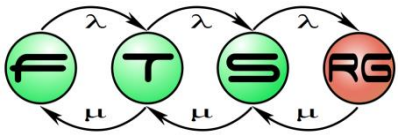
Bi- and Multidirectional Synchronization: Challenges and Approaches

a State of the Art Review by **Gábor Bergmann**

FTSRG Seminar, 2019-02-19



Hungarian Academy of Sciences

Supported by: the MTA-BME Lendület Cyber-Physical Systems Research Group, the János Bolyai Research Scholarship of the Hungarian Academy of Sciences and the ÚNKP-18-4 New National Excellence Program of the Ministry of Human Capacities.

Intro

- Bidirectional / multidirectional synchronization
  - Cloud drive on mobile ⇔ computer
  - Data binding, database editing based on view
  - Dentist appointments vs assistant
  - Healthcare, HIPAA
  - Complex MBE workflows
- Is all this really easy?
- How do I come into the picture?
  - MONDO, Dagstuhl 2018

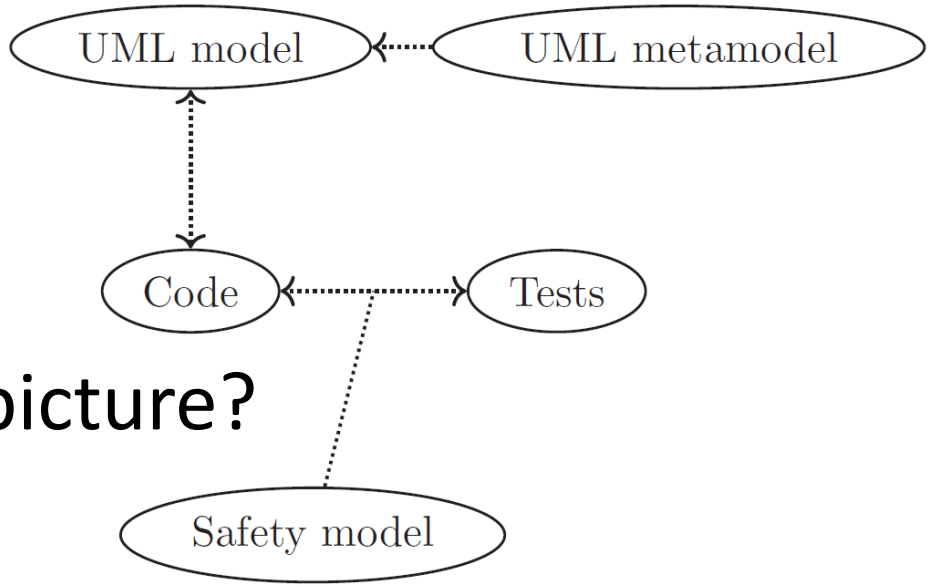
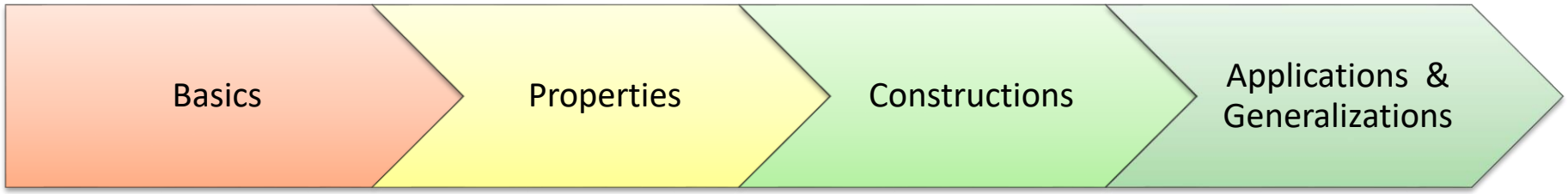
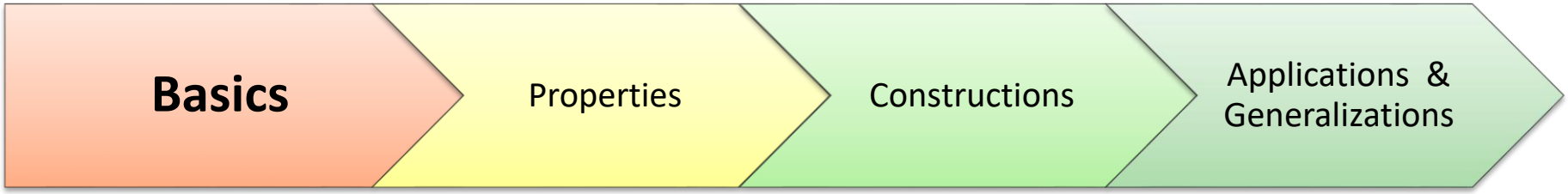


Image source: P. Stevens, "Bidirectional Transformations in the Large," *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Austin, TX, 2017, pp. 1-11. doi: 10.1109/MODELS.2017.8

Topics to cover





Basic Formal Notions

Symmetric / assymetric BX
Semantic variants

(Symmetric) BX

- Consistency between two models

$$R \subseteq M \times N \qquad R(m, n)$$

- + a pair of consistency restorers

$$\overrightarrow{R} : M \times N \longrightarrow N$$
$$\overleftarrow{R} : M \times N \longrightarrow M$$

- Bidirectional != bijective

„bijective transformations are the exception rather than the rule: the fact that one model contains information not represented in the other is part of the reason for having separate models"

Bidirectional model transformations in QVT: semantic issues and open questions

Perdita Stevens*

School of Informatics, University of Edinburgh

Abstract. We consider the OMG's Queries, Views and Transformations (QVT) standard as applied to the specification of bidirectional transformations between models. We discuss what is meant by bidirectional transformations, and the model-driven development scenarios in which they are needed. We analyse the fundamental requirements on tools which support such transformations, and discuss some semantic issues which arise. We argue that a considerable amount of basic research is needed before suitable tools will be fully realisable, and suggest directions for this future research.
Keywords: bidirectional model transformation, QVT

1 Introduction

The central idea of the OMG's Model Driven Architecture is that human intelligence should be used to develop models, not programs. Routine work should be, as far as possible, delegated to tools: the human developer's intelligence should be used to do what tools cannot. To this end, it is envisaged that a single platform independent model (PIM) might be created and transformed, automatically, into various platform specific models (PSMs) by the systematic application of understanding concerning how applications are best implemented on each specific platform. The OMG's Queries, Views and Transformations (QVT) standard [12] defines languages in which such transformations can be written.

In this paper we will discuss *bidirectional* transformations, focusing on basic requirements which such transformations should satisfy.

The structure of the paper is as follows. In the remainder of this section, we motivate bidirectional transformation, and especially, the need for non-bijective bidirectional transformations; we then discuss related work. Section 2 briefly summarises the most relevant aspects of the QVT standard. Section 3 discusses key semantic issues that arise. Section 4 proposes and motivates a framework and a definition of "coherent transformation". Finally Section 5 concludes.

In order to justify the considerable cost of developing a model transformation, it should ideally be reused; perhaps a vendor might sell the same transformation to many customers. However, in practice a transformation will usually have to be adapted to the needs of a particular application. Similarly, whilst we might

* Email: perdita@inf.ed.ac.uk. Fax: +44 131 667 7209

P. Stevens. Bidirectional Model Transformations in QVT: Semantic Issues and Open Questions. Journal of Software and Systems Modeling (SoSyM) 9(1):7–20, 2010.

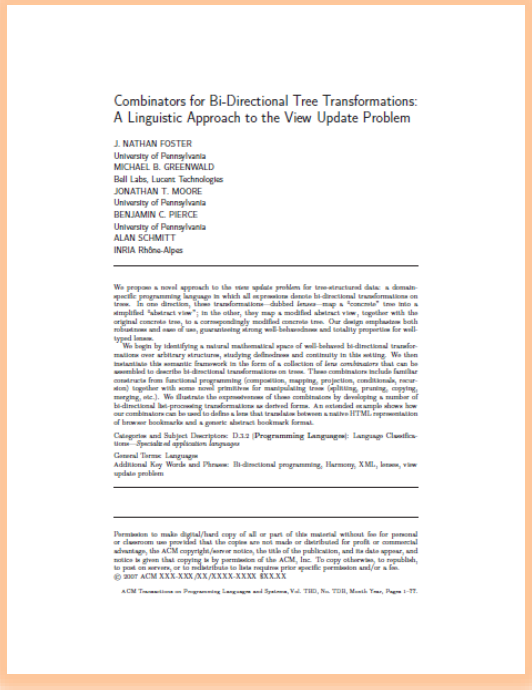
(Asymmetric) Lenses

- View-Update Problem @ „Harmony Group”
  - GET: view as function of source/master
  - PUT(BACK): update master from view

$$B = \boldsymbol{get}(A) \text{ and } A' = \boldsymbol{put}(B', A)$$

(asymmetric) lens, **a-lens**

- Trivial case: $A = \langle B, C \rangle$
  - GET/PUT ignores complement (C)
  - Again, this is not always applicable



Combinators for Bi-Directional Tree Transformations:
A Linguistic Approach to the View Update Problem

J. NATHAN FOSTER
University of Pennsylvania
MICHAEL B. GREENWALD
Bell Labs, Lucent Technologies
JONATHAN T. MOORE
University of Pennsylvania
BENJAMIN C. PIERCE
University of Pennsylvania
ALAN SCHMITT
INRIA Rhône-Alpes

Foster, J.N., Greenwald, M.B., Moore, J.T., Pierce, B.C.,Schmitt, A.: Combinators for bi-directional tree transformations: A linguistic approach to the view update problem. ACM Trans.Program. Lang. Syst.29(3), 17 (2007)

Orig. SIGPLAN paper: 2005 (coinage of „lens”)

Semantic Variants

- State-based vs. delta-based

  - Incrementality
  - Preservation of intent

Diskin, Z, Xiong, Y, Czarnecki, K. (2011). From State- to Delta-Based Bidirectional Model Transformations: the Asymmetric Case.. Journal of Object Technology. 10. 6: 1-25.10.5381/jot.2011.10.1.a6.

- Reflective updates

The best choice may include self-propagation from the updated view to itself

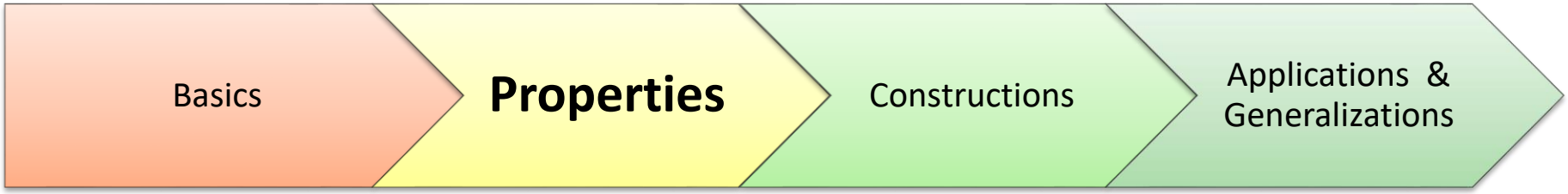
Diskin Z., König H., Lawford M. (2018) Multiple Model Synchronization with Multiary Delta Lenses. In: Russo A., Schürr A. (eds) Fundamental Approaches to Software Engineering. FASE 2018. Lecture Notes in Computer Science, vol 10802. Springer, Cham

- Partial consistency

  - Partially ordered degree of consistency
  - E.g. hyppocratic = monotonic

Stevens P. (2014) Bidirectionally Tolerating Inconsistency: Partial Transformations. In: Gnesi S., Rensink A. (eds) Fundamental Approaches to Software Engineering. FASE 2014. Lecture Notes in Computer Science, vol 8411. Springer, Berlin, Heidelberg

  - High-level consistency applicable only after low-level



Desirable Properties

(Very) well-behaved BX

Least surprise

Well-behaved

- Well-behaved BX: correct + hyppocratic

Correct

$$T(m, \overrightarrow{T}(m,n))$$

$$T(\overleftarrow{T}(m,n), n)$$

Hyppocratic

$$T(m,n) \implies \overrightarrow{T}(m,n) = n$$

$$T(m,n) \implies \overleftarrow{T}(m,n) = m$$

„First, do no harm.” Hippocrates, 450-355BC

- Special case for a-lens: PUTGET + GETPUT

$$l.get\ (l.put\ v\ s) = v \qquad \text{(PUTGET)}$$

$$l.put\ (l.get\ s)\ s = s \qquad \text{(GETPUT)}$$

Very well-behaved

- Very well-behaved (overwriteable):
  - Well-behaved + **history ignorant**

$$\overrightarrow{R}(m, \overrightarrow{R}(m', n)) = \overrightarrow{R}(m, n)$$

$$\overleftarrow{R}(\overleftarrow{R}(m, n'), n) = \overleftarrow{R}(m, n)$$

$$l.put\ v'\ (l.put\ v\ s) = l.put\ v'\ s \qquad \text{(PUTPUT)}$$

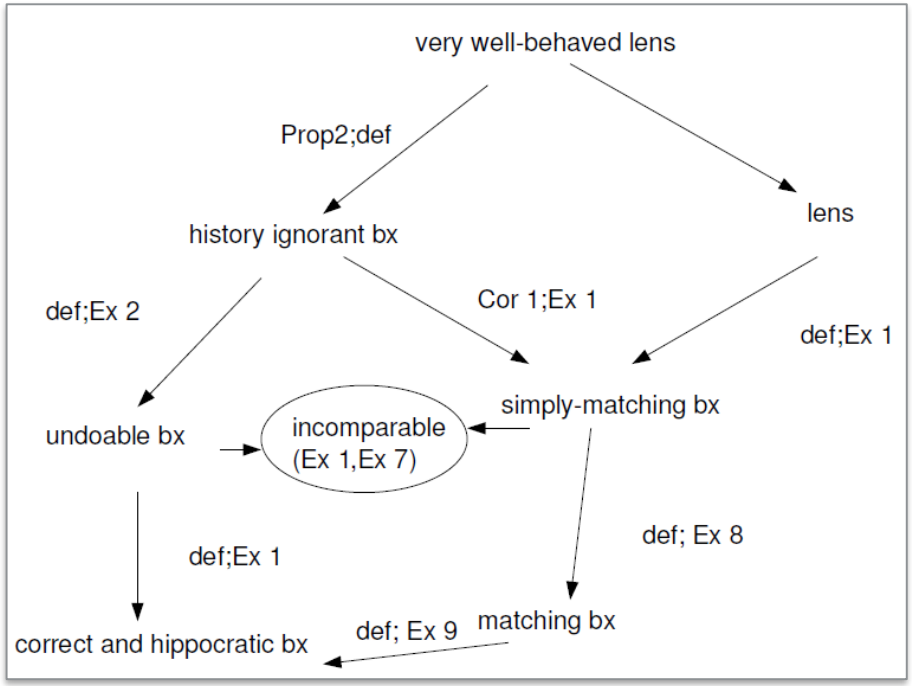
- This is an *optional* requirement

„fail to satisfy it for reasons that seem pragmatically unavoidable”

„a draconian restriction”

Very well-behaved cont'd.

- Very well-behaved:
  - Well-behaved + **history ignorant**
- Refined taxonomy



Stevens, Perdita. (2012). Observations relating to the equivalences induced on model sets by bidirectional transformations. EC-EASST 049. 10.14279/tuj.eceasst.49.714.

Least Change / Least Surprise

- Many correct choices for restorer / PUT (if non-bijective)
  - Which is the „best choice”?
  - c.f. „constant complement” in database view updates

> The action taken by the maintainer of a constraint after a violation should change no more than is needed to restore the constraint.

It turns out, however, that there are devils in the details.

JOURNAL OF OBJECT TECHNOLOGY
Published by AITO — Association Internationale pour les Technologies Objets
http://www.jot.fm/

On principles of Least Change and Least Surprise for bidirectional transformations

James Cheney[a] Jeremy Gibbons[b] James McKinna[a] Perdita Stevens[a]

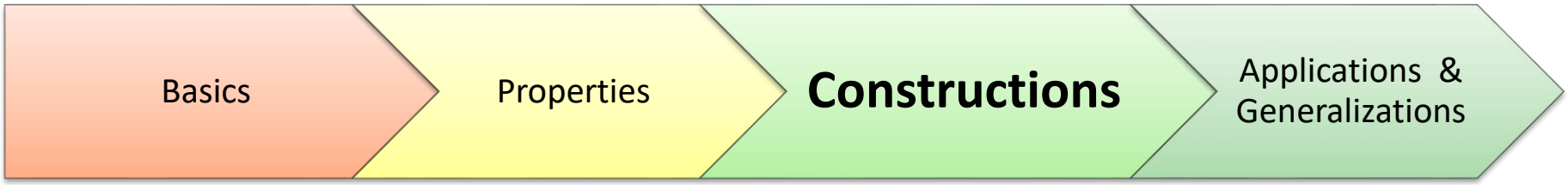
a. Laboratory for Foundations of Computer Science, School of Informatics, University of Edinburgh
b. Department of Computer Science, University of Oxford

Abstract In software engineering and elsewhere, different people may work intensively with different, but related, artefacts, e.g. models, documents, or code. They may use bidirectional transformations (bx) to maintain consistency between them. Naturally, they do not want their deliberate decisions disrupted, or their comprehension of their artefact interfered with, by a bx that makes changes to their artefact beyond the strictly necessary. This gives rise to a desire for a principle of Least Change, which has been often alluded to in the field, but seldom addressed head on. In this paper we present examples, briefly survey what has been said about least change in the context of bx, and identify relevant notions from elsewhere that may be applicable. We conclude that we cannot expect a Principle of Least Change to determine the optimal behaviour of a bx based on the consistency relation it embodies alone. Any such principle would bind the hands of the bx developer too tightly: the specification of *how* consistency is restored is as important a part of the development of a bx as the specification of *what* consistency means. Rather, what is required is a notion of *reasonable* behaviour of a bx that captures the idea that the bx's consistency restoration does not gratuitously surprise its user. We suggest considering continuity variants, particularly Hölder continuity. Such properties are too strong to expect them to hold universally, so we introduce the idea of a property holding piecewise on an *atlas* of subspace pairs.

Keywords bidirectional transformation; least change; least surprise; semantics; metrics; continuity; atlas

 In Journal of Object Technology, vol. 16, no. 1, 2011, pages 3:1–31. doi:10.5381/jot.2017.16.1.a3

James Cheney, Jeremy Gibbons, James McKinna, Perdita Stevens, “On principles of Least Change and Least Surprise for bidirectional transformations”, Journal of Object Technology, Volume 16, no. 1 (February 2017), pp. 3:1-31, doi:10.5381/jot.2017.16.1.a3.



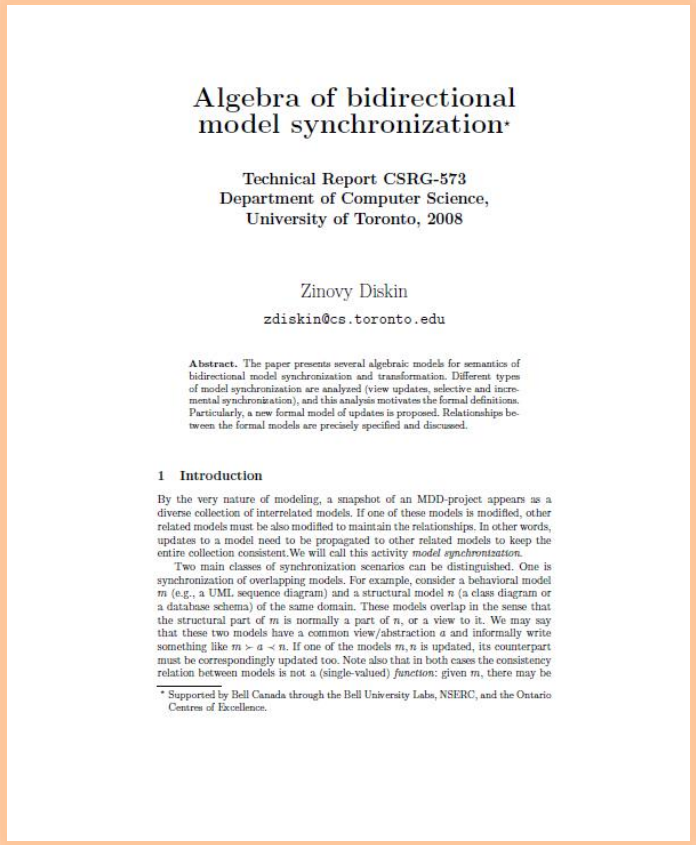
Constructions

Composition and decomposition
Bidirectional Programming
Grammar-based and federation-based synchronization

Composition of Lenses

- How to build large snyhcronization systems?
- Composition of lenses
  - Sequential composition
  - Span (shared source/master)
  - Co-span (shared target/view)
- Composition laws, e.g.

Proposition 10. The co-targetial composition $(\mu \bowtie \nu)$ of lenses is a (very) well-behaved/undoable diagonal system as soon as both lenses are such. In addition, the system is Hippocratic.



Algebra of bidirectional model synchronization*

Technical Report CSRG-573
Department of Computer Science,
University of Toronto, 2008

Zinovy Diskin
zdiskin@cs.toronto.edu

Abstract. The paper presents several algebraic models for semantics of bidirectional model synchronization and transformation. Different types of model synchronization are analyzed (view updates, selective and incremental synchronization), and this analysis motivates the formal definitions. Particularly, a new formal model of updates is proposed. Relationships between the formal models are precisely specified and discussed.

1 Introduction

By the very nature of modeling, a snapshot of an MDD-project appears as a diverse collection of interrelated models. If one of these models is modified, other related models must be also modified to maintain the relationships. In other words, updates to a model need to be propagated to other related models to keep the entire collection consistent.We will call this activity *model synchronization*.

Two main classes of synchronization scenarios can be distinguished. One is synchronization of overlapping models. For example, consider a behavioral model m (e.g., a UML sequence diagram) and a structural model n (a class diagram or a database schema) of the same domain. These models overlap in the sense that the structural part of m is normally a part of n, or a view to it. We may say that these two models have a common view/abstraction a and informally write something like $m \succ a \prec n$. If one of the models m, n is updated, its counterpart must be correspondingly updated too. Note also that in both cases the consistency relation between models is not a (single-valued) *function*: given m, there may be

* Supported by Bell Canada through the Bell University Labs, NSERC, and the Ontario Centres of Excellence.

Diskin Z. (2008) Algebraic Models for Bidirectional Model Synchronization. In: Czarnecki K., Ober I., Bruel JM., Uhl A., Völter M. (eds) Model Driven Engineering Languages and Systems. MODELS 2008. Lecture Notes in Computer Science, vol 5301. Springer, Berlin, Heidelberg

Decomposition of Lenses

- Decomposition: BX → a-lenses
  - Programming lenses may be easier!
  - Span of a-lenses: trivial
    - Any WB BX is ~ a span of lenses from a „union” master model
  - Co-span of a-lenses:

Definition 8 Let $R : M \leftrightarrow N$ be a bidirectional transformation inducing equivalences and a coordinate system as usual. We say that R is *matching* if there is a bijection $f : M_F \to N_B$ such that $R(m_F, f(m_F))$ for all $m_F \in M_F$. We say that R is *simply matching* if, in addition, $R(m_F, n_B)$ holds *only* when $n_B = f(m_F)$.

Corollary 1 *If R is history ignorant, then R is simply matching.*

Theorem 1 *A bidirectional transformation $R : M \leftrightarrow N$ is simply matching if and only if it can be decomposed into a pair of lenses working "tail to tail". Any such decomposition gives rise to a choice of matching transversal for R, and vice versa.*

Stevens, Perdita. (2012). Observations relating to the equivalences induced on model sets by bidirectional transformations. EC-EASST 049. 10.14279/tuj.eceasst.49.714.

Bidirectional Programming @ Harmony Group

- „Write one program, execute in 2 directions”
  - based on lens compositions (sequential, conditional…)
- Harmony: updateable XML views (fork, hoist, etc.)

Foster, J.N., Greenwald, M.B., Moore, J.T., Pierce, B.C.,Schmitt, A.: Combinators for bi-directional tree transformations: A linguistic approach to the view update problem. ACM Trans. Program. Lang. Syst. 29(3), 17 (2007)

- Relational Lens: updateable RelDB query (with FD)

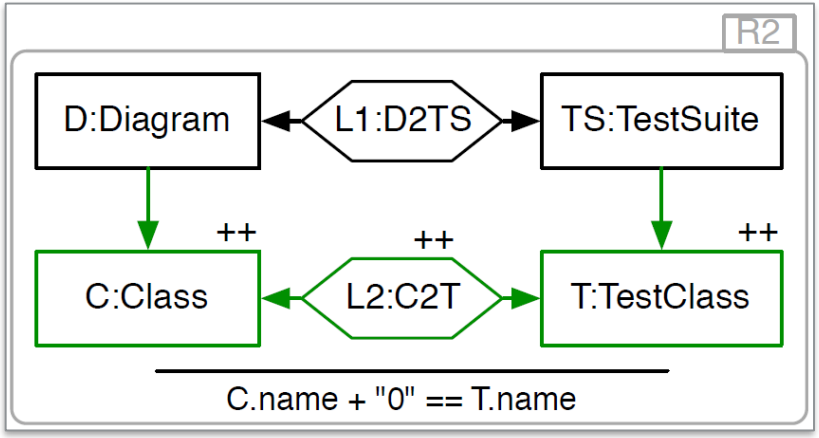
Bohannon, A., Vaughan, J.A., Pierce, B.C.: Relational lenses: a language for updateable views. In: Principles of Database Systems (PODS), Extended Version Available as University of Pennsylvania technical report MS-CIS-05-27 (2006)

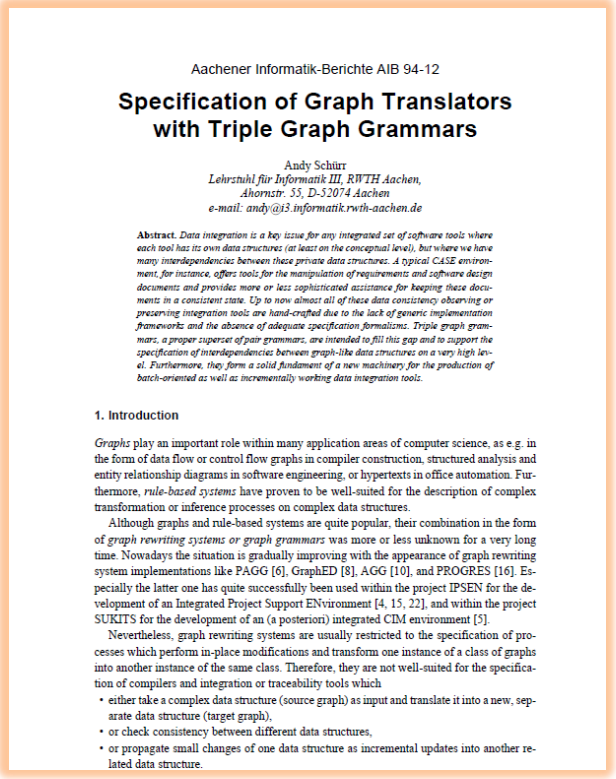
- Boomerang: bidirectional typed ~regexp (+dict)

Bohannon, A., Foster, J.N., Pierce, B.C., Pilkiewicz, A. and Schmitt, A.: Boomerang: Resourceful Lenses for String Data. In ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL), San Francisco, California, January 2008.

Triple Graph Grammars

- Rule-based specification for consistency
  - Automatically derived BX restorers
  - Correctness✓, other properties?
- ~ translation in string grammars





Aachener Informatik-Berichte AIB 94-12

Specification of Graph Translators with Triple Graph Grammars

Andy Schürr
Lehrstuhl für Informatik III, RWTH Aachen,
Ahornstr. 55, D-52074 Aachen
e-mail: andy@i3.informatik.rwth-aachen.de

Abstract. *Data integration is a key issue for any integrated set of software tools where each tool has its own data structures (at least on the conceptual level), but where we have many interdependencies between these private data structures. A typical CASE environment, for instance, offers tools for the manipulation of requirements and software design documents and provides more or less sophisticated assistance for keeping these documents in a consistent state. Up to now almost all of these data consistency observing or preserving integration tools are hand-crafted due to the lack of generic implementation frameworks and the absence of adequate specification formalisms. Triple graph grammars, a proper superset of pair grammars, are intended to fill this gap and to support the specification of interdependencies between graph-like data structures on a very high level. Furthermore, they form a solid fundament of a new machinery for the production of batch-oriented as well as incrementally working data integration tools.*

1. Introduction

Graphs play an important role within many application areas of computer science, as e.g. in the form of data flow or control flow graphs in compiler construction, structured analysis and entity relationship diagrams in software engineering, or hypertexts in office automation. Furthermore, *rule-based systems* have proven to be well-suited for the description of complex transformation or inference processes on complex data structures.

Although graphs and rule-based systems are quite popular, their combination in the form of *graph rewriting systems or graph grammars* was more or less unknown for a very long time. Nowadays the situation is gradually improving with the appearance of graph rewriting system implementations like PAGG [6], GraphED [8], AGG [10], and PROGRES [16]. Especially the latter one has quite successfully been used within the project IPSEN for the development of an Integrated Project Support ENvironment [4, 15, 22], and within the project SUKITS for the development of an (a posteriori) integrated CIM environment [5].

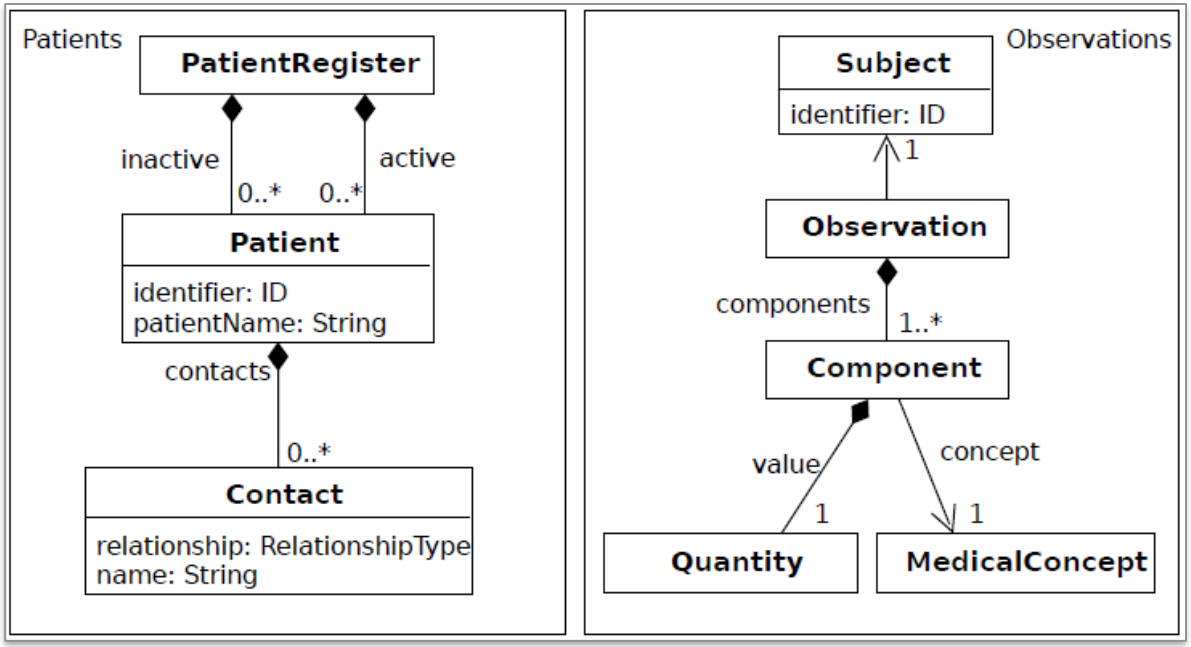
Nevertheless, graph rewriting systems are usually restricted to the specification of processes which perform in-place modifications and transform one instance of a class of graphs into another instance of the same class. Therefore, they are not well-suited for the specification of compilers and integration or traceability tools which

- either take a complex data structure (source graph) as input and translate it into a new, separate data structure (target graph),
- or check consistency between different data structures,
- or propagate small changes of one data structure as incremental updates into another related data structure.

Schürr A. (1995) Specification of graph translators with triple graph grammars. In: Mayr E.W., Schmidt G., Tinhofer G. (eds) Graph-Theoretic Concepts in Computer Science. WG 1994. Lecture Notes in Computer Science, vol 903. Springer, Berlin, Heidelberg

Metamodel composition for federation

- If models are split/merged based on metamodel…
  - Composition as „colimit”



Multimodel Correspondence through Inter-model Constraints

Patrick Stünkel
Western Norway University of Applied Sciences
Bergen, Norway
past@hvl.no

Harald König
FHDW Hannover
Hanover, Germany
Harald.Koenig@fhdw.de

Yngve Lamo
Western Norway University of Applied Sciences
Bergen, Norway
yla@hvl.no

Adrian Rutle
Western Norway University of Applied Sciences
Bergen, Norway
aru@hvl.no

ABSTRACT

1 INTRODUCTION

CCS CONCEPTS

KEYWORDS

ACM Reference Format:

Stünkel, P., König, H., Lamo, Y., Rutle, A.. 2018. Multimodel correspondence through inter-model constraints. In *Conference Companion of the 2nd International Conference on Art, Science, and Engineering of Programming* (Programming'18 Companion). ACM, New York, NY, USA, 9-17. DOI: https://doi.org/10.1145/3191697.3191715

Applications and Generalizations

Functional Programming
Access Control
MX

Monadic BX in Functional Programming

- Provide result under some monad, e.g.
  - Future<T>
  - User choice, e.g. Input<T>
  - State (transformation remembers history)
- Does it compose / commute with other monads?
- Many approaches

Abou-Saleh F., Cheney J., Gibbons J., McKinna J., Stevens P. (2015) Notions of Bidirectional Computation and Entangled State Monads. In: Hinze R., Voigtländer J. (eds) Mathematics of Program Construction. MPC 2015. Lecture Notes in Computer Science, vol 9129. Springer, Cham

Abou-Saleh F., Cheney J., Gibbons J., McKinna J., Stevens P. (2016) Reflections on Monadic Lenses. In: Lindley S., McBride C., Trinder P., Sannella D. (eds) A List of Successes That Can Change the World. Lecture Notes in Computer Science, vol 9600. Springer, Cham

Updateable Security Views

- RW access control as lens
  - Read access control = GET
  - Write access control = PUT…
    - …that can reject
- Contributions
  - Basic formalism
  - Desirable properties
  - Composition
  - Boomerang-like applications

Updatable Security Views

J. Nathan Foster Benjamin C. Pierce Steve Zdancewic
University of Pennsylvania

Abstract

Security views are a flexible and effective mechanism for controlling access to confidential information. Rather than allowing untrusted users to access source data directly, they are instead provided with a restricted view, from which all confidential information has been removed. The program that generates the view effectively embodies a confidentiality policy for the underlying source data. However, this approach has a significant drawback: it prevents users from updating the data in the view.

To address the "view update problem" in general, a number of bidirectional languages *have been proposed. Programs in these languages—often called* lenses*—can be run in two directions: read from left to right, they map sources to views; from right to left, they map updated views back to updated sources. However, existing bidirectional languages do not deal adequately with security. In particular, they do not provide a way to ensure the integrity of source data as it is manipulated by untrusted users of the view.*

We propose a novel framework of secure lenses *that addresses these shortcomings. We enrich the types of basic lenses with equivalence relations capturing notions of confidentiality and integrity, and formulate the essential security conditions as non-interference properties. We then instantiate this framework in the domain of string transformations, developing syntax for* bidirectional string combinators *with* security-annotated regular expressions *as their types.*

1. Introduction

Security views are a widely used mechanism for controlling access to confidential information in databases and other systems that manage structured information. By forcing users to access data via views that only expose public information, data administrators ensure that secrets will not be leaked, even if the users mishandle the data or are malicious. Security views are robust, making it impossible for users to leak the source data hidden by the view, and they are flexible: since they are implemented as arbitrary programs, they can be used to enforce extremely fine-grained access control policies. However, they are not usually updatable—and for good reason! Propagating updates to views made by untrusted users can, in general, alter the source data, including the parts that are hidden by the view.

Still, there are many applications in which having a mechanism for reliably updating security views would be extremely useful. For example, consider Intellipedia, a collaborative data sharing system based on Wikipedia that is used by members of the intelligence community. The data stored in Intellipedia is classified at the granularity of whole documents, but many documents actually contain a mixture of highly classified and less-classified data. In order to give users with low clearances access to the portions of documents they have sufficient clearance to see, documents often have to be regraded: i.e., the highly classified parts need to be erased or redacted, leaving behind a residual document—a security view!—that can be reclassified at a lower level of clearance. Of course (since we are talking about a wiki), we would like the users of these views to be able to make updates—e.g., to correct errors or add new information—and have their changes be propagated back to the original document.

In general, for a view to be updatable, the program that generates it needs to be *bidirectional*. That is, it must not only be able to transform sources to views but also to map updated views back to updated sources. In previous work, we and many others have proposed a family of languages for describing bidirectional transformations, often called *lenses* [19], [8], [7], [21], [37], [41], [26], [9], [24], [35], [17], [23], [30], [28]. Formally, a lens l mapping between a set S of "source" structures and a set V of "views" comprises three functions

$$l.get \in S \longrightarrow V$$
$$l.put \in V \longrightarrow S \longrightarrow V$$
$$l.create \in V \longrightarrow S$$

that obey "round-tripping" laws for every $s \in S$ and $v \in V$.

$$l.get\ (l.put\ v\ s) = v \qquad \text{(PUTGET)}$$
$$l.get\ (l.create\ v) = v \qquad \text{(CREATEGET)}$$
$$l.put\ (l.get\ s)\ s = s \qquad \text{(GETPUT)}$$

The *get* function defines the view and is a total function from S to V. There are two functions that handle updates: the *put* function takes an updated V and the original S and weaves them together to yield a correspondingly modified S, while the *create* function handles the special case where we need to compute an S from a V but have no S to use as the original (it fills in any source data that is not reflected in the view with default values).

J. N. Foster, B. C. Pierce and S. Zdancewic, "Updatable Security Views," *2009 22nd IEEE Computer Security Foundations Symposium*, Port Jefferson, NY, 2009, pp. 60-74. doi: 10.1109/CSF.2009.25

Unique MX Challenges

- Two seminal papers

P. Stevens, "Bidirectional Transformations in the Large," *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Austin, TX, 2017, pp. 1-11. doi: 10.1109/MODELS.2017.8

Diskin Z., König H., Lawford M. (2018) Multiple Model Synchronization with Multiary Delta Lenses. In: Russo A., Schürr A. (eds) Fundamental Approaches to Software Engineering. FASE 2018. Lecture Notes in Computer Science, vol 10802. Springer, Cham

- Dagstuhl Seminar 2018 Dec: many open questions
  - Non-local consistency?
  - Concurrent updates?
  - Whack-a-mole property?
  - Composition of BX into MX?
    - Cyclic networks?
  - MX type theory etc.