1B-7

Realizing Bidirectional Graph Transformations From Bidirectional Tree Transformations

Yingfei Xiong[1], Zhenjiang Hu[1], Dongxi Liu[1], Haiyan Zhao[2], Hong Mei[2], Masato Takeichi[1]

[1]Department of Mathematical Informatics
Graduate School of Information Science and Technology
University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Tokyo 113-8656, Japan
{hu,liu,takeichi,Yingfei_Xiong}@mist.i.u-tokyo.ac.jp
[2]Institute of Software
School of Electronics Engineering and Computer Science
Peking University, Beijing, 100871, China
{meih,zhhy}@sei.pku.edu.cn

Abstract

Bidirectional transformations are useful to maintain consistency between source data and target data. So far, researchers have proposed several theories and tools for bidirectional transformations on tree structures. However, as far as we know, there is no widely-recognized theory for bidirectional transformations on graphs.

In this paper, we propose an approach to constructing bidirectional graph transformations from existing bidirectional tree transformations and show how they can be useful to support better consistency and traceability between different models in software development.

1 Introduction

In many cases people need to view the data from different perspectives, thus we have to map the source data into different views. Such kind of mappings are called transformations. On the other hand, in some cases people want to modify the view and then reflect the modification backwardly to the source. A transformation that supports such kind of backward transformation is called a bidirectional transformation.

Bidirectional transformations have many applications in different kinds of areas. For example, bidirectional transformations can be used to support the synchronization among heterogeneously pieces of data [4], to create visual editors that generate and maintain views from the editing actions of users [5], even to support the classic view-updating problem in database systems [1].

Recently, researchers have proposed many tools and theories to support bidirectional transformations, especially on tree data structures. Two notable examples are Harmony [2] and BiXJ [6]. These two tools both work on XML files. Users write the code for forward transforming XML files and these tools automatically support the backward transformation.

Model-driven architecture (MDA) [3] is a new software development paradigm where models are first-class entities and the refinement relationships between different development stages are described by model transformations. Research has shown that introducing bidirectionality into model transformations can ensure better consistency and traceability between different models [4]. However, to the best of our knowledge, current model transformation tools either only can support uni-directional transformation, or require users to manually implement the backward transformation. Without automatical tool support, MDA cannot benefit from the consistency and traceability enhancement.

Model transformations are essentially transformations on graph structures. A bidirectional graph transformation tool will be able to support model transformations in MDA. In this paper we propose an approach to constructing bidirectional graph transformations from existing bidirectional tree transformations. In this way we can make use of the widely-existing tree transformation tools like BiXJ, and benefit from the stability and matureness of these tools.

2 Approach

In this section we describe our approach. Due to space limit, our description will be in plain text, but all the definitions and theories in this section have formal counterparts, which will be summarized in an extended version.

2.1 Duplication Transformation

Here we consider a specific type of bidirectional transformation: duplication. A duplication transformation duplicates some data items in the source, and when any replica is modified, this modification can be reflected back to the data item in the source and all the replicas. For

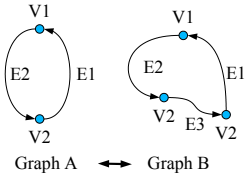

Figure 1. A Graph Transformation

example, Figure 1 describes a duplication graph transformation. In Figure 1 the vertexes and the edges are corresponding if they have the same value. The node "V2" in Graph A are duplicated twice in Graph B. If we modified any "V2" in Graph B into, say, "V3". The "V2" in Graph A and the other "V2"s in Graph B will also be changed into "V3". To support graph transformations, we will make use of the duplication transformation in the tree transformation.

2.2 Realization of Bidirectional Transformation

The first step in realizing graph transformations with tree transformations is to represent graphs using trees. Comparing graphs with trees, we can discover that graphs are in fact trees with node sharing. For example, in Graph A, the vertex "V1" is shared by two edges: "E1" and "E2". Then we can represent a graph with a meta tree and a tree transformation that captures the node-sharing information.

We illustrate this process by an example. Consider Graph A in Figure 1. The corresponding meta tree and the tree transformation are shown in the left part of Figure 2. All the edges and vertexes of Graph A are listed as the leaves of the node "edges" and the node "vertexes", respectively. The node-sharing information is missing in the meta tree, that is, we do not know to which nodes the vertexes are connected. Then in Transformation T1, we duplicate the vertexes nodes as the corresponding leaves of the edge nodes. The direct sub node of an edge means the source of that edge and the sub node of the source is the target of the edge. The grey lines in Figure 2 show from which nodes in the source nodes in the target originate. For simplicity, we only draw the lines related to a typical vertex, "V2", in Figure 2.

In order to transform the tree back into a graph, we also assigned a unique ID to each vertex node and each edge node. These IDs are also managed by Transformation T1. For simplicity, the IDs are not shown in Figure 2.

The next step in realizing graph transformation is to represent graph transformations by tree transformations. The transformation between Graph A and Graph B in Figure 1 is represented by the Transformation T2, which is shown the right part of Figure 2. The tree of Graph B is constructed by duplicating the nodes in the tree of Graph A, in a way similar to Transformation T1. To better illustrate this, we also draw the grey lines related to the vertex

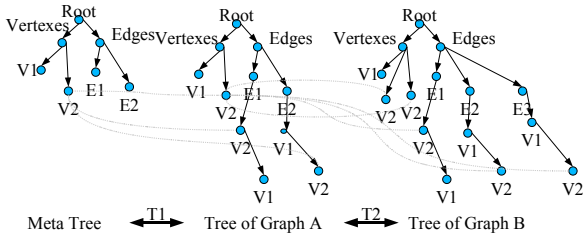

Figure 2. Bidirectional Graph Transformations via Bidirectional Tree Transformation

"V2" in Figure 2.

With the composite effect of T1 and T2, we can ensure the modifications in Graph B can be correctly reflected back. For example, suppose one "V2" node in Graph B is modified to, say, "V3". Transformation T2 ensures that the "V2" node under the "vertexes" node in the tree of Graph A is modified to "V3" and all replicas in the tree of Graph B are modified to "V3". On the other hand, Transformation T1 ensures that all "V2" in the tree of Graph A are modified to "V3" and the meta tree is also modified correspondingly.

To verify the feasibility of our approach, we also experimented the approach by using BiXJ to model a model-enabled software development process [4]. The result shows that our approach is adequate to support the transformation between models.

References

[1] A. Bohannon, J. A. Vaughan, and B. C. Pierce. Relational lenses: A language for updateable views. In *Principles of Database Systems (PODS)*, 2006. Extended version available as University of Pennsylvania technical report MS-CIS-05-27.

[2] 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 *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), Long Beach, California*, pages 233–246, 2005.

[3] D. S. Frankel. *Model Driven Architecture: Applying MDA to Enterprise Comput ing*. John Wiley & Sons, 2003.

[4] Z. Hu, D. Liu, H. Mei, M. Takeichi, Y. Xiong, and H. Zhao. A compositional approach to bidirectional model transformation. Technical Report METR 2006-54, Department of Mathematical Informatics, University of Tokyo, October 2006.

[5] Z. Hu, S.-C. Mu, and M. Takeichi. A programmable editor for developing structured documents based on bidirectional transformations. In *Proceedings of ACM SIGPLAN 2004 Symposium on Partial Evaluation and Program Manipulation*, pages 178–189. ACM Press, 2004.

[6] D. Liu, Z. Hu, M. Takeichi, K. Kakehi, and H. Wang. A Java library for bidirectional XML transformation. *JSSST Computer Software*, to appear, 2006.