Bidirectional Transformations for Spreadsheets

Jácome Cunha, João Saraiva, Martin Erwig

September 6, 2010

Summary

Spreadsheets are widely used by non-professional programmers to develop business applications. These programmers vastly outnumber the professional ones creating millions of spreadsheets every year (Scaffidi et al. 2005). Spreadsheets are quite flexible, but they are also very error prone (Rajalingham et al. 2001).

We have shown that the use of business logic specifications for spreadsheets can prevent errors and improve users productivity (Cunha 2010). Using such a Model-driven Engineering (MDE) approach, the user is responsible to synchronize the (spreadsheet) models and instances. This is a complex and error prone task, since users may introduce data that violate the model. Moreover, the spreadsheet model may evolve over time and, therefore, the data has to be adapted to conform to the new model.

In this project we will study bidirectional transformations (BT) (Czarnecki et al. 2009) not only to synchronize a model with its instances, but also to allow updates in different views on spreadsheets. We will also study how specific features of spreadsheets that can influence BTs, namely, the space embedded computations, the lack of recursion and the use of layout.

State of the art

Bidirectional Transformations

BTs is a recent and active area of research. A unidirectional program transformation is a function that maps inputs, usually called source, into output, called target. A bidirectional transformational system considers also a function that maps the target into the source. In such systems users can update both the source and the target and the BT system is responsible for synchronizing the two objects.

Consider for example a WYSIWYG HTML editor: users can edit both a pretty printed version of the document as displayed by a browser or the HTML textual representation. BT provides programmers with powerful mechanisms to interact with software systems.

A BT between two sources of information A and B (e.g., a database and a view) comprises a pair of unidirectional transformations: one from A to B, called the forward transformation, or "get" , and another from B back to A, called the backward or reverse transformation or "put" . In these cases, A is called the input (source) and B is called the output (target). In BTs, the "get"

can be an arbitrary function and since it may discard information, the "put" typically takes two arguments: an updated output and the original input as illustrated in Figure 1 in the file attached to this proposal.

Programming Languages: Programs in the language biXid consist of pairs of intertwined grammars (Kawanaka and Hosoya 2006), and the transformations are obtained by parsing according to the rules in one grammar and pretty printing according to the rules in the other.

Pierce et al. designed lenses theory, that is, primitives that denote two transformations and combining forms that preserve bidirectionality (Bohannon et al. 2008).

Databases: To solve one of the main problems in databases, the view update problem (Dayal and Bernstein 1982), recent studies focused in the investigation of algebraic transformations with known properties, for instance, Both-as-View (McBrien and Poulovassilis 2003).

Research has been done to investigate transformations between different metamodels of databases, in particular: objects, XML and relations. The objects-relations case is well established (Melnik et al. 2008), but the other cases are substantially more difficult, given the difference of their expressiveness (Lämmel and Meijer 2007).

Model-Driven Engineering (MDE): Diskin proposed an algebraic approach (Diskin 2008) for MDE. Another automatic approach is described by Xiong et al. in (Xiong et al. 2007).

The model-transformation scenario for co-evolution of metamodels and models, is closely related to BT. Such co-evolution is an instance of the notion of coupled transformations (Vermolen and Visser 2008).

In previous work we have explored model-based evolution of spreadsheets, under the MDE setting (Cunha 2010).

Spreadsheets

There is little work by the programming language community on the foundations of spreadsheets, being the work of Burnett and Erwig the exception.

Abraham et al. have applied to spreadsheets various software engineering principles In (Abraham and Erwig 2007) they describe techniques for checking the impact of changing a cell value in the rest of the spreadsheet.

They also describe how to automatically infer templates from spreadsheets (Abraham and Erwig 2006). These templates ensure that the user's changes conform to the template. The performance of a tool to automatically do the inference is also presented.

In the context of the PURe project, Oliveira (Oliveira 2007) describes a pointfree calculus and refinement laws for spreadsheet. He shows how a binary representation of an n-ary relation makes sense in the context of a database theory, and how lengthy formulae can become very simple and elegant. We have used this calculus in our work on mapping spreadsheets to databases and back (Cunha et al. 2009).

Objectives

With our work, we will answer the following questions:

- There are a plenty of opportunities to study BTs in the spreadsheet realm, but the particularities of spreadsheets pose some challenges. For example, will the fact that the data and the computations are not separated cause difficulties in the calculation of the new inputs (using "put")? Are the current theories and techniques for BTs enough for spreadsheets? Can the lack of recursion in spreadsheets be beneficial in the application of BTs since it is a simplification compared to other languages?

- We will study to which extent it is possible to keep two or more spreadsheets synchronized. For example, can we synchronize without loosing any information from both spreadsheets? Will we lose data or layout information? To which extent can the specifics of spreadsheet help in the synchronization?

- To which extent can we keep a spreadsheet and a database synchronized? Can we keep the layout information in the database representation, for example, using an auxiliary table? And how to create a layout for the data in a spreadsheet? Each table in a different sheet?

- We will study the best way to keep spreadsheet instances conform to their business models. Do we need to constantly enforce this conformance? Or can we change one of them and then synchronize?

- The ultimate objective is to produce techniques that can help users. Thus, some questions arise: Can these techniques help users to safely work with spreadsheets? Will they help users committing fewer errors? And helping them being faster doing their tasks?

- Can all the tasks we proposed result in non-invasive techniques and tools? The less interruption and distraction we cause to the user, the better: if the users get new features with no effort, our results will have more impact.

Description of the Tasks

Bidirectional Spreadsheets

Spreadsheets have two possible views: one view contemplates the formulas and another focused on the results of these formulas.

One of the biggest sources of errors in spreadsheets is the replacement of a formula by a value, involuntarily or to have the correct value in such cell. In fact, this can be useful: if this change was back-propagated to the rest of the spreadsheet the user could have the perception of the reason why that result is wrong. In fact, a similar approach is followed in (Abraham and Erwig 2007). We will use BTs to achieve this goal. The "get" function is the usual spreadsheet mechanism, that is, receives a spreadsheet and calculates the results of all formulas. The "put" function will receive the same spreadsheet and the results that the user wants and will produce a new updated spreadsheet with the correct values in the corresponding cells. As expected, the "put" function may be ambiguous. How to handle this ambiguity is a challenge and deserves further studied.

Spreadsheets greatly differ from other languages also because they have space embedding computations. Their layout must be considered in the transforma-

tions, which does not occur in other languages. In fact, this can help to produce better "put" functions because usually it is possible to infer knowledge from the layout (Cunha et al. 2009).

Another peculiarity of spreadsheets is that they lack recursion and thus computations are simpler. This means that the calculation of the "put" function can be more precise.

Incremental computation is about maintaining the input-output relationship of a program, as the input undergoes changes. The changes in the input may be such that one do not need a complete re-computation of the output. Adaptive Programming and Function Memoization are among the techniques proposed to achieve incremental computation (Acar et al. 2002). We will develop the BTs techniques here presented in an incremental way.

Synchronization of Spreadsheets

Imagine a situation in a company where one department works on part of a spreadsheet, a view, and another department on a second view. The administration probably needs to view the complete spreadsheet. The synchronization of two or more software artifacts is not an easy task.

In previous work we defined transformations under the theory of data refinements (Cunha et al. 2009) defining two levels of transformations: on one level, functions to transform one datatype into another and vice versa, and on a second level, functions to migrate the values of such types back and forth. This is almost a BT, but it does not synchronize the changes. Instead it erases the old value and replaces it by the new one.

We will define a formal language for specifying the synchronization of spreadsheets. With this language it will be possible to define a spreadsheet transformation that is bidirectional. We will take under consideration spreadsheet specifics, that we know from experience (Cunha et al. 2010), can be very useful.

Synchronizing Spreadsheets and Databases

One can imagine a situation where a database is accessed by several kinds of users: professional programmers probably access the database using the most common language, SQL, and non-professional users can access it using a spreadsheet as a view.

In previous work we defined a formal connection between spreadsheets and databases (Cunha et al. 2009). Although this connection allows the transformation of a spreadsheet into a database and vice versa, it does not allow synchronization between them. In this task we will define BTs between spreadsheets and databases. Both the spreadsheet and the database can be the source and the view. Thus, the transformations must be studied in both directions, that is, we need to define the "put" and the "get" functions from spreadsheets to databases and vice versa. The changes performed by each kind of users should be reflected in all the views of the database. This problem is know as the view-update problem and has been studied in the context of databases (Dayal and Bernstein 1982). To create a solution for databases and spreadsheets is challenging for several reasons: in contrast to databases, spreadsheets have spacial constraints and, moreover, spreadsheets are usually not normalized as databases.

Notice that tools such as Excel can import data from database, but synchronization is not possible. Our work will extend importing to synchronization.

Bidirectional Models for Spreadsheets

Another interesting research path is the BT between a model of a spreadsheet and its instances. One can imagine a situation where a professional programmer could define a formal model for a spreadsheet using, for example, ClassSheet (Engels and Erwig 2005) models. On the other hand, end users would have their own spreadsheet to work on.

We will define techniques to keep the spreadsheets synchronized with the models. Moreover, to reflect changes in the spreadsheet in the model is also an objective. In fact, the bidirectional interaction between the model and the instances is the goal. In this case, the source is the spreadsheet and the target the model meaning that the "get" would be from the spreadsheet to the model and the "put" from the model (and a spreadsheet) to the new updated spreadsheet.

Human Validation

It is important to formally validate the techniques developed during this project, but it is equally important to validate them and the results with users of spreadsheets.

We will organize and run studies to assess how well BTs work in practice. We will study their impact in error committing and speed of users doing their tasks. The experience of the team in such studies can help us to define studies that produce significant validation for our techniques.

This post-doc project will be developed in the context of both a research project granted by FCT, "SpreadSheets as a Programming Paradigm", and an industrial connection to Software Improvement Group, The Netherlands, that is involved in projects with large and real spreadsheets. We will use some of those spreadsheets to validate our approach.

Bidirectional Spreadsheets Framework

The techniques produced by our work can have much more impact if tools are available for users. This team has a lot of experience developing tools, also for end users (Cunha et al. 2009, 2010; Abraham and Erwig 2006). We have the needed experience to design and implement a framework integrating the techniques studied in this work. A web page containing the information of the project will be available and this framework will be distributed as open source software so it can be reused in other projects.

Notice that usually end users are reluctant to change the way they normally work. To overcome this issue, we will produce techniques and tools that work as automatically as possible, that is, without interrupting the user. In fact, we should let end users do work in the way they are used to working, but inject good design decisions into their existing practices.

We would like to stress that the line of work described here is not a simple continuation of the recent PhD thesis of the candidate. However, the expertise acquired during the PhD period will be of great relevance to achieve the goals of this project.

References

Robin Abraham and Martin Erwig. Inferring templates from spreadsheets. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 182–191, New York, NY, USA, 2006. ACM. ISBN 1-59593-375-1. doi: http://doi.acm.org/10.1145/1134285.1134312.

Robin Abraham and Martin Erwig. Goaldebug: A spreadsheet debugger for end users. In *ICSE '07: Proceedings of the 29th international conference on Software Engineering*, pages 251–260, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2828-7. doi: http://dx.doi.org/10.1109/ICSE.2007.39.

Umut A. Acar, Guy E. Blelloch, and Robert Harper. Adaptive functional programming. In *POPL '02: Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 247–259, New York, NY, USA, 2002. ACM. ISBN 1-58113-450-9. doi: http://doi.acm.org/10.1145/503272.503296.

Aaron Bohannon, J. Nathan Foster, Benjamin C. Pierce, Alexandre Pilkiewicz, and Alan Schmitt. Boomerang: resourceful lenses for string data. In *POPL '08: Proceedings of the 35th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 407–419, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-689-9. doi: http://doi.acm.org/10.1145/1328438.1328487.

Jácome Cunha. *Model-based Spreadsheet Engineering*. PhD thesis, University of Minho, 2010.

Jácome Cunha, ao Saraiva, Jo and Joost Visser. From spreadsheets to relational databases and back. In *PEPM '09: Proceedings of the 2009 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation*, pages 179–188, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-327-3. doi: http://doi.acm.org/10.1145/1480945.1480972.

Jácome Cunha, Martin Erwig, and João Saraiva. Automatically inferring classsheet models from spreadsheets. In *VLHCC '10: Proceedings of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE Computer Society, 2010. to appear.

Krzysztof Czarnecki, J. Nathan Foster, Zhenjiang Hu, Ralf Lämmel, Andy Schürr, and James F. Terwilliger. Bidirectional transformations: A cross-discipline perspective. 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 260–283. Springer, 2009. ISBN 978-3-642-02407-8.

Umeshwar Dayal and Philip A. Bernstein. On the correct translation of update operations on relational views. *ACM Trans. Database Syst.*, 7(3):381–416, 1982. ISSN 0362-5915. doi: http://doi.acm.org/10.1145/319732.319740.

Zinovy Diskin. Algebraic models for bidirectional model synchronization. In *MoDELS '08: Proceedings of the 11th international conference on Model Driven Engineering Languages and Systems*, pages 21–36, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-87874-2. doi: http://dx.doi.org/10.1007/978-3-540-87875-9_2.

G. Engels and M. Erwig. ClassSheets: automatic generation of spreadsheet applications from object-oriented specifications. In David Redmiles, Thomas Ellman, and Andrea Zisman, editors, *20th IEEE/ACM Int. Conf. on Automated Sof. Eng., Long Beach, USA*, pages 124–133. ACM, 2005.

Shinya Kawanaka and Haruo Hosoya. biXid: a bidirectional transformation language for XML. In *ICFP '06: Proceedings of the eleventh ACM SIGPLAN international conference on Functional programming*, pages 201–214, New York, NY, USA, 2006. ACM. ISBN 1-59593-309-3. doi: http://doi.acm.org/10.1145/1159803.1159830.

Ralf Lämmel and Erik Meijer. Revealing the x/o impedance mismatch: changing lead into gold. In *SSDGP'06: Proceedings of the 2006 international conference on Datatype-generic programming*, pages 285–367, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-76785-1, 978-3-540-76785-5.

Peter McBrien and Alexandra Poulovassilis. Data integration by bi-directional schema transformation rules. In Umeshwar Dayal, Krithi Ramamritham, and T. M. Vijayaraman, editors, *Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India*, pages 227–238. IEEE Computer Society, 2003. ISBN 0-7803-7665-X. URL `http://csdl.computer.org/comp/proceedings/icde/2003/2071/00/20710227abs.htm`.

Sergey Melnik, Atul Adya, and Philip A. Bernstein. Compiling mappings to bridge applications and databases. *Acm Transactions On Database Systems*, 33(4), November 2008. doi: DOI10.1145/1412331.1412334.

José N. Oliveira. Transforming data by calculation. In Ralf Lämmel, Joost Visser, and João Saraiva, editors, *GTTSE*, volume 5235 of *Lecture Notes in Computer Science*, pages 134–195. Springer, 2007. ISBN 978-3-540-88642-6.

K. Rajalingham, D.R. Chadwick, and B. Knight. Classification of spreadsheet errors. *European Spreadsheet Risks Interest Group (EuSpRIG)*, 2001.

Christopher Scaffidi, Mary Shaw, and Brad Myers. Estimating the numbers of end users and end user programmers. *VLHCC '05: Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 207–214, 2005. doi: 10.1109/VLHCC.2005.34.

Sander Vermolen and Eelco Visser. Heterogeneous coupled evolution of software languages. In *MoDELS '08: Proceedings of the 11th international conference on Model Driven Engineering Languages and Systems*, pages 630–644, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-87874-2. doi: http://dx.doi.org/10.1007/978-3-540-87875-9_44.

Yingfei Xiong, Dongxi Liu, Zhenjiang Hu, Haiyan Zhao, Masato Takeichi, and Hong Mei. Towards automatic model synchronization from model transformations. In R. E. Kurt Stirewalt, Alexander Egyed, and Bernd Fischer, editors,

22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007), November 5-9, 2007, Atlanta, Georgia, USA, pages 164–173. ACM, 2007. ISBN 978-1-59593-882-4. URL `http://doi.acm.org/10.1145/1321631.1321657`.