Practical Aspects of Bidirectional Graph Transformations

Zhenjiang Hu
National Institute of Informatics, Japan
hu@nii.ac.jp

Abstract

Bidirectional transformation [1, 7] consists of a pair of transformations, describing not only a forward transformation from a source to a view, but also a backward transformation showing how to reflect the changes in the view to the source. Bidirectional transformation provides a novel mechanism for synchronizing and maintaining the consistency of information between input and output, and has many potential applications in software development, including model synchronization, round-trip engineering, software evolution, multiple-view software development, reverse software engineering, as well as the well-known view updating mechanism which has been intensively studied in the database community for decades.

To support systematical development of well-behaved bidirectional transformations, much research has been devoted to design of bidirectional languages that can be interpreted both forwardly and backwardly while guaranteeing the roundtrip property between the forward and the backward transformations. Despite many promising results, most of them are limited to lists and trees [2, 6, 9, 10]. In fact, there are challenges in designing a language for bidirectional transformation on graphs. First, unlike lists and trees, there is no unique way to represent, construct, and decompose a general graph, which requires more precise definition of equivalence between two graphs. Second, graphs have sharing nodes and cycles, which makes forward computation much more complicated than that on trees (let alone to say about backward computation), where naïve computation on graphs would visit the same nodes many times and possibly infinitely often.

We have challenged the problem of bidirectional transformations on graphs, and succeeded in bidirectionalizing graph queries (in UnQL) [3, 4, 8] and implementing a bidirectional graph transformation engine called GRoundTram [5]. In GRoundTram, graphs are treated as regular trees and manipulated by structural recursion that enjoys a nice bulk and bidirectional semantics. Although GRoundTram has been successfully applied to nontrivial model-code co-evolution [11], there are still many practical issues that should be addressed to make it be more useful.

In this talk, I shall briefly explain our solution to the problem of bidirectional graph transformation and demonstrate some applications in bidirectional model-driven software development, and focus on discussing practical issues in manipulating various graphs (such as unordered, ordered, and probability graphs), determining backward transformation, and improving efficiency and scalability.

PEPM'13, January 21–22, 2013, Rome, Italy.
ACM 978-1-4503-1842-6/13/01.

Categories and Subject Descriptors Software [*PROGRAMMING TECHNIQUES*]: Automatic Programming

General Terms Languages

Keywords Bidirectional Graph Transformation

References

[1] K. Czarnecki, J. N. Foster, Z. Hu, R. Lämmel, A. Schürr, and J. F. Terwilliger. Bidirectional transformations: A cross-discipline perspective. In *International Conference on Model Transformation (ICMT 2009)*, pages 260–283. LNCS 5563, Springer, 2009.

[2] 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 Trans. Program. Lang. Syst.*, 29(3), 2007.

[3] S. Hidaka, Z. Hu, K. Inaba, H. Kato, K. Matsuda, and K. Nakano. Bidirectionalizing graph transformations. In *ACM SIGPLAN International Conference on Functional Programming*, pages 205–216. ACM, 2010.

[4] S. Hidaka, Z. Hu, K. Inaba, H. Kato, K. Matsuda, K. Nakano, and I. Sasano. Marker-directed optimization of uncal graph transformations. In G. Vidal, editor, *LOPSTR*, volume 7225 of *Lecture Notes in Computer Science*, pages 123–138. Springer, 2011.

[5] S. Hidaka, Z. Hu, K. Inaba, H. Kato, and K. Nakano. GRoundTram: An integrated framework for developing well-behaved bidirectional model transformations (short paper). In *26th IEEE/ACM International Conference On Automated Software Engineering*, pages 480–483. IEEE, 2011.

[6] 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(1-2):89–118, 2008.

[7] Z. Hu, A. Schürr, P. Stevens, and J. F. Terwilliger. Bidirectional transformation "bx" (dagstuhl seminar 11031). *Dagstuhl Reports*, 1 (1):42–67, 2011.

[8] K. Inaba, S. Hidaka, Z. Hu, H. Kato, and K. Nakano. Graph-transformation verification using monadic second-order logic. In P. Schneider-Kamp and M. Hanus, editors, *PPDP*, pages 17–28. ACM, 2011.

[9] K. Matsuda, Z. Hu, K. Nakano, M. Hamana, and M. Takeichi. Bidirectionalization transformation based on automatic derivation of view complement functions. In *12th ACM SIGPLAN International Conference on Functional Programming (ICFP 2007)*, pages 47–58. ACM Press, Oct. 2007.

[10] J. Voigtländer, Z. Hu, K. Matsuda, and M. Wang. Combining syntactic and semantic bidirectionalization. In *ACM SIGPLAN International Conference on Functional Programming*, pages 181–192. ACM, 2010.

[11] Y. Yu, Y. Lin, Z. Hu, S. Hidaka, H. Kato, and L. Montrieux. Maintaining invariant traceability through bidirectional transformations. In M. Glinz, G. C. Murphy, and M. Pezzè, editors, *34th International Conference on Software Engineering*, pages 540–550. IEEE, 2012.