Patterns for Semi-automatic Evolution and Refactoring of RDF Knowledge Bases

Christoph Rieß

c.riess.dev@googlemail.com

1 Introduction

With the emerging semantic web, RDF/OWL knowledge bases of all sizes came into existence and use. While applications are evolving, ontology concepts should change with them, but ontology refactoring was left behind for missing tool support or overcharging complexity of operations. To overcome this gap between the poorly structured data and the thoroughly engineered applications, a semi-automatic way of evolution on knowledge bases is presented here. Our approach will use evolution patterns and a processing system for them to restructure and assure data quality on modern semantic web data sources. Because of the nature of this problem not being completely new, there was knowledge from schema evolution on present-day relational database systems and software code refactorings in object-oriented programming used in background, as well as user's experience with RDF graph evolution at their domains. This work is structured mainly into 5 sections, where in Section 2 the conception and correspondence to underlying semantic web technologies is explained. Section 3 gives an overview over how patterns for our approach were acquired with methods of statistics and classified into three main classes. In the next section, Section 4, the integration of such a system with the OntoWiki framework, with a real-world example, is presented. A short overview of related concepts, not only in the field of ontology evolution, is given then (Section 5). Finally in the last Section 6 a conclusion and outlook to possible future work is given.

2 Concept and Technology

To ensure effective evolution of knowledge bases, patterns, like defined in this work, are trying to correspond as close as possible with the RDF data model. Patterns are executed by a pattern engine on an underlying triple store. Generally a pattern is basic (atomic) or compound, in the compound case, it consists of one or more (basic or compound) patterns. A base pattern is a triple with a set of variables V , a SPARQL query template S, and multiple update patterns U. The compound pattern is a sequence of n base or other compound patterns. An evolution process for a basic pattern is split into multiple steps. First there must be all variables bound, in second step the SPARQL query template is filled with bindings for V . It is executed and the result from the query is stored. In the third step changes get generated by evaluation of all update patterns U. The process for compound patterns isn't differing much from the one for base patterns, except that all sub patterns are executed in a well-defined sequence.

3 Core Pattern Library

A comprehensive library of patterns was created by analyzing related concepts in software refactoring and database system domain. There were also interviews with domain experts of in-use ontologies conducted to collect use cases for the evolution of ontologies, from which the most common patterns were extracted. A three level classification was developed, in regard to operations on the level of description logic as highest level, the level of instances for maintaining data quality, or even the lowest level of single entities, containing URI, language or data type's changes. A vocabulary for pattern exchange and increased reusability is also[1] provided.

4 Implementation

OntoWiki[2] is a collaborative semantic data wiki. Since version 0.9 it has a flexible extension system, which perfectly meets the requirements for front- and backend integration of the presented evolution pattern engine. As central element the pattern processing engine is developed as an OntoWiki component. This software offers, besides user interfaces for pattern execution, pattern management functions, too. It's also integrated with the backend, which means function support for SPARQL querying the triple store and adding or deleting statements when patterns get executed. Loading and storing patterns in the respective RDF Format is implemented, too.

5 Related Work

A very short overview for existing work is listed here. A quite similar composition and description of patterns is given in [DA10]. The classification in levels is an extended idea from [AP09]. In [FMK+08] the general classification for ontology operations is evaluated for different papers. The differences and extended requirements for ontology against database schema evolution are discussed in [NK04]. Some patterns were transferred from the book [Fow04] for source code refactorings.

[1] Pattern serialization scheme available under http://ns.ontowiki.net/SysOnt/EvolutionPattern/namespace
[2] Project page: http://code.google.com/p/ontowiki

6 Conclusion and Future Work

With the theory and practice a functional alternative for bulk editing multiple statements at once, converting ontology schemes, or maintaining data quality on RDF knowledge bases is given by our pattern system. The core pattern library has a starting set for most common use-cases and could be easily extended. Real-world implementation integrated with an existing collaborative semantic wiki enables the user to evolve their ontologies in a descriptive way, a lot like the querying of knowledge bases with SPARQL, but extended with various operations. Further investigations which cover the field of pattern management extensions and automatic recognition and execution have to be done.

References

[AP09] Muhammad Javed 0002, Yalemisew M. Abgaz, and Claus Pahl. A Pattern-Based Framework of Change Operators for Ontology Evolution. In Robert Meersman, Pilar Herrero, and Tharam S. Dillon, editors, OTM Workshops, volume 5872 of Lecture Notes in Computer Science, pages 544–553. Springer, 2009.

[DA10] Rim Djedidi and Marie-Aude Aufaure. NTO-EVO an Ontology Evolution Approach Guided by Pattern Modeling and Quality Evaluation. In Sebastian Link and Henri Prade, editors, FoIKS, volume 5956 of Lecture Notes in Computer Science, pages 286-305. Springer, 2010.

[FMK+08] Giorgos Flouris, Dimitris Manakanatas, Haridimos Kondylakis, Dimitris Plexousakis, and Grigoris Antoniou. Ontology change: classification and survey. Knowledge Eng. Review, 23(2):117–152, 2008.

[Fow04] Martin Fowler. Refactoring - improving the design of existing code. Addison-Wesley, 13. print. edition, 2004.

[NK04] Natalya Fridman Noy and Michel C. A. Klein. Ontology Evolution: Not the Same as Schema Evolution. Knowl. Inf. Syst., 6(4):428–440, 2004.