Bi-Directional Ontology Updates using Lenses

Andreas Textor
Distributed Systems Lab
RheinMain University of Applied Sciences
Unter den Eichen 5, D-65195 Wiesbaden, Germany
andreas.textor@hs-rm.de

***Abstract*—Ontologies can be used to unambiguously describe the semantics of the entities of a domain. Furthermore, ontologies can also contain instances that represent states of real world systems. When an ontology is dynamically updated to reflect changes in the real world, or vice versa (reaction to new information added by a reasoner), data needs to be mapped in both directions. In many systems, this happens through an ad-hoc implementation. Maintaining translations in both directions can be complex and time-consuming. Also, it is often difficult to split a mapping into reusable components. In this paper we examine how *lenses*, an approach to the view update problem originating from database research, can be applied to value updates in ontologies. Lenses provide bi-directional composable translations from one model to another. The application of the approach in the domain of IT management, where an ontology is constantly updated with values from managed systems, is described as part of the ongoing project.**

Keywords-ontology*; *ontology update*; *ontology mapping*; *lenses*; *view update

I. Introduction

With the ever growing amount of data in all areas of computing and Information Technology, effective means for managing information become more and more important. Especially when data exists in many different heterogenous sources and formats, integration of information and interoperability of applications that process the data are essential. Furthermore, as syntactic translation of data between different sources is often not sufficient, ontologies are increasingly used to capture semantic information. However, using ontologies comes with its own range of problems that need to be solved, in particular when a single ontology is not sufficient. When multiple (sub-) ontologies, possibly from different sources or authors, are used, they need to be integrated. This leads to research questions such as ontology merging and mapping, matching and ontology alignment, distributed querying and distributed reasoning and others. When information sources and formats external to the ontology need to be dynamically connected to the ontology (i.e., values and/or model structures need to be synchronized), this often results in large amounts of boilerplate code, which is hard to maintain and poorly reusable.

Both cases, regular ontology alignment and the alignment of an ontology with other external models are comparable problems. In ontology alignment, relations between vocabularies of different ontologies are established, while in the alignment with external models relations between concepts in the ontology and concepts in the external model are defined. Depending on the type of model, such relations between concepts can be of the types one-to-one, one-to-many or many-to-one. When the ontology is not only used as a passive information store, but is dynamically updated with information from the external data source, and vice versa (i.e., new facts found by a reasoner are pushed to the external system), information needs to flow in both directions. Translations of data formats and structures need to be performed each time data flows corresponding to the mapping of the external format to the ontology. If we assume the ontology to be a domain model that formally captures the domain and uses this semantic basis to connect other ontologies to it, possibly from different domains, it creates a comprehensive information base. Updating an external system using data from this compound ontology can pose a loss of information, as it only captures a part of the ontology (e.g., an IT management system probably does not include accounting information). On the other hand, importing data from the external system into the ontology may require incomplete data to be complemented to "fit" the data model of the ontology.

This problem is known in database research as the View Update Problem [1]. To approach the problem in the context of ontologies, we examine how *lenses*, a structure for bi-directional composable translations can be adapted to ontologies, with a focus on modularity. Lenses are well examined for the application in database systems, but to be able to be used with ontologies, different requirements must be taken into account. Therefore, we first explain the idea of lenses and then the basic application of lenses to ontologies.

This paper is structured as follows: Section II briefly explains the concept of lenses, as it is defined for the database context. Section III examines existing work in the areas of ontology update, view update and lenses. In Section IV, the approach for the application of lenses in the context of ontology values is described. Section V describes the work in progress, where the approach is applied in the domain of IT management. The paper closes with a summary and future work in Section VI.

 ISBN: 978-1-61208-240-0

II. EXPLANATION OF LENSES

Lenses were first proposed by Foster et al. in [1] to address the View Update Problem - how can changes made to views be propagated back into the underlying tables. The authors show that the abstract concept of lenses is not only applicable to database schemas, but to other data models as well, and give concrete lenses for the transformation of trees. The concept was further examined in the context of relational databases by Bohannon et al. in [2]. How lenses can be implemented and more use cases are given in [3].

The definition for lenses given in [2] is as follows:

Definition [Lenses]: Given schemas Σ and Δ, a *lens* v from Σ to Δ (written $v \in \Sigma \leftrightarrow \Delta$) is a pair of total functions $v \nearrow \in \Sigma \to \Delta$ ("$v \nearrow$" is pronounced "v get") and $v \searrow \in \Delta \times \Sigma \to \Sigma$ (pronounced "v putback").

Intuitively, a lens combines the pair of functions `get` and `putback`, as shown in the visual explanation in Figure 1, which is derived from [4]. The `get` and `putback` functions define the mapping between the original data source (e.g., the database tables) and the external model (e.g., database views). Together, they provide a different view onto the data, hence the name *lens*.

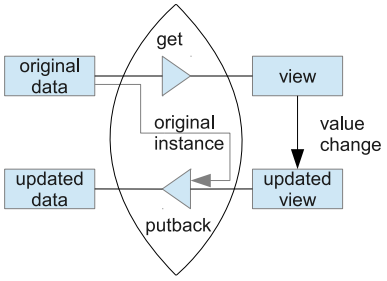

Figure 1. Visual explanation of a lens

The `putback` function is intended to be the inverse of the `get` function in a sense that the resulting lens is *reasonable*, i.e., that the `putback` function only revises the model structures and instances that are necessary for the change. For this reason the function not only depends on the updated structure from the external model, but also on the original structures and instances the change refers to.

To specify this requirement, the authors in [2] define so-called *well-behaved* lenses that must satisfy certain laws:

Definition [Well-behaved lenses]: Given schemas Σ and Δ along with a lens $v \in \Sigma \leftrightarrow \Delta$, we say that v is a *well-behaved* lens from Σ to Δ (written $v \in \Sigma \Leftrightarrow \Delta$) if it satisfies the laws GETPUT and PUTGET:

$$v \searrow (v \nearrow (I), I) = I \quad \text{for all } I \in \Sigma \qquad \text{(GETPUT)}$$
$$v \nearrow (v \searrow (J, I)) = J \quad \text{for all } (J, I) \in \Delta \times \Sigma \qquad \text{(PUTGET)}$$

The GETPUT law, which is also called *Stability* in [4], states that the original model should not be changed, if the external model is not changed. This means that the `putback` function should not touch (e.g., set to zero) fields in the original model, if they were not touched in the update operation. The PUTGET law (also called *Acceptability* in [4]) states that updates to the original model should be performed so that the next call of `get` yields exactly the previously put information. This law could be violated, if the `putback` function would write a value other than the one that was updated in the external model (e.g., if `putback` always writes a constant value). The result of a subsequent call of `get` would then be different than the updated value.

The lens laws assure one important property: The composability of lenses, i.e., the creation of new lenses through the composition of existing lenses, similar to function composition. This property allows the creation of separate lenses for each structural or data translation, which are then composed together to form the original specified mapping. When well-behaved lenses are chained together, Foster et al. [1] show that the resulting lens satisfies the lens laws as well.

III. RELATED WORK

The approach presented here cuts different areas of research: ontology-based information integration, the View Update Problem, ontology updates and ontology mapping. Firstly, publications in which ontologies are employed to achieve information integration range over various domains, and usually describe a mapping of external data formats to the ontology. Representative for the problem at hand is [5], which describes an architecture where an ontology is used for mashups of streaming and stored data. They feature a semantic integration service that allows queries over independent heterogenous data sources. This is implemented by providing individual mappings for each data source to a central ontology. However, it only works in one direction, as they do not specify how data is propagated back.

Updating ontologies still poses different questions than updating tables in a Relational Database Management System (RDBMS), because updated knowledge may not contradict existing knowledge (which was possibly deduced by a reasoner, rather than added manually), because it would render the ontology inconsistent. Belief update, and more specifically, ontology update, has been examined in several publications. For example, in [6], Lösch et al. propose an ontology update framework where ontology update specifications, which are similar to database triggers, describe certain change patterns that can be performed. Only when an update specification accounts for a change request, the request is accepted, otherwise it is denied. Most of the work on ontology updates is focused on changing the ontology structure, which poses a different problem than updating ontology values and is therefore not directly comparable to our approach. Ontology mapping has been discussed in many publications. Shvaiko and Euzenat [7] give a comprehensive overview of different ontology mapping approaches.

 ISBN: 978-1-61208-240-0

Scharffe and De Bruijn [8] propose requirements for a language to specify ontology mappings (which can also be bi-directional), while in [9], Belhadef gives a method for bi-directional ontology matching that relies on terminological, syntactical and structural comparisons. Again, this focuses on the ontology structure and is not directly comparable to our approach.

IV. Approach

In this section, we want to examine how the abstract concept of lenses can be applied in scenarios, where external data sources need to be synchronized with an ontology. The approach is orthogonal to existing works, as the goal is not to develop mappings, but an abstraction that allows mappings to be composed out of reusable smaller parts. Regardless of the actual domain, data model or mapping specification, this synchronization is usually implemented in a way that performs structural and value translations, according to the external model. For example, when data from an existing address book should be synchronized with an ontology that also contains other personal data, the ontology might have object properties and data properties that do not directly map to fields in the address book, and values with types such as date time, which might need to be converted from an internal representation to `xsd:dateTime` format, or strings, which might need an encoding conversion. Thus, with each conversion step between the external representation and the ontology, several sub-steps might be necessesary. Instead of ad-hoc handling each sub-step in the data conversion implementation, the mapping should be modularized so that each sub-step is a separate entity, and one conversion step is just a composition of the individual sub-steps.

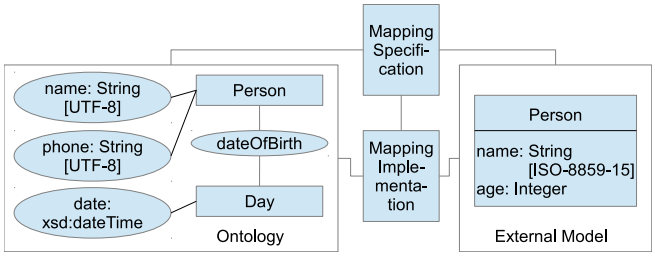

Figure 2. Mapping external data sources to ontologies

If we consider the example given in Figure 2, we can easily see what the mapping specification should look like: The Person class can be mapped to an `owl:Class`, string encodings must be translated, and the `age` property of the class should be converted to the right date format using a reference date. However, two problems arise, when the mapping implementation is straightforwardly derived or generated from the specification. First of all, if the mapping is implemented in a monolithic fashion, i.e., without further modularization into the sub-steps, the single conversion sub-steps (i.e., conversion of structure, data types, data values) are neither reusable nor easily maintainable. Secondly, the specification and the implementation need to take of cases, when data is converted bi-directionally. When a Person instance record from the ontology is extracted and converted to an instance of the external Person type, the `phone`attribute is simply omitted. When the instance is then updated in the external model (i.e., the name is changed), and the corresponding ontology instance should be updated accordingly, it is desirable that the `phone` attribute from the original ontology Person instance remains unchanged. The mapping implementation therefore needs to consider existing Person instances in the ontology that represent the same external instance as well as newly inserted instances. Both problems, modularization of bi-directional translations, and the accounting for difference in structure and merging of existing and new fields, can be solved with the application of lenses. In this context, the ontology represents the original source of data, as it is intended to comprehensively aggregate the existing domain knowledge, while the external model can be compared to a database view, as it only covers a subset of the available information (hiding information is often the reason to define a view, while the external model only contains the data structures that are essential for the external system). In the definition of lenses, the GetPut and PutGet laws are necessary for the translation of data in both directions, but as the laws from the original lenses definition originate from database schemas rather than ontologies, the laws are not sufficient to guarantee the consistency of the ontology, because updating certain facts can lead to contradictions with existing facts. The process of changing beliefs to take into account a new piece of information about the world is called *belief change*. This problem has been extensively studied and most formal studies on belief change are based on the work of Alchourrón, Gärdenfors and Makinson (see, e.g., [10]). They specify postulates for contraction (i.e., removal of beliefs from a knowledge base) and revision (changing or updating beliefs in a knowledge base) operators, that must be satisfied by all rational belief change operators. Although belief change theory is not directly applicable to ontologies, Ribeiro and Wassermann [11] have shown that the theory can be applied to ontologies when certain postulates are adapted accordingly. The PutGet law can be related to the *Closure*, *Success* and *Expansion* postulates. The *Closure* postulate ($K*\alpha = Cn(K*\alpha)$, where K is the knowledgebase, α is the fact to be revised, $(*)$ is the belief revision operator and Cn is the closure function) states that the knowledge base should be logically closed after the new fact is added, the *Success* postulate ($\alpha \in K*\alpha$) states that new information should be successfully accepted, and the *Expansion* postulate ($K*\alpha \subseteq K+\alpha$) states that the revised knowledgebase should not contain more facts than the result of K expanded by α (i.e., the fact added without consideration of consistency). The *Consistency* ($K*\alpha$ is inconsistent, only if $\vdash \neg\alpha$), *Preservation* (If $\neg\alpha \notin K$ then $K+\alpha \subseteq K*\alpha$) and

Extensionality (If $\alpha \equiv \beta$ then $K * \alpha = K * \beta$) postulates do not apply to relational databases and are for this reason not reflected by the lens laws. In order to maintain consistency in the ontology when applying updates through lenses, the lens, which can be considered a revision operator, must therefore be implemented to satisfy the remaining postulates as well (if no other measures for maintaining consistency are taken).

As the composability of lenses makes it possible to create a library of lenses for common or very specific updates to ontologies, the revision postulates should be considered when lenses for ontology updates are created.

V. Application

The concept of lenses is currently being used in the implementation of an ontology-based automated IT management system. We are working on the implementation of a concrete set of lenses for the translation between an OWL (Web Ontology Language) ontology representation and the external data source of a CIM environment. The Common Information Model (CIM, [12]) is an object-oriented model to represent entities and relationships of IT systems, and is used in IT management and storage management tools. A translation of CIM to OWL was previously examined in [13], and used in an architecture for automated IT management [14]. Preliminary results show that the application of lenses to implement the mapping between the ontology and the CIM environment, rather than the previously used prototypical monolithic implementation, can greatly contribute to the modularity and extensibility of the architecture.

VI. Conclusion and Future Work

In this paper, we have examined the abstract concept of lenses, an approach to the view update problem in databases, and its applicability to the context of ontologies. We have shown that for scenarios where an ontology serves as aggregation of domain knowledge that is dynamically updated with an external model, the ontology can be thought of as the original data source, while the external model can be thought of as a view. This allows the use of lenses for synchronization between the models. As ontology updates differ from updates of relational databases, we have examined how the lens laws relate to the postulates that belief revision operators must satisfy, and found that a lens that performs ontology updates can not solely rely on the lens laws, but must still follow the postulates. Future work therefore includes the completion of the formalisation of belief revision for ontology update lenses and the further evaluation of the approach in the domain of IT management.

References

[1] J. N. Foster, M. B. Greenwald, J. T. Moore, B. C. Pierce, and A. Schmitt, "Combinators for bi-directional tree transformations: A linguistic approach to the view update problem," in *In ACM SIGPLANSIGACT Symposium on Principles of Programming Languages (POPL)*. ACM Press, 2005, pp. 233–246.

[2] A. Bohannon, B. C. Pierce, and J. A. Vaughan, "Relational Lenses : A Language for Updatable Views," in *Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2006.

[3] A. Wider, "Towards Combinators for Bidirectional Model Transformations in Scala," in *Post-Proceedings of the 4th International Conference on Software Language Engineering (SLE'11)*. Braga, Portugal: Springer, 2011, pp. 1–10.

[4] B. C. Pierce, "The Weird World of Bi-Directional Programming," March 2006, ETAPS invited talk.

[5] A. J. G. Gray, R. García-Castro, K. Kyzirakos, M. Karpathiotakis, J.-p. Calbimonte, K. Page *et al.*, "A Semantically Enabled Service Architecture for Mashups over Streaming and Stored Data," in *Proceedings of the 8th Extended Semantic Web Conference*, 2011.

[6] U. Lösch, S. Rudolph, D. Vrandečić, and R. Studer, "Tempus fugit - Towards an Ontology Update Language," in *6th European Semantic Web Conference (ESWC 09)*, vol. 1. Springer, January 2009, pp. 278–292.

[7] P. Shvaiko and J. Euzenat, "Ontology matching: state of the art and future challenges," *IEEE Transactions on Knowledge and Data Engineering*, 2012.

[8] F. Scharffe and J. de Bruijn, "A Language to Specify Mappings Between Ontologies," in *Proceedings of the 1st International Conference on Signal-Image Technology and Internet-Based Systems (SITIS 2005)*, R. Chbeir, A. Dipanda, and K. Yétongnon, Eds., Yaounde, Cameroon, 2005, pp. 267–271.

[9] H. Belhadef, "A New Bidirectional Method for Ontologies Matching," *Procedia Engineering*, vol. 23, pp. 558–564, Jan. 2011.

[10] C. E. Alchourrón, P. Gärdenfors, and D. Makinson, "On the logic of theory change: Partial meet contraction and revision functions," *The Journal of Symbolic Logic*, vol. 50, pp. 510 – 530, 1985.

[11] M. M. Ribeiro and R. Wassermann, "First Steps Towards Revising Ontologies," in *Proc. of WONRO'2006*, 2006.

[12] Distributed Management Task Force, "Common Information Model (CIM)," http://www.dmtf.org/standards/cim/. Last access 2012-07-09.

[13] A. Textor, J. Stynes, and R. Kroeger, "Transformation of the Common Information Model to OWL," in *10th International Conference on Web Engineering - ICWE 2010 Workshops*, ser. LNCS, vol. 6385. Springer Verlag, July 2010, pp. 163–174.

[14] A. Textor, F. Meyer, and R. Kroeger, "Semantic Processing in IT Management," in *Proceedings of the Fifth International Conference on Advances in Semantic Processing (SEMAPRO)*, 2011, pp. 87–90.

 ISBN: 978-1-61208-240-0