How to Regain Equilibrium without Losing your Balance? Scenarios for Bx Deployment (Discussion Paper)

James McKinna Perdita Stevens
LFCS, School of Informatics
University of Edinburgh
firstname.lastname@ed.ac.uk

Abstract

Much of the bx literature, especially that couched in terms of consistency restoration, presupposes a workflow based on strict sequentialisation of model updates, and in some cases explicitly via a centralised broker of shared state. Put another way: the underlying mathematical semantics of bx take little account of the deployment context for such artefacts, which, from a user's perspective, are typically concurrent and distributed. This discussion proposal invites discussion of the bx deployment context.

1 Introduction

This "short paper" is a proposal that the workshop have a space for discussing the following issue which we think needs more attention than it has had so far. We include some points for discussion, and a non-exhaustive list of relevant references; we hope that other points and references can be added during discussion, and perhaps a full paper may be produced in response.

Going back to Meertens [Mee98, Mee05], a bidirectional transformation or bx can be considered to be an artefact which embodies both a specification of consistency between two or more models, e.g. as a relation on a pair of model spaces, and a specification of how consistency should be restored. The latter is often – by Meertens, in the lens literature, in MDD settings including by the authors – presented via pure functions, the consistency restorers, which take one or more models as arguments, and return a result which is understood as a modified version of one of the arguments, intended to replace it. In other approaches, invoking a consistency restorer may directly edit the user's model, or may return an edit that could be applied to a model, rather than a full model.

We have in recent years developed approaches to such bx which make explicit the side-effecting — effectful — nature of the consistency restorers [ACG+15, ASMG15]; in our approach, the user of the bx is presented with a monadic interface allowing `get` and `set` operations, instead of with pure functions such as $\overrightarrow{R}$ and $\overleftarrow{R}$. In particular, the monadic interface allows us to model explicitly the use of effects such as I/O or non-determinism in computing model revisions. Interestingly, our models seem, for mathematical tractability, to oblige us to consider each model as a derived view of a "central" — shared (even if hidden) — state space, similar to Johnson and Rosebrugh's recent work on spans of lenses [JR14].

Copyright © by the paper's authors. Copying permitted for private and academic purposes.

In: A. Anjorin, J. Gibbons (eds.): Proceedings of the Fifth International Workshop on Bidirectional Transformations (Bx 2016), Eindhoven, The Netherlands, April 8, 2016, published at http://ceur-ws.org

All such approaches, however, change rather than solve the problem: when (if ever) does the user of the bx invoke these operations and what (if anything) do they have to do before they can continue with their real work? Suppose a bx engine implemented bx as presented in any of these ways; how should it be used? Perhaps this issue has not had as much attention as it should have done because of the connections with bidirectional *programming*, a rather different proposition from bidirectional *transformation*.

2 Points for discussion

1. The connection between bx and related concepts including version control systems, synchronizers and collaborative working systems can be both helpful and deeply confusing.
2. It can be helpful because most of us have extensive experience working with the latter, whether it is when we maintain copies of parts of our filesystem in more than one place, or when we collaborate on papers.
3. It is confusing because it tempts us to think in terms of "conflict", a very unhelpful notion for bx (to be deliberately provocative). The very notion of "conflict" between your model and mine presupposes that our models are supposed to be identical (at least in the part where we can observe conflict). We mean by the term something like: things that should be identical are not identical, must be made identical, and it is not clear which of the current things should "win".
4. Implicit or explicit is the presence of some kind of archive: the basic picture of consistency restoration has been augmented with auxiliary machinery such as generalised `diff3` [KKP07].
5. But neither the "to be consistent is to be identical" view, nor the presence of an archive, is essential to talking about concurrent editing of two models.
6. People sometimes assume that in evaluating, say, $\overrightarrow{R}(m,n)$, we may assume that n is a model that was consistent with some earlier version of model m. No: nothing in the formalism requires this.
7. [CGMS15] discussed the distinction between *strong* and *weak* versions of properties universally quantified over pairs of models, such as continuity. Essentially the strong form considers concurrent editing of the models, while the weak form does not.
8. In MDD, if I am working on a model which is connected via a bx to your model, what kind of control do I *want* over when consistency is restored, and what *can I have* in principle? What are the implications for a usable tool?
9. In version control systems we are used to explicitly invoking operations, and we are familiar with the ways in which this can fail (e.g., minimise conflicts by always doing `svn up` (get) before you `svn commit` (set)), but we know that this will not always avoid conflict.
10. In cloud systems such as Dropbox, we do not explicitly invoke operations, but the systems' behaviour can be perplexing even when not broken [HPAN15].
11. If the bx is not history ignorant – and we know it is unlikely to be – then exactly when consistency is restored may materially affect the result. So perhaps we don't want it done invisibly in the background.
12. There is a tension between: *I don't want to work for a long time on a model that is now inconsistent with your model, in case I am wasting effort*; and, *I don't want my flow broken by my tool telling me that something has changed in your model.*
13. Is it acceptable for the bx engine to change part of my model that I'm not actually looking at right now, without telling me, in order to restore consistency?
14. What can be learned from categorisations such as [DGWC16, AC07]? Or from available (bx or round-tripping) tools?
15. There is a distinction between models having been concurrently edited, and the user requiring both models to be changed in order to restore consistency, studied in [XSHT09] for example. What are the implications, for practical scenarios, of our choices?

References

[AC07] Michal Antkiewicz and Krzysztof Czarnecki. Design space of heterogeneous synchronization. In Ralf Lämmel, Joost Visser, and João Saraiva, editors, *Generative and Transformational Techniques in Software Engineering II, International Summer School, GTTSE 2007, Braga, Portugal, July 2-7, 2007. Revised Papers*, volume 5235 of *Lecture Notes in Computer Science*, pages 3–46. Springer, 2007.

[ACG+15] Faris Abou-Saleh, James Cheney, Jeremy Gibbons, James McKinna, and Perdita Stevens. Notions of bidirectional computation and entangled state monads. In Ralf Hinze and Janis Voigtländer, editors, *Mathematics of Program Construction - 12th International Conference, MPC 2015, Königswinter, Germany, June 29 - July 1, 2015. Proceedings*, volume 9129 of *Lecture Notes in Computer Science*, pages 187–214. Springer, 2015.

[ASMG15] Faris Abou-Saleh, James McKinna, and Jeremy Gibbons. Coalgebraic aspects of bidirectional computation. In Alcino Cunha and Ekkart Kindler, editors, *4th International Workshop on Bidirectional Transformations*, pages 16–30. CEUR Workshop Proceedings, July 2015.

[CGMS15] James Cheney, Jeremy Gibbons, James McKinna, and Perdita Stevens. Towards a principle of least surprise for bidirectional transformations. In *Proceedings of Bx 2015*, volume 1396 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[DGWC16] Zinovy Diskin, Hamid Gholizadeh, Arif Wider, and Krzysztof Czarnecki. A three-dimensional taxonomy for bidirectional model synchronization. *Journal of Systems and Software*, 111:298–322, 2016.

[HPAN15] John Hughes, Benjamin C. Pierce, Thomas Arts, and Ulf Norell. Mysteries of dropbox, October 2015. Downloaded January 2016.

[JR14] Michael Johnson and Robert Rosebrugh. Spans of lenses. `http://ceur-ws.org/Vol-1133/#bx`, 2014.

[KKP07] Sanjeev Khanna, Keshav Kunal, and Benjamin C. Pierce. A formal investigation of diff3. In Arvind and Prasad, editors, *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, December 2007.

[Mee98] Lambert Meertens. Designing constraint maintainers for user interaction. Unpublished manuscript, available from `http://www.kestrel.edu/home/people/meertens/`, June 1998.

[Mee05] Lambert Meertens. Designing constraint maintainers for user interaction. In Shin-Cheng Mu, editor, *Third Workshop on Programmable Structured Documents*, pages 1–3, PSD Laboratory, Tokyo University, 2005.

[XSHT09] Yingfei Xiong, Hui Song, Zhenjiang Hu, and Masato Takeichi. Supporting parallel updates with bidirectional model transformations. In Richard F. Paige, editor, *Theory and Practice of Model Transformations, Second International Conference, ICMT 2009, Zurich, Switzerland, June 29-30, 2009. Proceedings*, volume 5563 of *Lecture Notes in Computer Science*, pages 213–228. Springer, 2009.