Report from Dagstuhl Seminar 11031

Bidirectional Transformations "bx"

Edited by

Zhenjiang Hu[1], Andy Schürr[2], Perdita Stevens[3], and James Terwilliger[4]

1 **National Institute of Informatics - Tokyo, JP,** hu@nii.ac.jp
2 **TU Darmstadt, DE,** Andy.Schürr@es.tu-darmstadt.de
3 **University of Edinburgh, GB,** perdita@inf.ed.ac.uk
4 **Microsoft Corporation - Redmond, US,** james.terwilliger@microsoft.com

Abstract

Bidirectional transformations (*bx*) are a mechanism for maintaining the consistency of two (or more) related sources of information. Researchers from many different areas of computer science including databases (DB), graph transformations (GT), software engineering (SE), and programming languages (PL) are actively investigating the use of *bx* to solve a diverse set of problems. Although researchers have been actively working on bidirectional transformations in the above mentioned communities for many years already, there has been very little cross-discipline interaction and cooperation so far. The purpose of a first International Meeting on Bidirectional Transformations (GRACE-BX), held in December 2008 near Tokyo, was therefore to bring together international elites, promising young researchers, and leading practitioners to share problems, discuss solutions, and open a dialogue towards understanding the common underpinnings of *bx* in all these areas. While the GRACE-BX meeting provided a starting point for exchanging ideas in different communities and confirmed our believe that there is a considerable overlap of studied problems and developed solutions in the identified communities, the Dagstuhl Seminar 11031 on "Bidirectional Transformations" also aimed at providing a place for working together to define a common vocabulary of terms and desirable properties of bidirectional transformations, develop a suite of benchmarks, solve some challenging problems, and launch joint efforts to form a living *bx* community of cooperating experts across the identified subdisciplines. This report documents the program and the outcomes of Dagstuhl Seminar 11031 with abstracts of tutorials, working groups, and presentations on specific research topics.

Seminar 17.–21. January, 2011 – www.dagstuhl.de/11031
1998 ACM Subject Classification D.2 Software Engineering, D.3 Programming Languages, H.1 Models and Principles
Keywords and phrases Bidirectional Languages, Transformation, Model/Data Synchronisation
Digital Object Identifier 10.4230/DagRep.1.1.42
Edited in cooperation with Anthony Anjorin

Except where otherwise noted, content of this report is licensed under a Creative Commons BY-NC-ND 3.0 Unported license
Bidirectional Transformations "bx", *Dagstuhl Reports*, Vol. 1, Issue 1, pp. 42–67
Editors: Zhenjiang Hu, Andy Schürr, Perdita Stevens, and James Terwilliger

Dagstuhl Reports
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Executive Summary

Zhenjiang Hu
Andy Schürr
Perdita Stevens
James Terwilliger

License Creative Commons BY-NC-ND 3.0 Unported license
© Zhenjiang Hu, Andy Schürr, Perdita Stevens, James Terwilliger

This Dagstuhl Seminar was the second meeting bringing together 39 researchers from 13 countries across disciplines that study bidirectional transformations. The first one was the GRACE International Meeting on Bidirectional Transformations held in December 2008 near Tokyo, Japan [1]. The GRACE meeting consisted of short introductions from each of the participants on their background and work, followed by some longer presentations and demonstrations on some representative technologies from each field, concluding in some open discussion time. A primary takeaway from the GRACE meeting was an opportunity for each discipline to get some initial exposure to each other.

bx Dagstuhl Seminar 11031 Programme, Variant 2

	Monday	Tuesday	Wednesday	Thursday	Friday
09:00-10:30	Seminar Opening Round of Introductions	Basic DB Tutorial, P. 2 WG Proposals	Working Groups	Working Groups	4 Position Statements WG Report
10:30-11:00	Coffee Break				
11:00-12:00	General bx Intro Basic PL Tutorial	Basic GT Tutorial, P. 1	6 Position Statements	6 Position Statements	Closing Session
12:15-14:00	Lunch				
14:00-15:30	Basic PL Tutorial, Part 2	Basic GT Tutorial, P. 2 Basic SE Tutorial, P. 1	Excursion to Trier Dinner	WG Reports 3 Position Statements	
15:30-16:00	Coffee Break			Coffee Break	
16:00-17:30	Basic DB Tutorial Part 1	Basic SE Tutorial, P. 2		Working Groups	
18:00-.....	Dinner			Dinner	

Seminar Opening	Information about background, organizational matters, ...
Round of Introductions	about 1 minute per person without any slides
Basic Tutorials	Introduction to one subdomain; often combination of several tutorial proposals
WG Proposals	Presentation of ideas for working groups (up to 10 minutes)
Working Groups	Parallel working groups on different topics
WG Reports	Summaries of Working Group Activities
Position Statements	Short presentations of up to 10 minutes length

Figure 1 Time Table of Seminar - Updated Final Version

The Dagstuhl seminar intended to go a step further and begin to identify commonalities between the disciplines and start to set a cross-disciplinary research agenda. The first part of the seminar (cf. Fig 1 with a time table) consisted of tutorials from each of the four represented disciplines on the various bidirectional transformation solutions that field has to offer. The second part consisted of cross-disciplinary working groups dedicated to investigating specific examples of commonality between solutions or identifying requirements, terminology, or scenarios that may reach across fields. There were also a number of sessions reserved for participants to give position statements regarding their individual fields of research.

Participation at both the Dagstuhl and GRACE seminars came from four disciplines: (1) Programming Languages (PL), (2) Graph Transformations (GT), (3) Software Engineering

(SE), and (4) Databases (DB). Each of the first three disciplines made up about 2/7 of the participants, while databases took the remaining 1/7 out of about 39 participants. Representation from the database field was, nevertheless, an improvement over the turnout from the GRACE meeting.

Tutorials

The first part of the workshop was allocated for representatives from each of the four disciplines giving deep tutorials of about two hours length on the various solutions to bidirectional transformation problems offered by that discipline. In all cases these tutorials were presented by groups of scientists, who were responsible to offer the attendants of the other disciplines a basic introduction into their field as well as to highlight different aspects of ongoing research activities. The PL-related tutorial consisted of five parts presented by Nate Foster, Robert Glück, Zhenjiang Hu, Benjamin Pierce, and Janis Voigtländer. It contained a general introduction to the whole research area of bx. Furthermore, it addressed a large variety of different aspects ranging from specific types of bx programming languages on one hand to model/graph transformation approaches in the software-engineering world and lense-based view definition approaches in the database system world on the other hand. As a consequence this set of (sub-)tutorials also was responsible for highlighting already existing links between the existing bx subcommunities. The DB-tutorial offered by Jean-Luc Hainaut and James Terwilliger nicely complemented the preceding tutorial. A survey of various opportunities for using bx transformations in database engineering processes as well as for specifying views by means of bx technologies was presented. The GT-related tutorial afterwards given by Andy Schürr and Frank Hermann focused on one special category of bidirectional graph transformations. Its two parts dealt with practical as well as theoretical aspects of bx graph transformations. Finally, the SE-related tutorial of Krzysztof Czarnecki and Stephan Hildebrandt concluded the first 2.5 days long introduction to the field of bx with a discussion of model synchronization and incremental change propagation techniques from the point of view of the software engineering community. For a more detailed description of the four (sets of) tutorials the reader is referred to their abstracts collected in this Proceedings.

Working Groups

After the ground had been prepared by the above mentioned list of tutorials the participants had the opportunity to present specific research results and challenges in the form of 27 short presentations. Furthermore, six working groups were organized related to cross-disciplinary topics that were identified during the first days of the Seminar. The summaries of three of these working groups only finally made into the Proceedings, but all of them were very helpful to deepen our understanding of common challenges and triggered new research cooperations across the four disciplines. The list of addressed topics included aspects like the design of bx benchmarks, the identification of typical bx application scenarios and their requirements as well as options for the joint development of a new generation of bx languages that would incorporate and combine elements from solutions developed in different disciplines. The individual position statements covered a very broad spectrum of topics; they were used to intensify already started discussions of the first tutorial days and gave further input for the identification of new workshop topics. Again the reader is referred to the abstracts collected here for more details.

Summary and Future Work

We went into the seminar knowing that longer-term ideas like a common research agenda or a benchmark would take longer than a single week. The participants decided on several follow-up actions to keep work progressing:

- A follow-up meeting in the same style as the GRACE and Dagstuhl seminars to continue collaborating on a cross-disciplinary research agenda
- Workshops at conferences associated with each of the disciplines to work toward specific, targeted goals (a first one has already been scheduled associated with GTTSE 2011, and will focus on developing a benchmark[1]; a second follow-up event has just been accepted as a satellite workshop for ETAPS.)
- Tutorials and other assorted smaller, education-minded events at conferences to continue bringing awareness of bidirectional solutions from other disciplines, as well as awareness of the general BX effort
- Smaller-scale research cooperations that combine techniques from different fields like merging concepts from bidirectional programming languages and triple graph grammars as envisaged in one of the seminar's working groups.

In particular, one goal of the upcoming seminars and workshops is to increase participation from the database community. The bidirectional transformation problem has origins deep in the database community, but now has grown so that solutions are being driven from many different directions in different fields across computer science. The plan is to hold some of the tutorials or workshops at database venues to help solicit more ideas and opportunities for collaboration; details will be made available once they are scheduled.

References

1 K. Czarnecki, J. N. Foster, Z. Hu, R. Lämmel, Andy Schürr, and J. F. Terwilliger. Bidirectional Transformations: A Cross-Discipline Perspective.*ICMT 2009*, 260–283.

[1] http://www.di.univaq.it/CSXW2011/

2 Table of Contents

Executive Summary
Zhenjiang Hu, Andy Schürr, Perdita Stevens, James Terwilliger 43

Overview of Tutorials

Languages for Bidirectional Transformations
Nate Foster 49

Complement-Based Bidirectionalization
Janis Voigtländer 49

Trace-based Bidirectionalization
Zhenjiang Hu 49

Principles of Reversible Programming Languages
Robert Glück 50

Bidirectional Transformations in Database Research and Practice
James Terwilliger 50

The transformational approach to data-intensive system engineering and evolution
Jean-Luc Hainaut 51

Triple Graph Grammars in a Nutshell
Andy Schürr 51

Analysis of Model Transformations based on TGGs
Frank Hermann 52

Incremental Model Synchronization
Holger Giese 52

Model Synchronization: Theory and Practice
Krzysztof Czarnecki 52

Update propagation via tiles
Zinovy Diskin 53

Overview of Position Statements

Inconsistency detection and resolution in heterogenous model-based specifications using Maude
Artur Boronat 53

On the role of Triple Graph Grammars Concerning Requirements for Enterprise Modeling
Christoph Brandt 54

Co-evolving schemas and programs using coupled transformations
Anthony Cleve 54

Type-safe Evolution of Spreadsheets
Jacome Cunha 55

From State- to Delta-Based Bidirectional Model Transformations: Unweaving Alignment and Update Propagation
Zinovy Diskin 55

Propagation of Constraints along Model Transformations
Hartmut Ehrig 56

HOT Topics in Bidirectional Programming
Jeremy Gibbons 56

Lenses are Coalgebras for the Costate Comonad
Jeremy Gibbons 56

Lenses, Coalgebraically
Jeremy Gibbons 57

Direction Neutral Language Transformation with Metamodels
Martin Gogolla 57

Unified (Bidirectional) Transformation Language
Joel Greenyer 58

Model Integration and Synchronization
Frank Hermann 58

Bidirectional Graph Transformations based on Structural Recursion
Soichiro Hidaka 59

Incremental Bidirectional Model Synchronization
Stephan Hildebrandt 59

Symmetric lenses
Martin Hofmann 60

Triple Graph Grammars: Concepts, Extensions, Implementations, and Application Scenarios
Ekkart Kindler 60

Some challenges of integrating bidirectional transformation technologies
Ekkart Kindler 61

Towards Systematic Development of Bidirectional Transformations
Jochen M. Kuester 61

Right Inverses in Bidirectionalization
Kazutaka Matsuda 61

Bidirectional transformations and inter-modelling
Richard F. Paige 62

Bidirectional and change propagating transformations in MDE
Alfonso Pierantonio 63

Reversible Higher Order Pi Calculus
Alan Schmitt 63

Efficiency of Bidirectional Transformations
Janis Voigtländer 64

Change-based incremental updates
Meng Wang 64

Bx 'killer applications' in health care
Jens-Holger Weber-Jahnke 64

Fix Generation
Yingfei Xiong 65

A reversible programming language
Tetsuo Yokoyama 65

Overview of Working Groups

Model Consistency Management in Software Engineering
Krzysztof Czarnecki 65

Relationships between BX and View Updates
Soichiro Hidaka 66

Toward a Bidirectional Transformation Benchmark and Taxonomy
James Terwilliger 66

Participants 67

3 Overview of Tutorials

3.1 Languages for Bidirectional Transformations

Nate Foster (Cornell University, US)

License Creative Commons BY-NC-ND 3.0 Unported license
© Nate Foster

In recent years, a wide variety of approaches for describing bidirectional transformations have been proposed in the programming languages community. This multi-part tutorial surveyed recent work in this area and presented some of most promising approaches in detail. The first talk, by Nate Foster, introduced the key semantic issues and identified some connections to prior work on the view update problem in databases. The second talk, by Robert Glück, described *reversible languages*, which are languages in which every program denotes an injective function and can be effectively inverted. The third talk, by Janis Voigtländer, described a technique called *semantic bidirectionalization* that constructs a backward function from a polymorphic forward function, using parametricity and free theorems to prove well-behavedness. The fourth talk, by Benjamin Pierce, described *lens combinators* and focused on the use of type systems to establish well-behavedness as well as the issues surrounding the handling of ordered data. The final talk, by Zhenjiang Hu, presented a language for describing *bidirectional graph transformations* that uses trace information to guide the reverse transformation.

3.2 Complement-Based Bidirectionalization

Janis Voigtländer (Universität Bonn, DE)

License Creative Commons BY-NC-ND 3.0 Unported license
© Janis Voigtländer

This was part of the programming languages tutorial. The slides used are here:
http://www.dagstuhl.de/mat/Files/11/11031/11031.VoigtlaenderJanis1.Slides.pdf.

3.3 Trace-based Bidirectionalization

Zhenjiang Hu (NII - Tokyo, JP)

License Creative Commons BY-NC-ND 3.0 Unported license
© Zhenjiang Hu

Traces play an important role in bidirectional transformation, widely used for data exchanges between two databases, such as relational database and relational database, relational database and tree database, tree database and tree database, and even graph database and graph database. Traced-based bidirectionalization effectively handles updates on the view whose destination has origins (can be traced back) to updatable input data, by first checking whether the updates can be traced back to its origin and then propagating the updates to the source in a safe way. In this mini-tutorial, I explain the basic idea of trace-based bidirectionalization, and show how to compute traces effectively and how to propagate the view updates to the original database safely.

3.4 Principles of Reversible Programming Languages

Robert Glück (University of Copenhagen, DK)

License Creative Commons BY-NC-ND 3.0 Unported license
© Robert Glück

Reversible computing is the study of computing models that exhibit both forward and backward determinism. Understanding the fundamental properties of such models is not only naturally relevant for reversible programming, but has also been found important in bidirectional model transformation and program transformations such as inversion. The unique design features of these languages include explicit postcondition assertions, direct access to an inverse semantics (e.g. uncall), and the possibility of clean (garbage-free)computation of injective functions.

References

1 Yokoyama T., Axelsen H. B., Glück R., Principles of a reversible programming language. In: Conference on Computing Frontiers. 43-54, ACM 2008. http://doi.acm.org/10.1145/1366230.1366239

2 Axelsen H. B., Glück R., Yokoyama T., Reversible machine code and its abstract processor architecture. Lecture Notes in Computer Science, LNCS 4649, 2007. http://dx.doi.org/10.1007/978-3-540-74510-5_9

3.5 Bidirectional Transformations in Database Research and Practice

James Terwilliger (Microsoft Corporation - Redmond, US)

License Creative Commons BY-NC-ND 3.0 Unported license
© James Terwilliger

The primary tool for transformations in databases is the query.

Substantial research has been done on attempting to turn this basic tool into a bidirectional transformation. These attempts have developed a substantial amount of theoretical progress but hardly any uptake in practical applications.

This tutorial covered four research topics introduced in the past five years that provide new solutions with practical applications to the bidirectional transformation problem in databases. PRISM is a tool - with a formal predicate calculus backing - designed to handle the schema versioning problem in applications. Object-Relational Mappings are a way to bridge the traditional impedance mismatch between application and database, where an application must reliably be able to store and update data.

Guava is a framework that allows an application the freedom to query and update against a virtual schema, as well as evolve that schema, with a flexible mapping to its physical storage. Finally, the field of data exchange has invested research in being able to invert mappings expressed using predicate calculus.

3.6 The transformational approach to data-intensive system engineering and evolution

Jean-Luc Hainaut (University of Namur, BE)

License Creative Commons BY-NC-ND 3.0 Unported license
© Jean-Luc Hainaut
Joint work of Hainaut, Jean-Luc; Anthony, Cleve
Main reference Hainaut, J-L, The Transformational Approach to Database Engineering, in Lämmel, R., Saraiva, J., Visser, V., (Eds), Generative and Transformational Techniques in Software Engineering, pp. 95-143, LNCS 4143, Springer, 2006
URL http://dx.doi.org/10.1007/11877028_4

Transformation-based software engineering has long been considered a major scientific approach to build reliable and efficient programs. According to this approach, abstract specifications (or model) can be converted into correct, compilable and efficient programs by applying selected, correctness-preserving operators called transformations.

In the database engineering realm, an increasing number of bodies (e.g., OMG through the MDE proposal) and of authors recognize the merits of transformational approaches, that can produce in a systematic way machine-processable database structures from abstract models. Transformations that are proved to preserve the correctness of the source specifications have been proposed in virtually all the activities related to schema engineering: schema normalization, schema quality evaluation, logical design, schema integration, views derivation, schema equivalence, database migration, data conversion, reverse engineering, schema optimization, ETL, wrapper generation and others. The proposed tutorial addresses both basic and practical aspects of database transformation techniques. The concept of transformation is developed, together with its properties of semantics-preservation. Major database engineering activities are redefined in terms of transformation techniques, and the impact on CASE technology is discussed. We also show in this tutorial that schema transformations can be used as a formal basis for deriving data transformations as well as program transformation.

3.7 Triple Graph Grammars in a Nutshell

Andy Schürr (TU Darmstadt, DE)

License Creative Commons BY-NC-ND 3.0 Unported license
© Andy Schürr
Main reference F. Klar, M. Lauder, A. Königs, A. Schürr: "Extended Triple Graph Grammars with Efficient and Compatible Graph Translators", in: A. Schürr, C. Lewerentz, G. Engels, W. Schäfer, B. Westfechtel (eds.): Graph Transformations and Model Driven Enginering - Essays Dedicated to Manfred Nagl on the Occasion
URL http://dx.doi.org/10.1007/978-3-642-17322-6_8

The first part of the Triple Graph Grammar (TGG) tutorial is a gentle introduction to the basic ideas of and motivations for the development of a rule-based and declarative specification formalism that allows for the high-level description of functional and non-functional relationships between pairs of graphs (models). A family of graph (model) transformations is derived from such a TGG specification that supports batch transformation and incremental change propagation scenarios in both directions as well as checking the consistency of given pairs of models or graphs. Traceability relationships between elements of related pairs of graphs are created and updated as a side effect. The presentation prepares the ground for the 2nd part of the tutorial which sketches the formal background of TGGs

based on category theory and presents techniques for the verification of important properties of TGGs and derived graph (model) translators.

3.8 Analysis of Model Transformations based on TGGs

Frank Hermann (TU Berlin, DE)

License Creative Commons BY-NC-ND 3.0 Unported license
© Frank Hermann
Main reference Frank Hermann, Hartmut Ehrig, Ulrike Golas, and Fernando Orejas: Efficient analysis and execution of correct and complete model transformations based on triple graph grammars. In: Proc. MDI '10, Jean Bezivin, Richard Mark Soley, and Antonio Vallecillo (Eds.), ACM 2010.
URL http://dx.doi.org/10.1145/1866272.1866277

Triple graph grammars (TGGs), introduced by Schürr in 1994, are a formal and intuitive concept for the specification of bidirectional model transformations. The key advantages of TGGs are the automatic derivation of operational rules, which simplifies specification, and powerful analysis capabilities based on the underlying formal foundation. This tutorial presents several automated analysis techniques concerning important properties of model transformations based on TGGs. In the first part, we present execution formalisms, which closely correspond to available implementations and ensure syntactical correctness and completeness of model transformations. Moreover, termination of the execution is guaranteed by static checks of non-restrictive conditions.

In the second part, we present how functional behaviour and information preservation of model transformations are checked by automated techniques. These techniques are additionally used for detection and resolution of conflicts between model transformation rules as well as for improving efficiency of the execution.

3.9 Incremental Model Synchronization

Holger Giese (Hasso-Plattner-Institut GmbH, DE)

License Creative Commons BY-NC-ND 3.0 Unported license
© Holger Giese

In system development the size of models and their proper synchronization can become a major problem. In this tutorial we will discuss what is theoretically wanted for model synchronization and what additional needs and limitations come into play in practice. The problem and existing solutions that extend triple graph grammars are outlined using an example from an industry project on model synchronization.

3.10 Model Synchronization: Theory and Practice

Krzysztof Czarnecki (University of Waterloo, CA)

License Creative Commons BY-NC-ND 3.0 Unported license
© Krzysztof Czarnecki

Model-driven engineering often requires many overlapping models of a system, each supporting a particular kind of stakeholder or task. The consistency among these models needs to be

managed during system development. Consistency management occurs in the space of multiple model replicas, versions over time, and different modeling languages, which make the process complex and challenging. In this tutorial, I will introduce the key concepts and mechanisms of consistency management, including mapping definitions to specify overlap, model alignment to establish differences, reconciliation to resolve conflicts, and change propagation to re-establish consistency. Most importantly, I show how bidirectional transformation using asymmetric and symmetric lenses used to check consistency and to propagate changes. I will illustrate these concepts and mechanisms with practical examples of model synchronization, including synchronizing architectural models and code, business process specifications and implementations, and structural and behavioral models in UML.

3.11 Update propagation via tiles

Zinovy Diskin (University of Waterloo, CA)

License Creative Commons BY-NC-ND 3.0 Unported license
© Zinovy Diskin
Main reference Diskin, Z., Model synchronization: Mappings, Tiles, and Categories. In Generative and Transformational Techniques in Software Engineering III, Springer LNCS'6491/2011, 92-165
URL http://dx.doi.org/10.1007/978-3-642-18023-1_3

The tutorial presents a novel algebraic framework for specifying architectures of model synchronization tools. The basic premise is that synchronization procedures, and hence algebraic operations modeling them, are *diagrammatic*: they take a configuration (diagram) of models and mappings as their input and produce a diagram as output. Many important synchronization scenarios are based on diagram operations of square shape. Composition of such operations amounts to their *tiling*, and complex synchronizers can thus be assembled by tiling together simple synchronization blocks. This gives rise to a visually suggestive yet precise notation for specifying synchronization procedures and reasoning about them. And the last but not least, specification and design with tiles are enjoyable.

Details can be found in my paper *Model synchronization: Mappings, Tiles, and Categories. In GTTSE'2009, Springer LNCS'6491/2011, pp. 92-165.*

4 Overview of Position Statements

4.1 Inconsistency detection and resolution in heterogenous model-based specifications using Maude

Artur Boronat (University of Leicester, GB)

License Creative Commons BY-NC-ND 3.0 Unported license
© Artur Boronat
Joint work of Boronat, Artur; Meseguer, José
Main reference A. Boronat, J. Meseguer: Automated Model Synchronization: a Case Study on UML with Maude. Tenth International Workshop on Graph Transformation and Visual Modeling Techniques. Collocated with ETAPS 2011. Saarbrücken (Germany). April 2-3, 2011. Pre-Proceedings.

Software-intensive systems comprise a wide range of heterogeneous software artefacts, which are usually developed by distributed teams. In model-based development, these software artefacts are given as models, abstracting relevant features of a system from implementation details, enhancing development, maintenance and evolution processes. In heterogeneous

software specifications, these models describe parts of the same system from different points of view and from different levels of abstraction, hence overlapping in various ways. Parallel updates in different models may result in unanticipated inconsistencies in the overlapped part. An approach based on rewriting logic for detecting inconsistencies in heterogeneous model-based specifications and to synchronise their constituent models will be presented. In this approach, strategies are used to guide the synchronisation process in an optimal way, filtering out many non-desirable solutions.

4.2 On the role of Triple Graph Grammars Concerning Requirements for Enterprise Modeling

Christoph Brandt (University of Luxembourg, LU)

License Creative Commons BY-NC-ND 3.0 Unported license
© Christoph Brandt
Joint work of Brandt, Christoph; Hermann, Frank
Main reference Christoph Brandt and Frank Hermann, How Far Can Enterprise Modeling for Banking Be Supported by Graph Transformation?, ICGT, Springer, LNCS, 6372, 2010, 3-26
URL http://dx.doi.org/10.1007/978-3-642-15928-2_2

Reconstructed requirements for enterprise modeling are presented that came up during an ongoing evaluation of enterprise modeling practices at Credit Suisse. They will help to reorganize, reframe and reconstruct today's modeling approaches leading to better results built on top of sound formal techniques. First investigations show that triple graph grammars are suitable to provide solutions for several of these requirements.

4.3 Co-evolving schemas and programs using coupled transformations

Anthony Cleve (University of Namur, BE)

License Creative Commons BY-NC-ND 3.0 Unported license
© Anthony Cleve
Joint work of Cleve, Anthony; Hainaut, Jean-Luc
Main reference Anthony Cleve and Jean-Luc Hainaut,Co-transformations in database applications evolution. In Ralf Lämmel, João Saraiva, and Joost Visser, editors, Generative and Transformational Techniques in Software Engineering, volume 4143 of Lecture Notes in Computer Science, pages 409421. Springer, 2006.
URL http://dx.doi.org/10.1007/11877028_17

In this short position statement we will elaborate on the use of coupled transformations to support the co-evolution of inter-dependent artefacts in data-intensive systems. We will particularly focus on the co-evolution of database schema and programs in such scenarios as schema refactoring, database migration and database design.

4.4 Type-safe Evolution of Spreadsheets

Jacome Cunha (Universidade de Minho - Braga, PT)

License Creative Commons BY-NC-ND 3.0 Unported license
© Jacome Cunha
Main reference Jãcome Cunha, Joost Visser, Tiago Alves, and João Saraiva. Type-safe evolution of spreadsheets. In FASE '11: Proc. of the 13th Int'l Conf. on Fundamental Approaches to Software Engineering: Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011.
URL http://alfa.di.uminho.pt/ jacome/down/fase11.pdf

Spreadsheets are notoriously error-prone. To help avoid the introduction of errors when changing spreadsheets, models that capture the structure and interdependencies of spreadsheets at a conceptual level have been proposed. Thus, spreadsheet evolution can be made safe within the confines of a model. As in any other model/instance setting, evolution may not only require changes at the instance level but also at the model level. When model changes are required, the safety of instance evolution can not be guarded by the model alone.

We have designed an appropriate representation of spreadsheet models, including the fundamental notions of formula and references. For these models and their instances, we have designed coupled transformation rules that cover specific spreadsheet evolution steps, such as the insertion of columns in all occurrences of a repeated block of cells. Each model-level transformation rule is coupled with instance level migration rules from the source to the target model and vice versa. These coupled rules can be composed to create compound transformations at the model level inducing compound transformations at the instance level. This approach guarantees safe evolution of spreadsheets even when models change.

4.5 From State- to Delta-Based Bidirectional Model Transformations: Unweaving Alignment and Update Propagation

Zinovy Diskin (University of Waterloo, CA)

License Creative Commons BY-NC-ND 3.0 Unported license
© Zinovy Diskin
Joint work of Diskin, Zinovy; Xiong, Yingfei; Czarnecki, Krzysztof
Main reference Diskin, Zinovy; Xiong, Yingfei; Czarnecki, Krzysztof. From State- to Delta-Based Bidirectional Model Transformations. Theory and Practice of Model Transformations, Third International Conference, ICMT 2010. Proceedings
URL http://dx.doi.org/10.1007/978-3-642-13688-7

Existing bidirectional model transformation languages are mainly state- based: model alignment is hidden inside update propagating procedures, and model deltas are implicit. Weaving alignment with update propagation complicates the latter and makes it less predictable and less manageable.

We propose to separate concerns and consider two distinct operations: delta discovery and delta propagation. This architecture has several technological advantages, but requires a corresponding theoretical support.

We present an algebraic framework of delta lenses, and discuss an emerging landscape of delta-based model synchronization. An essentially updated version of our ICMT paper is submitted to JOT.

4.6 Propagation of Constraints along Model Transformations

Hartmut Ehrig (TU Berlin, DE)

License Creative Commons BY-NC-ND 3.0 Unported license
© Hartmut Ehrig
Main reference Hartmut Ehrig, Frank Hermann, Hanna Schölzel and Christoph Brandt: Propagation of Constraints along Model Transformations Based on Triple Graph Grammars. In: Proc. Graph Transformation and Visual Modeling Techniques (GT-VMT 2011), EC-EASST (To appear 2011).

Within the approach of model transformations based on TGGs, it is an interesting problem to transform not only source language to target language models, but to transform also properties of models given by graph constraints. This technique is called propagation of constraints and should satisfy the following property: If the source model satisfies the source constraint, then also the target model should satisfy the propagated target constraint. First investigations show that it is useful to construct first a propagated integrated constraint, which is satisfied by the integrated model obtained as an intermediate step of the model transformation.

4.7 HOT Topics in Bidirectional Programming

Jeremy Gibbons (University of Oxford, GB)

License Creative Commons BY-NC-ND 3.0 Unported license
© Jeremy Gibbons
URL http://www.dagstuhl.de/mat/Files/11/11031/11031.GibbonsJeremy1.Slides.pdf

There is a lot of interesting structure in bidirectional programming problems - structure that should be exploited in solving those problems. The kinds of structure I have in mind are higher-order and typed techniques from PL theory - what Bob Harper called "HOT" topics; Janis Voigtländer's work on exploiting parametricity is a good example. Other techniques that look promising are higher-order representations, indexed- and dependently-typed programming, and relational algebra. Perhaps it is evidence of my own ignorance, but it seems to me that this rich structure is not currently being exploited to the full. I will explain why I think HOT techniques show promise, and why I hope that they will lead to a more semantic rather than syntactic approach to bidirectional programming.

4.8 Lenses are Coalgebras for the Costate Comonad

Jeremy Gibbons (University of Oxford, GB)

License Creative Commons BY-NC-ND 3.0 Unported license
© Jeremy Gibbons
URL http://www.dagstuhl.de/mat/Files/11/11031/11031.GibbonsJeremy2.ExtAbstract.pdf

The three laws of a "very well-behaved lens" are exactly the condition that the lens (that is, the pairing of its get and put functions) is a coalgebra for the costate comonad. This is my explanation of an observation due to Russell O'Connor.

4.9 Lenses, Coalgebraically

Jeremy Gibbons (University of Oxford, GB)

License Creative Commons BY-NC-ND 3.0 Unported license
© Jeremy Gibbons
Main reference Submitted for publication
URL http://web.comlab.ox.ac.uk/publications/publication4599-abstract.html

Lenses are a heavily studied form of bidirectional transformation with diverse applications including database view updating, software development and memory management. Recent work has explored lenses category theoretically, and established that the category of lenses for a fixed "view" V is, up to isomorphism, the category of algebras for a particular monad on Set/V. In this paper we show that in addition lenses are the coalgebras for the comonad generated by the cartesian closure adjunction on Set. We present a fully constructive proof of the coalgebra correspondence, we note that the algebra correspondence extends to arbitrary categories with products and that the coalgebra correspondence extends to arbitrary cartesian closed categories, and we show that both correspondences extend to isomorphisms of categories. The resulting isomorphism between a category of algebras and a category of coalgebras is unexpected, and we analyze it isolating its underlying generality, and also the particularity that restricts its applicability. We end with remarks about the utility of the two different treatments of lenses,especially for obtaining further, more realistic, generalizations of the notion of lens.

4.10 Direction Neutral Language Transformation with Metamodels

Martin Gogolla (Universität Bremen, DE)

License Creative Commons BY-NC-ND 3.0 Unported license
© Martin Gogolla

The aim of this work is to sketch a general metamodel-based frame for describing potentially bidirectional transformations between software languages. We propose to describe a single language with a metamodel consisting of a UML class diagram with classes, attributes and associations and accompanying OCL constraints. A language description is separated into a syntax and a semantics part. The allowed object diagrams of the syntax part correspond to the syntactically allowed words of the language. The semantics can associate with every word of the language certain semantical objects. However, only finite fractions of the semantics can be handled in metamodel tools. Having two languages described by their metamodels, a transformation between them is established by another metamodel, a transformation model. The transformation model can associate a syntax object from one language with a syntax object from the other language in a direction neutral way and opens the possibility for bidirectionality. Analogously, semantical objects can be connected. Transformation properties like ‘equivalence’ or ‘embedding’ can and must be expressed with constraints. Thus, the approach describes syntax and semantics of the languages, their transformation and their properties in a uniform way by means of metamodels.

4.11 Unified (Bidirectional) Transformation Language

Joel Greenyer (Universität Paderborn, DE)

License Creative Commons BY-NC-ND 3.0 Unported license
© Joel Greenyer
Main reference J. Greenyer and E. Kindler: Comparing relational model transformation technologies: implementing Query/View/Transformation with Triple Graph Grammars. Software and Systems Modeling (SoSyM) 2010, Vol. 9, 21–46, 1
URL http://dx.doi.org/10.1007/s10270-009-0121-8

In the recent past, the declarative, relational transformation languages QVT-Relations (part of the OMG's Query/View/Transformation standard) and TGGs (Triple Graph Grammars) have gained in popularity in the domain of model-based software engineering. That is partly because many transformation problems can be described very conveniently with these languages, and because, in principle, the same set of transformation rules can be interpreted both in a forward and backward transformation direction. Even though the principles of QVT-Relations and TGGs are different (specifying relations between model patterns vs. rules for producing valid corresponding graphs/models), the languages are very similar [GK10]. However, neither TGGs nor QVT-Relations have yet found a broad application in industry. Over the past years, we have developed a transformation tool for TGGs, called the TGG-Interpreter (http://www.cs.uni-paderborn.de/index.php?id=tgg-interpreter&L=1), and we have gained experience in a wide variety of transformation examples. From this experience, we identified some requirements that are crucial for a successful application of TGGs (and also QVT-Relations) in practice. These requirements ranges from techniques for analyzing for example the confluence and bi-directionality of a TGG to support for testing, debugging, to an extensible and maintainable transformation engine architecture. Especially, we feel that OCL, as it is used in QVT-Relations as well as TGGs in the TGG-Interpreter today, is insufficient for specifying attribute constraints in bi-directional transformations. Therefore, it should be investigated how to integrate TGGs (and QVT-Relations) with other bidirectional model transformation approaches, for example reversible programming, in the future.

4.12 Model Integration and Synchronization

Frank Hermann (TU Berlin, DE)

License Creative Commons BY-NC-ND 3.0 Unported license
© Frank Hermann
Main reference Hartmut Ehrig, Karsten Ehrig, Frank Hermann: From Model Transformation to Model Integration based on the Algebraic Approach to Triple Graph Grammars. ECEASST Vol. 10, EASST 2008.
URL http://journal.ub.tu-berlin.de/index.php/eceasst/article/view/154

Within the approach of model transformations based on TGGs, model integration and synchronization are important problems. Let us call a source and a target model consistent, if there exists an integrated model generated by a given TGG, such that the source and target components of the integrated model are equal to the given models. The integration problem is the following: Find out whether, for a given source and target model, both models are consistent and construct an integrated model in case of consistency. If both models are not consistent, the synchronization problem is to modify the source and/or the target model, such that both become consistent. If the modification of both models is allowed, then the problem is to find a synchronization, such that the consistent modified models can be constructed from the given ones with a "minimal number of changes". In contrast to several

approaches in the literature, we are not so much interested in efficient heuristic, but more in correct formal solutions.

4.13 Bidirectional Graph Transformations based on Structural Recursion

Soichiro Hidaka (NII - Tokyo, JP)

License Creative Commons BY-NC-ND 3.0 Unported license
© Soichiro Hidaka
Joint work of Hu, Zhenjiang; Inaba, Kazuhiro; Kato, Hiroyuki; Matsuda, Kazutaka; Nakano, Keisuke;
Main reference Soichiro Hidaka, Zhenjiang Hu, Kazuhiro Inaba, Hiroyuki Kato, Kazutaka Matsuda, and Keisuke Nakano. 2010. Bidirectionalizing graph transformations. In Proceedings of the 15th ACM SIGPLAN international conference on Functional programming (ICFP '10). ACM, New York, NY, USA, 205-216.
URL http://doi.acm.org/10.1145/1863543.1863573

We have been challenging bidirectional transformations problems within the context of graphs, by proposing a formal definition of a well-behaved bidirectional semantics for UnCAL, i.e., a graph algebra for the known UnQL graph query language. We utilize both the recursive and bulk semantics of structural recursion on graphs.

Existing forward evaluation of structural recursion has been refined so that it can produce sufficient trace information for later backward evaluation. In this position statement I will introduce our framework of bidirectional graph transformations and how they are implemented based on structural recursion with trace information, followed by optimization opportunities and remaining challenges.

4.14 Incremental Bidirectional Model Synchronization

Stephan Hildebrandt (Hasso Plattner Institut - Potsdam, DE)

License Creative Commons BY-NC-ND 3.0 Unported license
© Stephan Hildebrandt
Main reference Giese, Holger and Neumann, Stefan and Hildebrandt, Stephan; Model Synchronization at Work: Keeping SysML and AUTOSAR Models Consistent; Graph Transformations and Model Driven Enginering - Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday; LNCS Volume 5765; Pages 555-579
URL http://dx.doi.org/10.1007/978-3-642-17322-6_24

Pure model transformations are not sufficient to cope with concurrent changes, different information details in different modeling languages, and increasing model sizes in practical software engineering. Therefore, model synchronization technologies are required that only propagate changes between models. This avoids overwriting additional information in target models and can be executed faster than a complete model transformation. This talk presents MoTE, an incremental bidirectional model synchronization system based on triple graph grammars. Two case studies are presented, where the system has been successfully put to practice. Some open issues are discussed that were encountered in the case studies.

4.15 Symmetric lenses

Martin Hofmann (LMU München, DE)

License Creative Commons BY-NC-ND 3.0 Unported license
© Martin Hofmann
Main reference Martin Hofmann, Benjamin C. Pierce, Daniel Wagner: Symmetric lenses. Proc. ACM Symp. POPL 2011: 371-384, ACM Press, 2011.

Symmetric lenses have been proposed by Pierce, Wagner, and myself as a completely symmetric generalisation of lenses which in turn are a category-theoretic abstraction of a pair of a view extraction and update functions. This position statement gives a taste of definition, basic category-theoretic properties, datatypes and iterators, as well as a representation theorem. Details in our paper at POPL2011.

4.16 Triple Graph Grammars: Concepts, Extensions, Implementations, and Application Scenarios

Ekkart Kindler (Technical University of Denmark, DK)

License Creative Commons BY-NC-ND 3.0 Unported license
© Ekkart Kindler
Main reference Technical Report tr-ri-07-284, Software Engineering Group, Department of Computer Science, University of Paderborn, June 2007.
URL http://www2.cs.uni-paderborn.de/cs/ag-schaefer/Veroeffentlichungen/Quellen/Papers/2007/tr-ri-07-284.pdf

Triple Graph Grammars (TGGs) are a technique for defining the correspondence between two different types of models in a declarative way. The power of TGGs comes from the fact that the relation between the two models can not only be defined, but the definition can be made operational so that one model can be transformed into the other in either direction; even more, TGGs can be used to synchronize and to maintain the correspondence of the two models, even if both of them are changed independently of each other; i. e., TGGs work incrementally.

TGGs have been introduced more than 10 years ago by Andy Schürr.

Since that time, there have been many different applications of TGGs for transforming models and for maintaining the correspondence between these models. To date, there have been several modifications, generalizations, extensions, and variations. Moreover, there are different approaches for implementing the actual transformations and synchronizations of models.

In this paper, we present the essential concepts of TGGs, their spirit, their purpose, and their fields of application. We also discuss some of the extensions along with some of the inherent design decisions, as well as their benefits and caveats. All these are based on several year's of experience of using TGGs in different projects in different application areas.

4.17 Some challenges of integrating bidirectional transformation technologies

Ekkart Kindler (Technical University of Denmark, DK)

License Creative Commons BY-NC-ND 3.0 Unported license
© Ekkart Kindler

There are different kinds of uni- and bi-directional transformation mechanisms, which are based on different philosophies, paradigms, and principles. In concrete situations, one often would like to combine different technologies in order to achieve certain goals; but this is not possible for fundamental principal differences, some conceptual differences, or simply for technical reasons. This position statement heads at identifying some issues and triggering the work on a clear conceptual framework and interfaces that would allow comparing and integrating different transformation technologies.

4.18 Towards Systematic Development of Bidirectional Transformations

Jochen M. Kuester (IBM Research Zürich - Rüschlikon, CH)

License Creative Commons BY-NC-ND 3.0 Unported license
© Jochen M. Kuester

In order to adopt bidirectional model transformations in industry, an approach for their systematic development is required. Such an approach includes methods and tools for requirements specification, design and implementation of bidirectional model transformations.

In this presentation, we first report on lessons learnt from the development of a solution based on unidirectional model transformations and then establish several requirements for systematic development of bidirectional model transformations.

4.19 Right Inverses in Bidirectionalization

Kazutaka Matsuda (Tohoku University, JP)

License Creative Commons BY-NC-ND 3.0 Unported license
© Kazutaka Matsuda

Bidirectionalization is a program transformation that derives a bidirectional transformation from a unidirectional transformation automatically or manually.

We emphasize on the importance of right-inverse construction in syntactic bidirectionalization. Right inverses are useful not only for construction of so-called "create" functions but also for construction of bidirectional transformations whose behavior can be controlled by users.

4.20 Bidirectional transformations and inter-modelling

Richard F. Paige (University of York, GB)

License Creative Commons BY-NC-ND 3.0 Unported license
© Richard F. Paige
Main reference Esther Guerra, Juan de Lara, Dimitrios S. Kolovos, Richard F. Paige: Inter-modelling: From Theory to Practice. MoDELS (1) 2010: 376-391, LNCS, Springer-Verlag.
URL http://dx.doi.org/10.1007/978-3-642-16145-2_26

We have been developing, implementing and deploying model management tools for a number of years. Model management focuses on manipulating models once they have been constructed. Typical model management operations include model transformation (including model-to-model, model-to-text, text-to-model and update-in-place transformations), model merging, model comparison, model validation (e.g., inter-model consistency checking), and model migration. We have provided tool support for these operations in the Epsilon model management framework , which provides a suite of interoperable task-specific languages. We have also used this framework and these languages in a large number of industrial applications of MDE; some of these applications involved applying several of the languages in concert. As a result, we have improved our framework, increased our understanding of what makes MDE challenging and tractable, and what are some of the obstacles impeding its uptake.

A number of our applications of MDE have involved model transformation (of various flavours), and some of these applications have either included, or generated, requirements for some kind of support for bidirectionality. Bidirectionality, when it appeared in requirements, was usually associated with specific use cases; moreover, when investigating the requirement more carefully, it was often the case that a bidirectional transformation was only one of a number of possible ways in which the requirement could be supported.

As a result of our work on practical applications of MDE, we have determined that we have more questions than answers about bidirectional transformation. Some of these questions are as follows.

- What is the problem engineers are trying to solve with a bidirectional transformation (bx)?
- When is a bx a good solution to a problem, and when is it a bad solution?
- More specifically, what is the sweet spot in choosing whether to use a bx or to use more traditional MDE approaches (unidirectional transformation, inter-model consistency, etc)?
- What are scenarios in which bx applies?
- What patterns appear in a bx?
- Are there sufficient recurring patterns in a bx that suggests that we need dedicated languages for programming bidirectional transformations?

We have been thinking, initially, about the first question: what problem are engineers trying to solve? Fundamental in MDE is the notion of inter-modelling: you build different models (some of which are of languages or domains, i.e., metamodels) that are inter-related in various ways. The inter-relationships between models are interesting and diverse and come in many different flavours, but they are typically implemented as trace-links (or traceability links). Some example inter-model relationships include: simple dependencies, code generation, consistency, bx, synchronisation, etc. It is possible to define very strange inter-model relationships, including security and privacy relationships, for example: users with a particular role cannot see both X and Y. It is also possible to define very complicated inter-model relationships, e.g., between a PDF document and an EJB implementation where parts of the PDF document define concepts that are designed and implemented in the EJBs. This latter relationship is not really something that is a transformation problem, nor is it

really a consistency problem, though conceivably parts of the problem could be captured in suitable transformation and consistency tools. Overall, we think that the richness of inter-modelling relationships (and types of trace-links that can implement these relationships) illustrates the challenges that we have in understanding where bx is really useful and usable.

It may be helpful to have a unified conceptual model of MDE like that of inter-modelling, in order to better understand where bx are useful, where (for example) approaches like inter-model consistency checking are suitable, and what benefits can be obtained through bidirectional transformations directly. At the moment, it seems that there is much confusion in the bx-MDE field, possibly because we are not yet clear what are the different kinds of relationships there are that we want to construct between our models, and how they should best be constructed.

4.21 Bidirectional and change propagating transformations in MDE

Alfonso Pierantonio (Univ. degli Studi di L'Aquila, IT)

License Creative Commons BY-NC-ND 3.0 Unported license
© Alfonso Pierantonio
Main reference A. Cicchetti, D. Di Ruscio, R. Eramo and A. Pierantonio, JTL: a bidirectional and change propagating transformation language, 3rd International Conference on Software Language Engineering (SLE 2010), Eindhoven (The Netherlands)
URL http://dx.doi.org/10.1007/978-3-642-19440-5_11

In Model Driven Engineering bidirectional transformations are considered a core ingredient for managing both the consistency and synchronization of two or more related models. However, while non-bijectivity in bidirectional transformations is considered relevant, current languages still lack of a common understanding of its semantic implications hampering their applicability in practice. In this work, the Janus Transformation Language (JTL) is presented, a bidirectional model transformation language specifically designed to support non bijective transformations and change propagation. In particular, the language propagates changes occurring in a model to one or more related models according to the specified transformation regardless of the transformation direction. Additionally, whenever manual modifications let a model be non reachable anymore by a transformation, the closest model which approximate the ideal source one is inferred. The language semantics is also presented and its expressivity and applicability are validated against a reference benchmark. JTL is embedded in a framework available on the Eclipse platform which aims to facilitate the use of the approach, especially in the definition of model transformations.

4.22 Reversible Higher Order Pi Calculus

Alan Schmitt (INRIA Rhône-Alpes, FR)

License Creative Commons BY-NC-ND 3.0 Unported license
© Alan Schmitt

Reversible HOPi is a higher-order process calculus where every action can be undone. We are developing this calculus as a foundation for dependable distributed application. We will give an intuition as how this approach is different from sequential reversible languages, and how it may be extended with controlled rollback to better control when undoing occurs.

4.23 Efficiency of Bidirectional Transformations

Janis Voigtländer (Universität Bonn, DE)

License Creative Commons BY-NC-ND 3.0 Unported license
© Janis Voigtländer

I raise some questions related to the efficiency of programs coming out of linguistic approaches to bidirectional transformation. Do we even care (yet) about the efficiency of these programs, or are we still at a point where we have to struggle too much even to get their semantic behaviour right? If we do care about efficiency, what are the criteria/setups to measure? Can we hope to rely on standard program transformations (from the functional languages community) to improve efficiency, or do we have to invent new techniques tailored to bidirectionalization or specific DSLs?

The slides used are here:
http://www.dagstuhl.de/mat/Files/11/11031/11031.VoigtlaenderJanis.Slides.pdf.

4.24 Change-based incremental updates

Meng Wang (University of Oxford, GB)

License Creative Commons BY-NC-ND 3.0 Unported license
© Meng Wang
Joint work of Wang, Meng; Gibbons, Jeremy

To handle large data that are subject to relatively small modifications, incrementality of updates is the key. Incrementality has been explored in the settings of relational databases and graph transformations, where the data is typically not typed (in the sense not restricted to a particular shape). This flexibility in structure makes it relatively easy to divide the data into separate parts that can be transformed and updated independently. The same is not true if the data is to be encoded with more general purpose algebraic datatypes, with transformations defined as functions: dividing data into well-typed separate parts is tricky, and recursions typically create interdependencies. In this work, we look at the identification of transformations that support incremental updates, and device a constructive process to achieve it.

4.25 Bx 'killer applications' in health care

Jens-Holger Weber-Jahnke (University of Victoria, CA)

License Creative Commons BY-NC-ND 3.0 Unported license
© Jens-Holger Weber-Jahnke

Nations around the world are creating infrastructures for maintaining electronic health records (EHRs) at massive scale. These infrastructures need to translate between a large number of heterogenous data bases. The richness and sensitivity of medical data provide new challenges and opportunities for the bx transformation research community. Current transformation technologies applied in the eHealth sector are immature and lack fundamentally important properties. This results not only in cost overruns of EHR infrastructure projects, but also jeopardizes patient safety. My presentation has introduced typical variations of EHR

infrastructures and motivated fundamental required properties of bx approaches suitable for this domain.

4.26 Fix Generation

Yingfei Xiong (University of Waterloo, CA)

License Creative Commons BY-NC-ND 3.0 Unported license
© Yingfei Xiong

Bidirectional transformation maintains consistency between two pieces of data automatically. However, consistency cannot always be maintained automatically, and existing tools present users with a list of choices of modifying data, called "fixes". Fixes are widely used in many areas, but there is little support for fix generation in general. I will show that fix generation can be formalized in a similar way to BX, as two functions and laws governing their behavior, and ideas in BX could be used as a starting point to develop general support for fix generation.

4.27 A reversible programming language

Tetsuo Yokoyama (Nanzan University, JP)

License Creative Commons BY-NC-ND 3.0 Unported license
© Tetsuo Yokoyama

Reversible programming languages, which I focus on, are imposed restrictions in a way that no program loses information and thus the reconstruction of the previous states is always possible. In common with other programming paradigms, reversible programming has its own programming methodology. We introduced Janus, a high-level reversible language, and discussed the features of the languages such as cleanliness, structured programming and r-Turing completeness, and a guideline for the systematic development of efficient reversible programming.

5 Overview of Working Groups

5.1 Model Consistency Management in Software Engineering

Krzysztof Czarnecki (University of Waterloo, CA)

License Creative Commons BY-NC-ND 3.0 Unported license
© Krzysztof Czarnecki
Joint work of Czarnecki, Krzysztof; Diskin, Zinovy; Antkiewicz, Michal; Xiong, Yingfei
URL http://gsd.uwaterloo.ca/publications

Model-driven engineering often requires many overlapping models of a system, each supporting a particular kind of stakeholder or task. The consistency among these models needs to be managed during system development. Consistency management occurs in the space of multiple model replicas, versions over time, and different modeling languages, which make the process complex and challenging. In this tutorial, I will give an overview of the main consistency management scenarios, such as involving single model vs. N models, models

in the same language or in different languages, and precise automated alignment or semi-automated one, introduce the key concepts, and establish some basic terminology of model synchronization.

5.2 Relationships between BX and View Updates

Soichiro Hidaka (NII - Tokyo, JP)

License Creative Commons BY-NC-ND 3.0 Unported license
© Soichiro Hidaka

In bidirectionalization, which derives backward transformation from forward transformation, different forward transformations that can produce identical result may lead to different acceptance of the modifications on the view. For example, in the author's framework of bidirectional graph transformation, variable references to the entire source can accept any modification while copying transformation by replacing edge by edge with pattern matching, accepts limited editing operation in case of edge renaming.

In this working group we discussed the impact of the description of forward transformation on the accepted view updates using concrete example like a duplication function. Elements that are likely to make backward transformation harder are enumerated.

In summary, tight coupling of trace information and syntactic definition of the transformation is considered to be the source of the problem. Reflecting research subjects of the participants, concepts that play a role similar to trace like complement and postcondition had also been discussed. Decoupling, abstract postconditions or related concepts like contracts, could be investigated as a future direction.

Participants: Robert Glück, Soichiro Hidaka, Michael Johnson, Kazutaka Matsuda, Meng Wang, Tetsuo Yokoyama

5.3 Toward a Bidirectional Transformation Benchmark and Taxonomy

James Terwilliger (Microsoft Corporation - Redmond, US)

License Creative Commons BY-NC-ND 3.0 Unported license
© James Terwilliger
Joint work of Terwilliger, James; Kindler, Ekkart

The benchmark and taxonomy working group gathered for the purpose of documenting commonality across disciplines. The group heard from representatives across all four disciplines present at the seminar. The need for such a benchmark became abundantly clear over the course of the discussion as the participants realized that, despite working towards similar and sometimes identical research goals, the words used to describe concepts in those disciplines were vastly different. For instance, the word "model" in one discipline corresponds to "meta-model" in another discipline, and to "instance" in yet another discipline. In the short time that the group was gathered, the group primarily focused on documenting these differences in language rather than resolving them. In addition, the group also documented the various settings in which bidirectional transformations occur, the problems that Bx solutions are intending to solve, and the set of solutions currently available.

Participants

- Anthony Anjorin
TU Darmstadt, DE
- Artur Boronat
University of Leicester, GB
- Christoph Brandt
University of Luxembourg, LU
- Anthony Cleve
University of Namur, BE
- Jácome Cunha
Univ. de Minho - Braga, PT
- Krzysztof Czarnecki
University of Waterloo, CA
- Zinovy Diskin
University of Waterloo, CA
- Hartmut Ehrig
TU Berlin, DE
- Nate Foster
Cornell University, US
- Jeremy Gibbons
University of Oxford, GB
- Robert Glück
University of Copenhagen, DK
- Martin Gogolla
Universität Bremen, DE
- Joel Greenyer
Universität Paderborn, DE
- Jean-Luc Hainaut
University of Namur, BE
- Frank Hermann
TU Berlin, DE
- Soichiro Hidaka
NII - Tokyo, JP
- Stephan Hildebrandt
Hasso Plattner Institut - Potsdam, DE
- Martin Hofmann
LMU München, DE
- Zhenjiang Hu
NII - Tokyo, JP
- Michael Johnson
Macquarie Univ. - Sydney, AU
- Ekkart Kindler
Technical Univ. of Denmark, DK
- Jochen M. Küster
IBM Research - Zürich, CH
- Ralf Lämmel
Universität Koblenz-Landau, DE
- Marius Lauder
TU Darmstadt, DE
- Kazutaka Matsuda
Tohoku University, JP
- Richard F. Paige
University of York, GB
- Alfonso Pierantonio
Univ. degli Studi di L'Aquila, IT
- Benjamin C. Pierce
University of Pennsylvania, US
- Jan Rieke
Universität Paderborn, DE
- Michael Schlereth
Wilhermsdorf, DE
- Alan Schmitt
INRIA Rhône-Alpes, FR
- Andy Schürr
TU Darmstadt, DE
- Perdita Stevens
University of Edinburgh, GB
- James Terwilliger
Microsoft Res. - Redmond, US
- Janis Voigtländer
Universität Bonn, DE
- Meng Wang
University of Oxford, GB
- Andrzej Wasowski
IT Univ. of Copenhagen, DK
- Jens-Holger Weber-Jahnke
University of Victoria, CA
- Yingfei Xiong
University of Waterloo, CA
- Tetsuo Yokoyama
Nanzan University, JP