Dependency Updates and Reasoning in KiWi*

Jakub Kotowski[1], Stephanie Stroka[2], Francois Bry[1] and Sebastian Schaffert[2]

[1] {firstname.surname}@ifi.lmu.de
Institute for Informatics, University of Munich
[2] {firstname.surname}@salzburgresearch.at
Salzburg Research

Abstract. KiWi is a framework for semantic social software applications that combines the Wiki philosophy with Semantic Web technologies. Applications based on KiWi can therefore leverage i.a. reasoning and versioning to follow both aspects and even go beyond existing technologies. For example, KiWi allows composition of content items, which poses a challenge to the versioning system. In this paper we discuss versioning of composed content items and challenges related to reasoning in collaborative social software, as both topics are concerned with updates on the application state.

Versioning of composed content items. Versioning is an essential feature of Wikis. Currently, it is applied for a *transactional unit of work* in KiWi, which means that data is collected during one application transaction. Thus, one version affects only one content item[3]. Changes during one application transaction in textual/media content, or in meta-data (e.g. applied tags and RDF triples) are stored in one version.

Composed content items can either be created by copying or by referencing a content item into another one. If we apply a copy, the nested content item would be independent of the original document. A reference, however, could ensure that composed content items are kept consistent. This would lead to a reorganization of the versioning implementation. If one content item is referenced in another content item, both are placed in a new version if one of them changes. Hence, updates and versions have to be applied in a *user unit of work*, which means that not only updates during one application transaction are stored, but also referencing content items. This ensures the global consistency of dependent content and makes revisions more applicable for the user, who does not have to update each composed content item manually.

Furthermore, the system could inform the author about dependent updates. The user could decide, whether it is reasonable to keep the reference to composed content items or to change it to a copy, so that it gets decoupled from dependency updates.

* The research leading to these results is part of the project "KiWi - Knowledge in a Wiki" (http://www.kiwi-project.eu) and has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 211932.

[3] An entity that holds content and content-related data.

Reasoning in KiWi. Reasoning in the dynamic environments of social semantic software has its specificities such as need for user-defined rules, paraconsistency[4], need for simplicity and explanations[5]. Moreover, there are challenges due to the insufficient support of efficient and mature reasoning and triple store services by state-of-the-art software. KiWi aims to alleviate this problem by implementing its own rule-based reasoning over RDF data stored in a RDBMS. For simplicity and space reasons we limit ourselves only to forward-chaining implementation of the reasoner in this article.

RDF triple insert operation may take a long time if the inserted triple generates many derived triples based on the rule base. This may cause a long transaction blocking other operations. A question we would like to investigate is whether it is possible to split long transactions into several shorter ones. This may be possible due to the specific structure of RDF data and the nature of rule-based reasoning. Complementary approach to ensuring a good triple store throughput is to run the reasoning service in a separate thread which would process triple operations added to a queue. This way the system would not have to wait for the reasoning service to finish, but the user of the system may not always see a fully up to date state of the derived knowledge. We believe that this is a reasonable compromise for a responsive system.

The delete operation in current forward-chaining systems usually involves removing all derived data and recomputing the full closure from scratch. Alternative approaches have been suggested, e.g. [1]. Our intention is to further develop the reason-maintenance-based approach. It involves storing a kind of provenance information, called justification, with each derived triple. Justification consists of triples and the rule involved in the derivation of the triple. This allows faster closure recomputation by considering only the relevant dependency subtree using justifications. This approach also allows for explanation [2], may be necessary to provide change tracking [1] and may be equally important for the insert operation if the rule language includes negation as failure.

Conclusion. Reasoning and versioning in a Wiki-like environment pose interesting challenges that have yet to be tackled. In this short paper, we outlined the versioning of composed content items and our ideas for a sufficient reasoner.

References

1. J. Broekstra and A. Kampman. Inferencing and Truth Maintenance in RDF Schema - Exploring a naive practical approach. *Workshop on Practical and Scalable Semantic Systems (PSSS)*, 2003.
2. F. Bry and J. Kotowski. Towards Reasoning and Explanations for Social Tagging. *Proc. of ExaCt2008 - ECAI2008 Workshop on Explanation-aware Computing. Patras, Greece*, http://www.pms.ifi.lmu.de/publikationen#PMS-FB-2008-2, 2008.

[4] To support work in progress where inconsistencies easily arise.
[5] An approach to this aspect has been sketched in [2].