Knowledge-Based Systems 73 (2015) 212–226

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Algebraic graph transformations for formalizing ontology changes and evolving ontologies

Mariem Mahfoudh *, Germain Forestier, Laurent Thiry, Michel Hassenforder

MIPS EA 2332, Université de Haute Alsace, 12 rue des Frères Lumière, 68093 Mulhouse, France

ARTICLE INFO

Article history:
Received 26 April 2014
Received in revised form 27 August 2014
Accepted 5 October 2014
Available online 14 October 2014

Keywords:
Ontology evolution
Typed Graph Grammars
Algebraic graph transformations
Consistency
AGG

ABSTRACT

An ontology represents a consensus on the representation of the concepts and axioms of a given domain. This consensus is often reached through an iterative process, each iteration consisting in modifying the current version of the consensus. Furthermore, frequent and continuous changes are also occurring when the represented domain evolves or when new requirements have to be considered. Consequently, ontologies have to be adaptable to handle evolution, revision and refinement. However, this process is highly challenging as it is often difficult to understand all affected ontology parts when changes are performed. Thus, inconsistencies can occur in the ontology as the changes can introduce contradictory axioms. To address this issue, this paper presents a formal approach for evolving ontologies using Typed Graph Grammars. This method relies on the algebraic approach Simple PushOut (SPO) of graph transformations. It formalizes the ontology changes and proposes an a priori approach of inconsistencies resolution. The modified ontology does not need an explicit checking as an incorrect ontology version cannot actually be generated. To validate our proposal, an implementation is presented using the Attributed Graph Grammar (AGG) toolbox.

1. Introduction

Formalizing knowledge has always presented an existential obsession and an important challenge for humans. The proposed solutions in the literature are mainly organized around databases, data warehouses and more recently ontologies. Ontologies are often defined as *an explicit specification of a conceptualization of a domain* [1]. They make possible for a community to reach a consensus and to bridge the gap of the vocabulary heterogeneity and semantic ambiguities. Thanks to their advantages, ontologies are used in a large range of fields such as: semantic web [2], business decision support [3], image interpretation [4], peer-to-peer networks [5], etc. A counterpart of this popularity, is the constant augmentation of available ontologies. For example, the number of ontologies on the BioPortal increased of 67% in 2013.[1] Furthermore, as building an ontology is an iterative process [6,7], the creation of a new ontology actually creates a set of several ontologies versions which is also consistently growing. For example, 51 versions of the Gene Ontology (one of the most successfully ontologies) are monthly released since January 2010.[2] Thus, more and more new ontologies are created and the number of versions of existing ontologies is constantly increasing.

Generate a new ontology version is however not a trivial task. It presents several challenges and requires a comprehensive study of the ontology model in order to manage its evolution. *Ontologies Evolution* is defined by Stojanovic et al. as *the timely adaptation of an ontology to the arisen changes and the consistent propagation of these changes to dependent artefacts* [8]. This process consists in the modification of one or many ontology components (class, property, axiom, individual, etc.) and it may be at instances level (*Ontology Population*) and/or structural level (*Ontology Enrichment*) [9]. Moreover, to preserve ontology consistency, the application of ontology changes must preserve all the ontology model constraints [8]. However, ontologies are often developed in a collaborative manner and are usually large and expressive. This makes difficult for a user and/or ontologist to understand all their affected parts (i.e. dependent entities) when changes are made. Therefore, to keep ontology consistency, it is important to have a mechanism that controls how the ontology changes are made and avoids the possible inconsistencies generated due to these changes.

* Corresponding author. Tel.: +33 (0)0389336960.
E-mail addresses: mariem.mahfoudh@uha.fr (M. Mahfoudh), germain.forestier@uha.fr (G. Forestier), laurent.thiry@uha.fr (L. Thiry), michel.hassenforder@uha.fr (M. Hassenforder).

[1] bioportal.bioontology.org/ontologies.

[2] geneontology.org/ontology-archive.

http://dx.doi.org/10.1016/j.knosys.2014.10.007
0950-7051/

The ontology languages such as Ontology Web Language (OWL[3]) are prevalent in knowledge representation, although, they are not sufficient for formalizing changes. They are indeed effective to capture static semantics but not changes that require a consistency checking of the interaction between ontologies entities. That is why, the proposed approaches in the literature do not addressed the inconsistencies issue [10] or used an*a posteriori*process to identify inconsistencies [11–13], etc. Thus, unlike previous approaches, this paper focuses on the critical issue of presenting a formal approach for consistent ontologies evolution by using Typed Graph Grammars and Algebraic Graph Transformations. Typed Graph Grammars (*TGG*) are a mathematical formalism that permits to represent and manage graphs. They are used in several fields of computer science such as software systems modeling, pattern recognition and formal language theory [14]. Recently, they started to be used in the ontology field, in particular for the modular ontologies formalization [15], Resource Description Framework graphs representation [16], collaborative ontologies evolution [17] and consistent ontologies evolution [18].

In our previous work [18], we have introduced the formalization of the ontology changes with Typed Graph Grammars and have focused on the atomic changes. A deeper study is presented in this paper which presents an exhaustive list of the atomic ontology changes and describes how consistently formalize the composite and complex changes. A comparison between the ontology changes representation in the OWL and our *TGG* formalism is presented to highlight the advantages of the use of graph grammars in the ontologies evolution process. Indeed, *TGG* and algebraic graph transformations provide a new way to formalize ontology changes and offer mechanisms to control graph transformations while avoiding the inconsistencies. Furthermore, they can reduce the number of elementary changes required to apply the composite and complex changes. The proposed approach has been implemented using a graph transformation tool Attributed Graph Grammar (AGG). In addition, we also present a mechanism to log the ontologies versions and ontology changes with a formal representation. An application is presented with the EventCCAlps ontology developed in the frame of the CCAlps European project.[4]

The rest of the paper is organized as follow: Section 2 presents related work and introduces Typed Graph Grammars and algebraic graph transformations. Section 3 proposes a graph transformation model for evolving ontologies and describes the formalization of ontology changes with Typed Graph Grammars. Section 4 presents an application using the EventCCAlps ontology. Section 5 evaluates and discusses the proposed approach. Finally, a conclusion summarizes the presented work and gives some perspectives.

2. Background and review

2.1. Related work

Managing ontologies evolution has been an important and active field of research in the recent years [9]. The approach of Stojanovic et al. [19] is considered as one of the first works that have addressed this issue. It presents a methodology in six phases: change capturing, change representation, semantics of change, change implementation, change propagation and change validation. The approach focuses on the KAON ontologies and identifies three types of ontology changes: (1) *atomic change* is an ontology change that affects a single ontology entity; (2) *composite change* is an ontology change that modifies the neighborhood of an ontology entity; (3) *complex change* is an ontology change that can be decomposed into elementary and composite ontology changes. Later, Klein et al. [11] have proposed another classification. They distinguish two types of ontology changes: elementary (atomic) and composite (complex). These changes can be specified via logging of incremental changes or by ontology versions comparison. The authors have also studied the problem of inconsistencies ontologies and proposed strategies resolution for each ontology changes. However, it is important to note that, the work is focused on the "ontology enrichment" and do not specify specific operations for the instances. Then, Luong et al. [20] have addressed both the "ontology enrichment" and the "ontology population". They have studied the evolution management for a corporate semantic web while addressing the RDF[5] (Resource Description Framework) ontologies. This choice restricts the expressivity of the methodology as the others ontology languages (such as OWL) require further types of changes (cardinality changes, restrictions on the classes, etc.). Thus, Djedidi et al. [12] have proposed an approach of OWL ontologies evolution based on pattern conception. They have studied both the atomic and composite changes and have used the Pellet reasoner [21] to detect the inconsistencies. A deeper study of the composite changes is introduced by Javed et al. [22]. It has presented resolution strategies for several composite changes and has described a layered change log for the explicit operational representation of ontology changes. The change log is formalized using a graph-based approach and implemented by OWLAPI.[6] To identify ontologies inconsistencies, Gueffaz et al. [23] have proposed CLOCk (Change Log Ontology Checker) approach which use model checking. A transformation of the OWL ontologies into a specific language NuSMV[7] is needed. However, no strategies are proposed to solve the inconsistencies. Recently, some researches are interested to look for new formalisms to represent ontologies and find others alternatives to the standard ontology languages. Then, Liu et al. [24] have introduced SetPi calculus [25] to model ontologies evolution process. They have represented ontologies by using SetPi entities and have defined a new formalism for describing the ontology changes. The work presents many ontology changes (basic and composite). However, it does not study the inconsistencies problem and do not proposes any implementation.

As a summary, various approaches have been proposed to define and implement ontology evolution process. The Table 1 presents a comparison of some approaches according to the languages used, the implementation, the inconsistency management and the specificities. Thus, we can see that different ontology languages have been studied: KAON [8], RDF [20], OWL [11,12,22], etc. Based on these languages, several ontology changes were defined and different classifications of theses changes were proposed [8,11]. Despite its importance, the problem of inconsistencies resolution is not sufficiently studied. Indeed, some works do not address this issue [10,24]. Others approaches are only focused on the inconsistencies identification [23]. Some researches are interested, in addition, to resolve the inconsistencies [12,20,22]. However, they use*a posteriori*process of inconsistencies resolution which require the implementation of changes and then, use external resources to check if the ontology consistency is affected or not. In our work, we propose an a priori approach to avoid inconsistencies by using Typed Graph Grammars formalism.

2.2. Typed Graph Grammars

This section reviews the fundamental notions involved in Typed Graph Grammars and algebraic graph transformations.

Definition 1 (*Graph*). A graph $G(V, E)$ is a structure composed by a set of vertices V, a set of edges E and an application $s : E \rightarrow V \times V$ that attaches a source/target vertex to each edge.

[3] w3.org/TR/owl-ref.
[4] ccalps.eu, project reference number: 15-3-1-IT.
[5] w3.org/RDF.
[6] owlapi.sourceforge.net.
[7] nusmv.fbk.eu.