Lenses for View Synchronization in Metamodel-Based Multi-View Modeling

Arif Wider

Humboldt-Universität zu Berlin
Unter den Linden 6, D-10099 Berlin, Germany
`wider@informatik.hu-berlin.de`

Abstract. When using multiple views to describe a system, the underlying models of these views have to be kept consistent, which is called *model synchronization.* Manually implemented model synchronizations that are not simple bijections are hard to maintain and to reason about. Special languages for expressing *bidirectional transformations* can help in this respect, but existing languages applicable in *model-driven engineering* are often restricted to bijections or complex to use. I adapt *lenses*, a promising term-rewriting-based approach to bidirectional transformations, to model synchronization. This allows for flexible view synchronization that can be integrated with existing metamodel-based technologies.

Introduction

Modeling a system using multiple views is a common means nowadays to break down the complexity of the system description. A prominent example for that is the architecture of the UML, providing multiple diagram types which serve as aspect-specific views on a system. Using multiple views to describe a system imposes the problem of *inter-view consistency.*

In *model-driven engineering* (MDE) different views on a system can be implemented as different *domain-specific languages* (DSLs). This way, the system description consists of an ensemble of models described using these languages. This is called *multi-view modeling* or *domain-specific multimodeling* [11, 10]. In multimodeling inter-view consistency is achieved by synchronizing these models.

Naively implemented model synchronizations, i.e., pairs of forward and backward transformations described in a general-purpose language, can be hard to maintain and to reason about because consistency of forward and backward transformations has to be ensured and transformations can be arbitrarily complex. Special languages for describing *bidirectional transformations* provide notations to describe consistency relations between models. From this notation a forward and a backward transformation can be automatically inferred so that the consistency of these transformations is ensured by construction. This is easy if the relation is a bijection but gets hard if it is neither surjective nor injective. Unfortunately, as the idea of a view is to hide information which is not aspect-specific, bijections hardly occur in a multi-view setting.

Bidirectional transformations are researched for a long time, e.g, in the graph transformation community using *Triple Graph Grammars* (TGGs) [18, 6]. With

QVT Relational[1], there is even a standard by the OMG for describing bidirectional transformations of metamodel-based models. Nevertheless, languages for describing bidirectional model transformations are still not widely used in MDE. Concerning QVT, Stevens points out semantic issues that could be one reason for this limited acceptance [19]. Another practical issue could be the weak situation regarding tool support for QVT Relations: Although QVT specification was completed in 2008, there are only few implementations and even those are not maintained regularly.

Lenses [8, 7] is a *combinator-based* approach to bidirectional transformations: Foster et al. provide small, well-unterstood bidirectional transformations (called lenses) and a set of combinators that allow more complex transformations to be composed from those smaller ones. This greatly improves extensibility and comprehensibility. Furthermore, a type system guarantees that composed lenses preserve certain properties of their sub-lenses. This combinator-based approach is possible because lenses are restricted to the asymmetric case where one of the two models to be synchronized is an abstraction of the other, i.e., the relation is at least surjective. This way, the problem of model synchronization resembles the *view update problem* that has been studied in the database community for decades [4].

In contrast to less restricted symmetric approaches like TGGs and QVT that were designed for transformations of graphs and models, respectively, lenses were designed for synchronization of tree-like data and were mainly implemented for string transformations [3], e.g., synchronization of XML-data. This poses some conceptual challenges, when attempting to use lenses for model synchronization in a metamodel-based setting.

Related Work

There are several approaches using bidirectional transformations in the context of MDE, but most of them use symmetric bidirectional transformations and therefore lack the combinator-based nature that can be achieved using an asymmetric approach (e.g., the *AToM3* Framework [1]). Among them are also some, that integrate with existing metamodel-based technologies, e.g., the *Tefkat* transformation engine [16] that is closely connected with QVT and integrates with the *Eclipse Modeling Framework* (EMF)[2]. Recently, Hettel et al. presented an asymmetric approach for using the SQL-like Tefkat language for round-trip engineering [12].

An approach quite similar to lenses which is also used for view synchronization is the work of Hu et al. [14, 17]. Somehow similar to my approach, Garcia proposed to use their work in a metamodel-based context [9]. However, in their approach, changes in a model have to be explicitly marked to be synchronized. This prevents agnostic integration with existing metamodel-based technologies.

[1] http://www.omg.org/spec/QVT/1.0/
[2] http://www.eclipse.org/modeling/emf/

Furthermore, there are some approaches to bidirectional transformations that are heavily inspired by lenses or extend lenses, but are not used for view synchronization: Hidaka combines lenses with a query language but not in a metamodel-based context [13].

Probably closest to my work is the work of Xiong [22, 21, 5], who integrated concepts of lenses into his work on bidirectional transformations of metamodel-based models, but he proposes an update-based approach in contrast to the state-based approach of lenses and his work does not focus on integration with existing metamodel-based technologies.

Approach

My approach is to use lenses for view synchronization in metamodel-based multi-view modeling environments. In order to achieve this, I want to show that

1. the advantages of lenses, especially their composability, can be leveraged when describing bidirectional transformations of metamodel-based models,
2. that in conjunction with a synchronization architecture that incorporates a common model, lenses can be beneficially used for view synchronization and
3. that this approach allows for straightforward integration with existing metamodel-based technologies.

Lenses for Bidirectional Model Transformations In order to use lenses for model synchronization, the concepts of lenses have to be bridged from the grammarware technological space that lenses originate from to the modelware technological space [20]. For this, the following challenges are to be solved:

– **Typing:** In the original lens framework typing is mainly used for ensuring that certain lens properties are preserved when composing lenses. In MDE, transformations usually transform models conforming to one metamodel so that they conform to another metamodel. Therefore, typing of input and output data of lenses is highly desirable for model transformations.
– **Ordered data:** Originally, lenses work either on unordered tree-like data or certain keys have to be defined to be able to synchronize changes regarding order. With metamodel-based models, i.e., in an object-oriented setting, there is the object identity as an implicit key. This can be used to propagate changes in order and other complicated changes back to the original model.
– **References:** Models in general are graphs because they can contain references, whereas lenses were designed for synchronizing tree-like data. A pragmatic solution could be to make use of the containment hierarchy that is provided by many metamodeling frameworks anyway.

A Lens-Based Model Synchronization Architecture While the restriction to asymmetric synchronization does not seem to be flexible enough for the general MDE setting, it fits well to view synchronization: Lenses can be used to asymmetrically synchronize view-models with a common model. This common model can be a shared abstraction, i.e., it only contains those information that is represented in more than one view. In the database community, this approach to view synchronization was already proposed by Atzeni & Torlone in 1996 [2]. Another approach is to synchronize views with a shared complete model of the system containing the information of all models to be synchronized. In both cases the changes made in one view-model are propagated to the other view-models through the common model.

Technological Integration As stated before, it is my goal to provide a solution that can be integrated with existing metamodel-based technologies, in particular, with the *Eclipse Modeling Framework* (EMF). As EMF is a Java-based framework, I decided to implement lenses for model transformations as an internal DSL in the *Scala*[3] programming language. Scala code compiles to JVM bytecode and Scala provides great interoperability with Java-based frameworks. Moreover, Scala combines functional and object-oriented concepts, which fits to the task of adapting lenses that come from a functional background to be used in a metamodel-based, i.e., object-oriented setting. Finally, Scala has static typing and it is my goal to achieve compile-time type checking for transformations in as many situations as possible. Therefore, I make use of *heterogeneously typed lists* [15], that were originally developed for the Haskell programming language. My solution will be deliverable as a Scala library. As a consequence, no further tools than the Scala compiler and a Scala IDE plug-in should be needed to integrate the solution into existing projects and tool chains. Hopefully, this results in higher user acceptance compared to solutions like QVT that always depend on up-to-date tool support.

Evaluation and Expected Contributions

The evaluation of my approach is tightly coupled with the ongoing development of a *domain-specific workbench* for the development of optical nanostructures, which is subject of a cooperation with a group of physicists. This workbench provides different DSLs for describing different aspects of experiments in nanostructure development and is being implemented with EMF-based technologies. This project serves as a comprehensive case study for my solution. In particular, it has to be evaluated if a reasonably sized set of basic lenses and lens combinators can be provided that enable to accomplish common view synchronization tasks in a concise way.

As a result of my work, the following contributions can be expected:

- An extended formal lens framework, adapted for an object-oriented setting.
- A language for bidirectional model transformations implemented as an internal DSL in Scala, deliverable as a Scala library.
- A lens-based view synchronization architecture that integrates with EMF-based technologies and can be used in multi-view modeling environments.

[3] http://www.scala-lang.org

References

1. Francisco Pérez Andrés, Juan de Lara, and Esther Guerra. Domain specific languages with graphical and textual views. In Andy Schürr, Manfred Nagl, and Albert Zündorf, editors, *AGTIVE*, volume 5088 of *Lecture Notes in Computer Science*, pages 82–97. Springer, 2007.
2. Paolo Atzeni and Riccardo Torlone. Management of multiple models in an extensible database design tool. In *Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology, Avignon, France, March 25-29, 1996, Proceedings*, volume 1057 of *Lecture Notes in Computer Science*, pages 79–95. Springer, 1996.
3. Aaron Bohannon, J. Nathan Foster, Benjamin C. Pierce, Alexandre Pilkiewicz, and Alan Schmitt. Boomerang: Resourceful lenses for string data. In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), San Francisco, CA*, pages 407–419, January 2008.
4. U. Dayal and P.A. Bernstein. On the correct translation of update operations on relational views. *ACM Transactions on Database Systems (TODS)*, 7(3):381–416, 1982.
5. Zinovy Diskin, Yingfei Xiong, and Krzysztof Czarnecki. From state- to delta-based bidirectional model transformations. In *Theory and Practice of Model Transformations, Third International Conference, ICMT 2010, Malaga, Spain, June 28-July 2, 2010. Proceedings*, volume 6142 of *Lecture Notes in Computer Science*, pages 61–76. Springer, 2010.
6. H. Ehrig, K. Ehrig, C. Ermel, F. Hermann, and G. Taentzer. Information preserving bidirectional model transformations. In *Fundamental Approaches to Software Engineering, 10th International Conference, FASE 2007*, volume 4422, page 72. Springer, 2007.
7. J.N. Foster. *Bidirectional Programming Languages*. PhD thesis, University of Pennsylvania, 2009.
8. J.N. Foster, M.B. Greenwald, J.T. Moore, B.C. Pierce, and A. Schmitt. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 29(3):17, 2007.
9. Miguel Garcia. Bidirectional synchronization of multiple views of software models. In Dirk Fahland, Daniel A. Sadilek, Markus Scheidgen, and Stephan Weißleder, editors, *Proceedings of the Workshop on Domain-Specific Modeling Languages (DSML-2008)*, volume 324 of *CEUR-WS*, pages 7–19, 2008.
10. Anders Hessellund, Krzysztof Czarnecki, and Andrzej Wasowski. Guided development with multiple domain-specific languages. In *MoDELS*, pages 46–60, 2007.
11. Anders Hesselund. *Domain-specific Multimodeling*. PhD thesis, IT University of Copenhagen, 2009.
12. T. Hettel, M. Lawley, and K. Raymond. Towards model round-trip engineering: an abductive approach. *Theory and Practice of Model Transformations*, pages 100–115, 2009.
13. Soichiro Hidaka, Zhenjiang Hu, Hiroyuki Kato, and Keisuke Nakano. A compositional approach to bidirectional model transformation. In *ICSE Companion*, pages 235–238, 2009.
14. Z. Hu, S.C. Mu, and M. Takeichi. A programmable editor for developing structured documents based on bidirectional transformations. *Higher-Order and Symbolic Computation*, 21(1):89–118, 2008.

15. Oleg Kiselyov, Ralf Lämmel, and Keean Schupke. Strongly typed heterogeneous collections. In *Haskell '04: Proceedings of the ACM SIGPLAN workshop on Haskell*, pages 96–107. ACM Press, 2004.
16. M. Lawley and J. Steel. Practical declarative model transformation with Tefkat. In *Satellite Events at the MoDELS 2005 Conference*, pages 139–150. Springer, 2006.
17. K. Matsuda, Z. Hu, K. Nakano, M. Hamana, and M. Takeichi. Bidirectionalization transformation based on automatic derivation of view complement functions. In *Proceedings of the 12th ACM SIGPLAN international conference on Functional programming*, page 58. ACM, 2007.
18. Andy Schürr and Felix Klar. 15 years of triple graph grammars. In *ICGT*, pages 411–425, 2008.
19. P. Stevens. Bidirectional model transformations in QVT: Semantic issues and open questions. In *Proc. of the 10 Int. Conf. on Model Driven Engineering Languages and Systems*, Lecture Notes in Computer Science, pages 1–14. Springer, 2007.
20. M. Wimmer and G. Kramler. Bridging grammarware and modelware. In *Satellite Events at the MoDELS 2005 Conference*, volume 3844 of *Lecture Notes in Computer Science*, pages 159–168. Springer, 2005.
21. Yingfei Xiong. *A Language-based Approach to Model Synchronization in Software Engineering*. PhD thesis, Department of Mathematical Informatics, University of Tokyo, September 2009.
22. Yingfei Xiong, Dongxi Liu, Zhenjiang Hu, Haiyan Zhao, Masato Takeichi, and Hong Mei. Towards automatic model synchronization from model transformations. In *ASE*, pages 164–173, 2007.