATION APPROACH

This paper is about ontology inconsistency identification in the ontology evolution. More precisely, it is about the prediction and the identification of inconsistent change succession patterns in the evolution log into a system of ontology change management. Several studies have

shown the importance of the ontology evolution and the almost total missing of approaches to manage these changes.

The evolution consists in creating and managing different evolution of an ontology by treating the incompatibilities between the instances, the application and the ontologies that depend on them. To manage the evolution of ontologies, change management systems often generate an evolution log for each evolution (Djedidi, 2009) (Pittet and al, 2011) (Rogozan, 2008) (Jaziri, 2009). This log aims at tracking the changes made on the ontology. These changes made with ontology constructors can be additions or deletions of concepts, relations, properties or individuals.

To identify the ontology inconsistency our methodology has three phases. The first phase consists in transforming the evolution log into the NuSMV language (Gueffaz and al, 2011). The NuSMV graph is composed by nodes and arcs. The nodes represent the concepts and the arcs the properties of the ontology. The second phase consists in the generation of inconsistent axiom succession patterns. We use a subtype of ODP that we call change constraint pattern (CCP), to give the validity constraints corresponding to the change axioms. A first algorithm instantiates these constraint patterns with the elements of the NuSMV graph (concepts, properties, etc.). The second one transforms all these instantiated constraint patterns (ICP) into temporal logic formulas. Finally, the third phase uses the NuSMV model checker to check if one of these patterns can be found in the NuSMV graph using the temporal logic formulas. The NuSMV model checker steps chronologically through each node of the graph to find a node succession corresponding to one of the temporal logic inconsistency patterns. It is important to notice that nodes do not need to be direct successive neighbors to correspond to a pattern; they just need to appear chronologically in the same order.

Managing the effects of change implies not only the consistency identification but also its maintenance. The consistency maintenance consists in proposing and implementing a set of additional changes to resolve inconsistency. Once the incoherencies in the ontology evolution identified, there are many ways to resolve the inconsistency. However, the resolution phase of the inconsistency is beyond the scope of this paper and will be studied in other works.

Figure 1 describes the different steps to identify the inconsistency in the ontology evolution in our approach. The ontology user modifies it, and all the

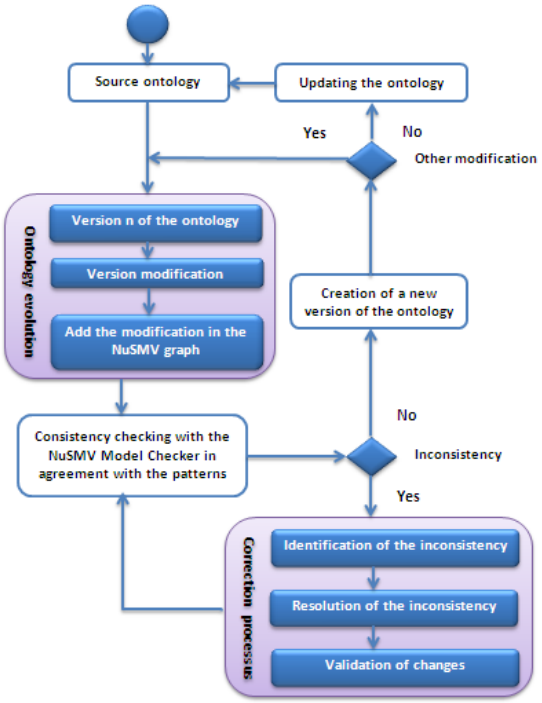

Figure 1: The inconsistency identification process.

modifications will be added in a graph represented by a NuSMV language. We transform our patterns into temporal logic formulas and give them the NuSMV model checker in order to identify the inconsistency in the NuSMV graph. If the model checker detects the presence of an inconsistent change succession corresponding to one of the temporal logic formulas, the system gives to the user the change succession which is in cause. The ontology is several times verified by the NuSMV model checker until there is no more inconsistence or incoherence. If there is no inconsistency in the ontology, the ontology developer creates and updates the source ontology using the new version of the ontology.

5 CONCLUSIONS & FUTURE WORKS

The paper presented a new methodology to identify inconsistency in the evolution of ontology combining the model checking technique and the ontology design patterns. We first introduced the work done in the detection and the resolution of ontology inconsistency. Several works identify and eliminate inconsistency but sometimes do not manage to detect all the inconsistences and

incoherences and the axioms in cause. Next, we describe the model checking technique and our global methodology. Our approach can predict all the potential logical inconsistency in the ontology before the addition of the incoherent change thanks to change constraints patterns derived from ontology design patterns. The inconsistent axiom succession patterns are then checked by the NuSMV model checker on the evolution log NuSMV graph, containing the whole change succession of the ontology. We also defined the ontology inconsistency in description logics, and we apply our approach on a simple example of incoherent ontology. This allowed us to identify the succession of axioms causing the inconsistency.

For future work, we are willing to apply our approach to both logical inconsistency and structural incoherency. We will also treat the inconsistency resolution based on this methodology in a next paper. In addition, we are looking forward to defining and integrating all the satisfiability constraints patterns of OWL DL in the implementation of our solution. Finally, we aim at implementing our solution on huge ontologies to measure the scalability and optimize our approach.

REFERENCES

Baier, C., Katoen, J. ”Principles of Model Checking,” The MIT Press, Cambridge, Massachusetts, London, England, 2008.

Bray, T., Paoli, J., Sperberg-McQueen., C., M., Maler, E., Yergeau, F., Cowan, J., 2006. Extensible Markup Language (XML) 1.1 (second edition) W3C recommendation, http://www.w3.org/TR/2006/REC-xml11-20060816/.

Cimatti, A., Clarke, E., Giunchiglia, F., Roveri, M., 2000. NuSMV: a new symbolic model checker.

Djedidi, R., 2009. Approche d'évolution d'ontologie guidée par des patrons de gestion de changement, PhD Thesis.

Gueffaz, M., Cruz, C., Nicolle, C., 2011. RDF2NµSMV: mapping semantic graphs to NµSMV model checker. The Third International Conference on Advances in Future Internet (AFIN 2011).

Haase, P., Stojanovic, L., 2005. Consistent Evolution of OWL Ontologies. In A.Gomez-Perez, J. Euzenat (Eds.), LNCS, vol.3532. The Semantic Web: Research and Applications (pp. 182-197). Berlin, Germany: Springer. doi: 10.1007/b136731

Jaziri, W., 2009. A methodology for ontology evolution and versioning., The Third International Conference on Advances in Semantic Processing(SEMAPRO 2009), pages 15-21, ISBN: 978-1-4244-5044-2, Sliema, Malta.

Pittet, P., Cruz, C., Nicolle, C., 2011. Guidelines for a Dynamic Ontology - Integrating Tools of Evolution and Versionning in Ontology. International Conference on Knowledge Management in Information Systems (KMIS 2011).

Plessers, P., & De Troyer, O., 2006. Resolving inconsistencies in evolving ontologies. In Y. Sure, & J. Domingue (Eds.), LNCS: Vol.4011. The Semantic Web: Research and Applications, Proceedings of the 3rd European Semantic Web Conference ESWC 2006 (pp. 200-214). Berlin, Germany: Springer.

Pnueli, A., 1977. The temporal logic of programs. In proc. 18th IEEE Symp. Foundations of Computer Science (FOCS’77), Providence, RI, USA. pages 46-57.

Rogozan, D., 2008. Gestion de l'évolution d'une ontologie: méthodes et outils pour un référencement sémantique évolutif fondé sur une analyse des changements entre versions de l'ontologie. PhD Thesis.

Sirin, E. et Parsia, B., 2004. Pellet: An owl dl reasoner. In Haarslev, V. et Möller, R. (editors), Proceedings of the International Workshop on Description Logics (DL2004).