ISSN 2186-7437

NII Shonan Meeting Report

No. 231

Bidirectional Transformations: Foundations and Applications

Kazutaka Matsuda
Romina Eramo
Michael Johnson
Vadim Zaytsev

May 26–29, 2025

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo, Japan

Bidirectional Transformations: Foundations and Applications

Organisers:
Kazutaka Matsuda (Tohoku University, Japan)
Romina Eramo (University of Teramo, Italy)
Michael Johnson (Macquarie University, Australia)
Vadim Zaytsev (University of Twente, The Netherlands)

May 26–29, 2025

Background and introduction

Bidirectional transformation [2, 8, 9, 13, 22], a mechanism to synchronise multiple data sources in different formats, is a recurring concept in software development and has various applications, including the classic view-update problem in database systems, along with synchronisation and communication in database systems, GUI/web application development, implementation of serialisers, compiler frontend, and model-driven software development. Accordingly, bidirectional transformations have been studied in multiple research disciplines, including databases, programming languages, software engineering, graph transformations, and applied mathematics, from various perspectives, such as:

- **Construction of bidirectional transformations:** including programming languages and systems (such as lenses and triple graph grammars), type systems, program synthesis (programming by example), program transformations to construct bidirectional programs from unidirectional ones, programming frameworks, and programming methodologies.
- **Applications of bidirectional transformations:** including synchronisation of software artefacts (such as object-oriented models) created in the software development process, tooling to support such applications, the classic view-update problem on relational databases or more modern architectures, incremental view maintenance, exploration of design space and data exchanges between databases.
- **Foundations of bidirectional transformations:** including mathematical models of bidirectional transformations which arose in various fields including those above, abstractions, properties and theorems, and mechanised formalisation of such approaches in theorem provers such as Coq/Lean/Agda.

These topics have their own right and have mainly been studied intensively in each community, but these perspectives share interests, goals, challenges,

and techniques and have strong synergy; well-designed construction methods encourage applications and provide hints for abstractions, applications suggest directions to which construction methods and foundations to evolve and serve as benchmarks, and mathematical foundations suggest new designs of construction methods and application design which are effective for various forms of consistency of systems.

Overview of the meeting

The focus of this Shonan meeting was to provide researchers from different disciplines an opportunity to meet up, exchange ideas, connect groups, renewing and boosting the results from the previous meetings held with the same spirit (2011 Dagstuhl [22], 2013 Banff [13], 2016 Shonan [2], 2018 Dagstuhl [8]), and from the series of the BX workshops (2012 Estonia, 2013 Italy, 2014 Greece, 2015 Italy, 2016 Netherlands, 2017 Sweden, 2018 France, 2019 USA, 2021 online, 2022 France). It has been 6 years since the last BX meeting at Dagstuhl and 3 years since the last BX workshop, and there has been a lot of research results in each field, justifying the importance of this Shonan meeting.

The organisers are part of the steering committee of the BX workshop series (started in 2012), which represents the area of bidirectional transformation research. This seminar aims to be broad and inclusive; the guest list contains new faces from the bidirectional transformation and neighbouring research communities, as well as long-term contributors in this area. Further information is available at: http://bx-community.wikidot.com.

In this four-day meeting, we wanted to include:

- summaries of recent research in each field given by selected invitees/organisers;
- talks from invitees that suggest new ideas and challenges and present ongoing research work;
- intra-community discussion to identify potential collaborations and select topics among the given talks;
- inter-community group discussions — participants divided into sub-groups to discuss specific challenges or new topics;
- wrap-up discussions.

Participants were experts from different fields who brought new ideas and perspectives, leading to further collaboration and new research results.

List of Participants

- **Romina Eramo**, University of Teramo, Italy (Organiser)
- **Kazutaka Matsuda**, Tohoku University, Japan (Organiser)
- **Michael Johnson**, Macquarie University, Sydney, Australia (Organiser)
- **Vadim Zaytsev**, University of Twente, The Netherlands (Organiser)
- **Meng Wang**, University of Bristol, UK
- **Soichiro Hidaka**, Hosei University, Japan
- **Alessio Bucaioni**, Mälardalen University, Sweden
- **Antonio Cicchetti**, Mälardalen University, Sweden
- **Alessandro Celi**, University of L'Aquila, Italy
- **Jules Hedges**, CyberCat Institute, University of Strathclyde, UK
- **Makoto Onizuka**, Osaka University, Japan
- **Bryce Clarke**, Tallinn University of Technology, Estonia
- **Jens Kosiol**, Philipps-Universität Marburg, Germany
- **Xing Zhang**, Peking University, China
- **David Jaz Myers**, Topos Institute London, UK
- **Tao Zan**, Longyan University, China
- **Samantha Frohlich**, University of Bristol, UK
- **Li-Yao Xia**, INRIA, France
- **Lars König**, Karlsruhe Institute of Technology, Germany
- **Keisuke Nakano**, Tohoku University, Japan
- **Anders Ågren Thuné**, Uppsala University, Sweden
- **Matthew Di Meglio**, University of Edinburgh, UK
- **Hiroyuki Kato**, National Institute of Informatics, Japan
- **Mark Barbone**, Cornell University, USA
- **Anders Miltner**, Simon Fraser University, Canada
- **Matthias Barkowsky**, Hasso-Plattner-Institut, Germany

Overview of Talks

NoSQL Schema Optimisation for Time-Dependent Workloads [34]

Makoto Onizuka, Osaka Univeristy, Japan

Predefined or predictable access patterns in database workloads are common in automated IoT applications or scientific analysis. These patterns are used to enhance database system performance through adaptive database migrations, optimising the database for the expected workload at any given time. To this end, we consider the problem of schema optimisation for time-dependent workloads in NoSQL databases. Our contributions are twofold. First, we formulate the optimisation problem of time series schema design as a single integer linear program (ILP) to minimise the total cost of time-dependent workload execution and database migration. Second, we propose a novel technique for pruning candidate schemas by decomposing the ILP designed for the original time-dependent workload via approximation into a hierarchy of local ILPs for smaller sub-workloads. The experiments confirm that our proposal is superior to static schema optimisation methods for typical time-dependent workloads, reducing cumulative latency by up to 40% across all time steps. Furthermore, our pruning technique demonstrates remarkable scalability with the number of time steps, diverging from the NP-complete nature of a naive approach that scales almost quadratically with the number of time steps.

Transaction Management in Collaborative Data Management [37]

Makoto Onizuka, Osaka Univeristy, Japan

We consider a transaction management problem in collaborative data management where servers are connected using views defined using bidirectional transformations. We propose a new approach, which corresponds to a distributed version of conservative two-phase lock (C2PL). We first introduce two notions to express update propagation scope, family record set and update participant (server) scope. The family record set expresses an atomic unit for distributed locking before making update propagation. The update participant scope effectively reduces the search space of participants for acquiring locks. Second, we design put rules for views defined with monotonic SPJU queries in order to ensure that BX propagates updates only to the records in the same family record set. Finally, we propose an adaptive mechanism that appropriately chooses either the simple approach (2PL mode) or the family record set based approach (C2PL mode) based on the contention rate of transactions. Intensive experiments verify the effectiveness of our proposals.

Characterisation of Computable Lenses

Keisuke Nakano, Tohoku University, Japan

Two topics related to the characterisation of computable asymmetric lenses, which are pairs of *get* and *put* functions, are presented in this talk.

Firstly, a characterisation of (possibly partial) well-behaved lenses is presented. Fischer et al. [11] have shown that well-behaved lenses can be characterised by a set of lens laws on the *put* function, which are necessary and sufficient for the existence and uniqueness of a *get* function that is well-behaved, assuming both *get* and *put* are total. In this talk, a similar characterisation was presented for the case where the *get* and *put* functions may be partial. The required conditions have been identified as (partial) injectivity and idempotency of the *put* function. This result has been published in PEPM 2025 [16].

Secondly, an idea for a computational model for computable well-behaved lenses is presented as ongoing work. Since well-behaved lenses can be characterised by the injectivity and idempotency of the *put* function, the computational model should be designed to capture these properties. In this talk, the speaker introduced his own work on a computational model for idempotent functions, which exactly covers all computable idempotent functions, with the aim of contributing to the design of a computational model for well-behaved lenses. This result has been published in MFCS 2021 [27].

Bidirectional Transformations for Coupling, Studying, and Certifying Dynamical Systems

David Jaz Myers, Topos Institute London, UK

Open dynamical systems whose dynamics depend on free parameters and which expose some variables of their state may be coupled by setting their parameters as functions of the exposed variables of other systems. Together with their parallel (Cartesian) product, these systems constitute a lax symmetric monoidal functor from a category of interfaces (consisting of parameter and exposed variable sets) and coupling laws (often expressed as wiring diagrams) to the category of sets — that is, we have a symmetric monoidal right module of systems over the symmetric monoidal category of interfaces and coupling laws. Schultz, Spivak and Vasilakopoulou [31] show that the behaviour of these systems may be expressed as a morphism of lax symmetric monoidal functors from this module of systems to a module of time-varying sets — sheaves on the interval domain of the real line.

In this talk, we'll see that the SSV behaviour functors — and many others similar behaviour functors — are in fact representable when seen not as concerning right modules of categories, but as concerning right modules over double categories. We will develop the theory of (loose) modules between double categories using an approach inspired by Joyal's "barrels" (joint work with Sophie Libkind [25]), and describe the cartesian pseudo-functoriality of restriction of loose right modules which allows for the pseudo-functorial construction of symmetric monoidal loose right modules of open dynamical systems from an abstract notion of "tangent bundle category". By expanding the definition of "tangent bundle" in this way, we include all sorts of generalised Moore machines (including not only systems of ordinary differential equations, but also partially observable Markov processes and various sorts of non-deterministic automata).

We'll then see a general result (joint work with Matteo Capucci [5]) giving conditions under which discrete opfibration classifiers in a 2-category K can be lifted to the 2-category of algebras and lax morphisms for a 2-monad T on K. We will use this result to show that a certain symmetric monoidal loose right module

of spans is a discrete opfibration classifier among symmetric monoidal loose right modules, and conclude by showing that a variety of behaviour functors for open dynamical systems are covariantly representable. Time permitting, we will also see that system safety and stability properties are often themselves contravariantly representable via the representability of Lyapunov and control barrier functions by functions into simple systems.

Using Lenses for Enabling Differentially Private Algorithms

Anders Miltner, Simon Fraser University, Canada

Differential privacy involves ensuring that aggregate responses are not expected to change substantially with the inclusion or exclusion of a single piece of data. A differentially private algorithm essentially ensures that malicious actors cannot identify the individuals that comprise the dataset. However, ensuring that an algorithm is differentially private is quite tricky, as it requires stability with respect to every individual data point, and so many of the algorithms make heavy requirements on the input data. In this talk, we investigate the use of lenses for enabling differentially private algorithms. In essence, while the data itself may not satisfy the rigid input requirements, one may be able to put the data in a view that does, necessitating a lens-like structure. We explore two examples, one in differential dataset release, and one in differential aggregation.

Partial-State Lenses

Kazutaka Matsuda, Tohoku University, Japan

An open issue in BX is a compositional and lawful treatment of multiple views that share the same source, where an update to one view may trigger corresponding changes in others, making it hard to maintain consistency while preserving the user's update. In this talk, we introduce our work-in-progress framework called partial-state lenses, which allow source and view states to be partially-specified. This concept refines the notion of update preservation to the preservation of specified information, allowing unspecified information to be modified by an update from other views. The strength of the framework lies in its support for compositional reasoning of partial-specifiedness-aware well-behavedness, which ensures update preservation.

Dependent Optics

Jules Hedges, CyberCat Institute, University of Strathclyde, UK

Two generalisations of (not-necessarily-well-behaved) simple lenses that arise in many applications of applied category theory are monoidal optics and dependent lenses (also known as morphisms of containers, and equivalently natural transformations of polynomial endofunctors). Roughly speaking, optics require almost nothing of their base category (only a symmetric monoidal product) and so can almost always be defined, whereas dependent lenses require a lot of the base category (finite limits and local Cartesian closure) but in return are incredibly well behaved. Different groups in applied category theory use both

structures for essentially the same reasons, but unifying them mathematically is formidably difficult. I discussed our progress on this problem, which we call the problem of dependent optics — joint work with Matteo Capucci, Bruno Gavranović, Eigil Rischel, Dylan Braithwaite and André Videla.

First we construct the category of monoidal optics by starting from the category of adaptors (i.e. pairs of morphisms in opposite directions) by freely extending it with a morphism $(X, Y) \to (1, 1)$ for each morphism $X \to Y$ in the base category. For this we use a theorem of Hermida and Tennent giving a concrete construction of this kind of free extension which we reformulate in the language of "weighted coparametrisation".

This reduces the problem to finding a category of "dependent adaptors", adaptors whose backwards types are indexed by forwards values. This is not possible in a standard formulation of dependent type theory, but it is possible using a recent extension known as quantitative type theory (QTT) that is implemented in Idris2 [4]. However the existing semantics of QTT was not sufficient to give a good semantic definition of dependent optics. For this we developed a novel categorical semantics of a fragment of QTT that can be defined in any regular category, but in the talk I skipped this for time.

I concluded by showing as an example that we can use this to calculate an intuitively plausible definition of dependent prisms.

Relational Hoare Lenses: Reasoning about Lenses for Configurable Systems

Mark Barbone, Cornell University, USA

Many programs are not complete as-is: they depend on some configuration. One instance of this is for network switches, which must be configured by the control plane. In this case, bidirectional transformations such as lenses can be used to translate between different versions or formats of the configuration.

To formally verify these configurable programs, we need to formally verify the lenses used for translating the configuration. This talk presents ongoing work on tools for formally verifying lenses between program configurations, in terms of relational properties of the programs.

Enabling Direct Manipulation of Plain-Text Output for Template Programs

Tao Zan, Longyan University, China

Templates play a crucial role in modern software development by automating the generation of artefacts across diverse domains, improving both productivity and consistency. However, most existing template languages are unidirectional: they generate output but offer limited support for modifying the template in response to changes in that output. Our previous system, BIT [17], supports reflecting code changes back into the model but does not accommodate template adaptation. To bridge this gap, we introduce bit2, a bidirectional template framework that enables direct manipulation of output while maintaining the ability to update the underlying template. Unlike traditional approaches that generate plain text, bit2 produces computation — structured out-

put—preserving the logic of the original template — which is then transformed into lambda-structured representations in a functional programming style. This enables reliable reverse mapping from output edits back to the template. We have implemented bit2 in TypeScript (18,000 lines of code), released as an NPM module, along with LiveT, a VSCode plugin that supports forward and backward evaluation. LiveT allows users to insert, delete, or replace elements directly in the output, making template modification intuitive and robust.

Bidirectional Transformations for Digital Twins

Alessio Bucaioni, Mälardalen University, Sweden
Alessandro Celi, University of L'Aquila, Italy
Antonio Cicchetti, Mälardalen University, Sweden
Romina Eramo, University of Teramo, Italy

Digital Twins (DT) are virtual counterparts of physical systems that enable continuous monitoring, simulation, and optimisation across the lifecycle of real-world entities. Digital Twin Engineering (DTE) encompasses the methodologies and technologies used to design, build, and evolve these twins in dynamic and complex environments.

This talk introduces the foundational concepts of DT and DTE, and highlights the critical role of Bidirectional Transformations (BX) in ensuring consistency and synchronisation between physical systems and their digital counterparts. BX techniques facilitate seamless updates across heterogeneous models, enabling integration, traceability, and automation in DT workflows. We survey recent research that leverages BX in DT contexts. A special focus is placed on emerging challenges, particularly the integration of Generative AI and Large Language Models (LLMs) to automate or assist in BX specifications. We explore both the promises and limitations of using LLMs in this domain, pointing to current gaps and future research directions.

Evolutionary Framework and Conflict Resolution for Multidirectional Transformations [19]

Soichiro Hidaka, Hosei University, Japan

Multidirectional transformations by bipartite network of well-behaved asymmetric bidirectional transformations are discussed. The bipartite network consists of a set of base tables and a set of views, respectively, while each edge corresponds to a bidirectional transformation from one of the base tables to one of the views. Although the architecture based on the bipartite network has been studied as Dejima network in existing work for collaborative data sharing, this work introduces a reconfiguration mechanism of the network while maintaining alignment of ranges of incident bidirectional transformations. We also talk about conflict resolution of updates reaching to identical nodes using operational transformations.

The Monadic Profunctor Paradigm of Bidirectional Programming (Monadic BX)

Samantha Frohlich, University of Bristol, UK
Li-yao Xia, INRIA, France

Converting data from one representation to another and vice versa is a common task in software. However, naïvely specifying both conversion directions separately is error prone and introduces conceptual duplication. Bidirectional programming advocates for the writing of one piece of code that can be interpreted in both directions (e.g., for a conversion and its inverse). Unfortunately, current bidirectional programming techniques generally require unfamiliar programming idioms such as restricted and specialised combinator libraries. Instead, we introduce a framework for composing bidirectional programs monadically, enabling bidirectional programming with a familiar functional programming abstraction. This abstraction leverages compositionality and equational reasoning for the verification of the required round tripping properties for such monadic bidirectional programs. We introduce partial monadic profunctors as a new, general paradigm of bidirectional programming, and demonstrate their use in a number of domains: parsers/printers, pickling, lenses, generators/predicates, and logic programming.

Monadic BX @ Generators

Samantha Frohlich, University of Bristol, UK

Expert users of property-based testing often labor to craft random generators that encode detailed knowledge about what it means for a test input to be valid and interesting. Fortunately, the fruits of this labor can also be put to other good uses. In the bidirectional programming literature, for example, generators have been repurposed as validity checkers, while Python's Hypothesis library uses them to shrink and mutate test inputs. To unify and generalise these uses (and more), we propose reflective generators, a new foundation for random data generators that can "reflect" on an input value to calculate the random choices that could have been made to produce it. Reflective generators combine ideas from two existing abstractions: free generators and partial monadic profunctors. They can be used to implement and enhance the aforementioned shrinking and mutation algorithms, generalising them to work for any values that could have been produced by the generator, not just ones for which a trace of the generator's execution is available. Beyond shrinking and mutation, reflective generators simplify and generalise a published algorithm for example-based generation; they can also be used as checkers, partial value completers, and test data producers like enumerators.

Monadic BX @ Logic Programming

Li-yao Xia, INRIA, France

Correctness in this bidirectional programming framework is characterised by round-tripping properties between "forward" and "backward" executions of the program. Unfortunately, the monadic-profunctor combinators do not quite

guarantee round-tripping properties by construction, because our framework allows arbitrary functions to be applied independently in each direction.

However, we show how to recover round-tripping properties "almost by construction": the proof obligations reduce to checking a semi-inverse relation between "forward" and "backward" mappings. We argue that these conditions are "domain-agnostic": we can prove them by equational reasoning, relying on the laws of monadic profunctors, without knowledge of the domain-specific instance of monadic profunctor in use.

We illustrate this reasoning on logic programming examples. Starting from a monad of backtracking search as the "forward" interpretation, our framework enables a "backward" interpretation of programs as checkers, validating the correctness of an input solution.

Reconciling Partial and Local Invertibility

Anders Ågren Thuné, Uppsala University, Sweden

Invertible (i.e., injective) functions form a particular subset of bidirectional transformations, where the result of composing the forward and backward directions must be the identity where defined. Programming languages for invertible transformations commonly impose strong restrictions to guarantee a direct decomposition into invertible steps, which limits the programming flexibility (e.g., [36]). On the other hand, recent work allows unidirectional and invertible program fragments to be mixed more freely [26], but obscures the correspondence to a locally invertible computation model.

In this talk, we describe our recent work [33] which aims to reconcile these approaches. In particular, we showcase four core programming constructs for expressive partially invertible computation, and explain how they can be lowered compositionally into a locally invertible setting. The core constructs are:

- partially invertible branching
- pinning invertible data
- partially invertible composition
- abstraction and application of invertible functions.

We demonstrate these constructs in the context of an invertible functional programming language, and outline how they can be interpreted using a foundation of reversible arrows [18, 23] representing functions $A \to B$ and parametrised bijections $A \to (B \leftrightarrow C)$. As an example, we illustrate how to interpret the composition of a function and a parametrised bijection (which is again a parametrised bijection), using a conjugation construction found in earlier work.

Finally, we point towards the potential of applying our approach to program classical state-based lenses.

Understanding Delta Lenses using Category Theory

Bryce Clarke, Tallinn University of Technology, Estonia

In this talk, I will demonstrate how category theory provides a useful framework to understand the properties of delta lenses. In particular, I will show how

delta lenses may be constructed in a variety of ways: as algebras for a monad, coalgebras for a comonad, and as certain indexed collections of sets. The close relationships with split opfibrations (c-lenses) and retrofunctors (cofunctors) will also be explored.

Bidirectional Live Programming Supporting Multiple Programming Paradigms

Xing Zhang, Peking University, China

Bidirectional live programming (BLP) [40] is an emerging paradigm that enables real-time synchronisation between program code and its output, allowing developers to edit either representation while changes are automatically propagated in both directions. By bridging the gap between textual programming and direct manipulation, BLP offers significant advantages for interactive development in domains like GUI design, document typesetting, and interactive application development. However, current BLP approaches face critical limitations in supporting mainstream programming paradigms and providing flexible customisation of direct manipulation operations. Our work presents a comprehensive solution that advances BLP's practicality through innovations in both programming language support and interactive editing capabilities.

We introduce BiFJ, an object-oriented BLP language that extends Java with bidirectional semantics, enabling live programming for imperative OOP code while maintaining correctness guarantees. To support early-stage development, we propose techniques for handling incomplete programs through bidirectional previews that work even with missing code fragments. For direct manipulation, we develop Delta, a domain-specific language that allows declarative specification of custom editing operations, along with a novel fusion algorithm that seamlessly fuses these operations with the underlying program. We further extend this approach to support function manipulation and lazy evaluation, enabling BLP for dynamic behaviors in interactive applications. Our implementations (Bidirectional Preview [39], BiOOP [38], FuseDM [40], FunDM) demonstrate the framework's effectiveness across diverse case studies spanning web development and graphic design.

Meeting Schedule

Check-in Day: May 25 (Sun)

- Welcome Banquet

Day 1: May 26 (Mon)

- Welcome and Setting the Goals of the Meeting
- Round of Extended Introductions
- Talks and Discussions

Day 2: May 27 (Tue)

- Refining Goals/Topics of the Meeting
- Talks and Discussions
- Excursion and Main Banquet

Day 3: May 28 (Wed)

- Refining Goals/Topics of the Meeting
- Talks and Discussions
- Group Photo Shooting
- Plenary Discussion and Forming of Working Groups
- Working Group Discussions
- Working Group Plenary Reports

Day 4: May 29 (Thu)

- Working Group Discussions
- Working Group Plenary Reports
- Wrap up and Future Plans Discussion

Summary of Discussions

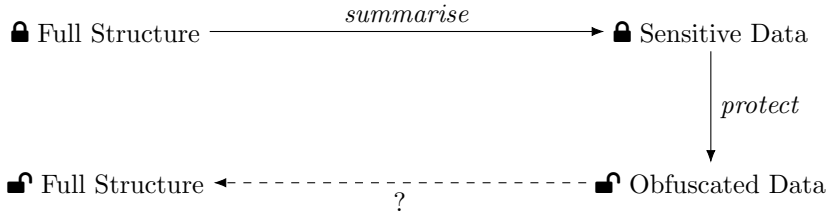
Differential Privacy

Samantha Frohlich and Anders Miltner et al

Goal

Differential privacy is concerned with allowing analysis of sensitive data sets without compromising individual privacy. For example, consider the case where a streaming service company wants to analyse the behaviour of its users. The streaming service wants to use this real data to understand what content is popular. However, for the user, it is essential that such an analysis does not leak information about them as an individual. A differentially private algorithm should be able to perform this analysis such that data of one individual does not change the result of the analysis as a whole, thus protecting the privacy of the individual.

Naturally, writing differentially private algorithms is hard. This is especially true for structured data such as graphs, because it is hard to untangle the private from the relevant. Current differentially private algorithms over such structures are complex and hard to maintain due to them being sensitive to changes in the structure. A solution is to *summarise* the graph into a simpler form that we know how to *protect* and publish that for analysis. However, this approach is lossy. What we really want is a way to re-add the lost structure to the now obfuscated and protected data, and publish that full and obfuscated structure for analysis.



For those familiar with the field of bidirectional programming (BX), this set-up will look very familiar. As such, it is our hope that BX can help solve this problem. Specifically, we hope that BX can fill in the backward transformation from obfuscated data to a full and obfuscated structure. The challenge will be ensuring that this transformation preserves the differential privacy provided by the pre-existing differential privacy algorithm (*protect*).

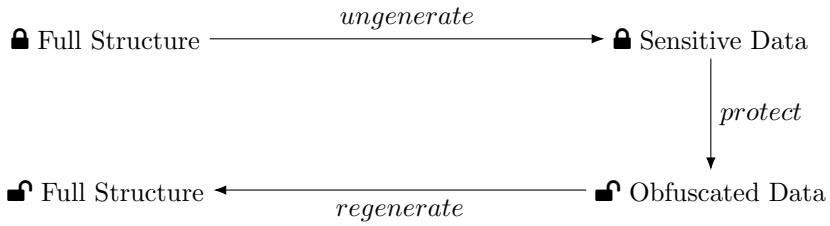
Summary.

This working group responds to questions posed by two of the talks: Miltner was looking to apply BX to the domain of differential privacy; Frohlich was looking for more application domains for reflective generators [15].

The group established that this was indeed a good fit: reflective generators seem like a promising tool for differential privacy. In our discussions, we narrowed down the specific domain that this would be useful in; established a novel example that this would enable; and planned next steps.

Bidirectional Differential Privacy will be useful in the case that the structure of the data is pertinent to its analysis. (We ruled out the case where only the contents of structure was relevant, because in that case the problem can be solved by using *fmap*.) In particular, we hope that applying reflective generators and the ideas of BX to the domain of differential privacy will provide a framework for creating differentially private algorithms over structures that promotes modularity and maintainability. This framework should allow for the reuse of existing algorithms, and create algorithms that are adaptable to changes in the structure.

We picture this working as follows:



The reflective generator will be used to *ungenerate* the private information (potentially with some summaries of the structure that are not sensitive to help with the regeneration), which will then be obfuscated using existing differentially private algorithm(s), this now publishable data will be used to *regenerate* a full structure, which will be used for more accurate analysis.

When the structure underlying the data are (different kinds of) graphs or trees, triple graph grammars (TGGs; [32]), another established BX formalism, might constitute an alternative approach to obtaining the *ungenerate* and *regenerate* functions. Via a TGG one would – in a declarative, rule-based manner – define how both structures correspond to each other and then automatically obtain derived rules that can be used to translate between both structures (in both directions). The central research question here is how to *orchestrate the derived rules* to define a suitable *regenerate* function. Classically, TGGs can be used for batch translation [32], which would amount to *regenerating* with just the public obfuscated data as input. This approach might result in public data that invalidates analysis results because it does not resemble the original private data enough. Alternatively, incremental translations have been developed (e.g., [12, 14]) that, in our case, would take the public obfuscated data, the *ungenerated* sensitive data and the original full data into account to define *regenerate* and compute the public full data. Applying such incremental approaches carelessly might invalidate the *protect* step, i.e., leak private data.

We believe that such a set-up, be it based on reflective generators or TGGs, will be useful in the following example that is currently unsolved in the domain of differential privacy. Consider the case where there already exists a differentially private algorithm for obfuscating the edges of nodes in a graph, and we want to apply this to all but one special node in a graph. Perhaps the graph in question contains information about the patients of a particular hospital and their connections to different medical centres and other patients. Here private information is the edges of the patient nodes, and the special node is the hospital, which has an edge connecting it to all of the patients. In this case, we want to hide the edges of all the patient nodes using the existing algorithm, then re-add the hospital node with connections to every other node in the *regenerate* step

because this structure is essential to the graph, but not private.

The next step for this project is to take the idea to the differential privacy experts and work with them to create a solution that preserves privacy. We are interested in exploring both mentioned approaches for feasibility and also in comparing their respective strengths and weaknesses.

Weak Well-Behavedness

David Jaz Myers et al

We discussed Prof. Matsuda's *partial state lenses* (ps-lenses), and noticed a categorical formulation which very closely matches the rules in Prof. Matsuda's slides. In particular, we see the ps-stability as a form of PutPut law.

The setting is partial orders; both view type V and state type S are partial orders.

Definition 1. Let $(S, \leq)$ and $(V, \leq)$ be partially ordered sets. A ps-lens between them consists of two functions:

$$\begin{cases} \mathsf{get} : S \to V \\ \mathsf{put} : S \times V \to S \end{cases}$$

There are three laws considered:

1. (ps-consistency; PutGet)

$$\frac{\mathsf{put}(s, v) \leq s'}{v \leq \mathsf{get}(s')}$$

2. (ps-acceptability; GetPut)

$$\frac{v \leq \mathsf{get}(s)}{\mathsf{put}(s, v) \leq s}$$

3. (ps-stability; PutPut)

$$\frac{\mathsf{put}(s, v_2) \leq s' \qquad v_2 \leq v_1 \leq \mathsf{get}(s')}{\mathsf{put}(s, v_2) \leq \mathsf{put}(s', v_1)}$$

A ps-lens is said to be well behaved when it satisfies ps-acceptability (GetPut) and ps-consistency (PutGet). It is said to be very well behaved if it also satisfies ps-stability.

What we noticed is that this sort of laws and their rules can be well captured (up to a slight difference that we don't fully understand) by generalizing the work of Johnson, Rosebrugh, and Wood [24].

Theorem 2 (Johnson-Rosebrugh-Wood)**.** *Fix a view type $V \in \mathcal{C}$ in a cartesian category $\mathcal{C}$. We have an adjunction*

$$\mathcal{C} \downarrow V \underset{\pi_2 : - \times V \to V}{\overset{\mathsf{dom}}{\rightleftarrows}} \mathcal{C}$$

This induces a monad $(\mathsf{get} : S \to V) \mapsto (\pi_2 : S \times V \to V)$ *on the slice category* $\mathcal{C} \downarrow V$*. The structure of a very lawful lens on a get function* $\mathsf{get} : S \to V$ *is precisely the structure of an* algebra *for the induced monad on* S*. Namely:*

1. *The algebra structure is a morphism*

$$\begin{array}{ccc} S \times V & \xrightarrow{\mathsf{put}} & S \\ {\scriptstyle \pi_2}\downarrow & & \downarrow{\scriptstyle \mathsf{get}} \\ V & = & V \end{array}$$

in $\mathcal{C} \downarrow V$*. The commutativity of the above square is the PutGet law:*

$$v = \mathsf{get}(\mathsf{put}(s, v)).$$

2. *The unit law*

$$\begin{array}{ccc} S & & \\ {\scriptstyle (\mathsf{id},\mathsf{get})}\downarrow & \searrow & \\ S \times V & \xrightarrow[\mathsf{put}]{} & S \end{array}$$

is the GetPut law:

$$\mathsf{put}(s, \mathsf{get}(s)) = s.$$

3. *The multiplication law*

$$\begin{array}{ccc} (S \times V) \times V & \xrightarrow{\mathsf{put} \times V} & S \times V \\ {\scriptstyle \pi_{1,3}}\downarrow & & \downarrow{\scriptstyle \mathsf{put}} \\ S \times V & \xrightarrow[\mathsf{put}]{} & S \end{array}$$

is the PutPut law:

$$\mathsf{put}(s, v_2) = \mathsf{put}(\mathsf{put}(s, v_1), v_2).$$

Our idea is to extend this to ordered sets *colaxly*. We make the following definition:

Definition 3. Let $V \in \mathcal{O}\mathsf{rd}$ be a partially ordered set. Consider the colax slice 2-category $\mathcal{O}\mathsf{rd} \downarrow_{\mathsf{colax}} V$ whose objects are monotone maps $\alpha : X \to V$ and whose morphisms are diagrams

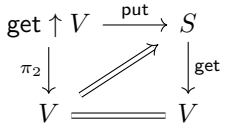
$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ {\scriptstyle \alpha}\downarrow & \nearrow & \downarrow{\scriptstyle \beta} \\ V & = & V \end{array}$$

that is, monotone maps $f : X \to Y$ such that $\alpha(x) \leq \beta(f(x))$ for all $x \in X$. The 2-cells $f \Rightarrow g$ express that $f(x) \leq g(x)$ for all $x \in X$. The operation which assigns $\alpha : X \to V$ to $\pi_2 : \alpha \uparrow V \to V$ where

$$\alpha \uparrow V := \{(x, v) \in X \times V \mid \alpha(x) \geq v\}$$

is a 2-monad on this 2-category with unit given by $\pi_1 : \alpha \uparrow V \to X$ and multiplication $\pi_{1,3} : \alpha \uparrow V \uparrow V \to \alpha \uparrow V$ (that is, $\pi_{1,3}(x, v_1, v_2) = (x, v_2)$). A *colax algebra* structure on a get map $\mathsf{get} : S \to V$ is the structure of a *colax lens*:

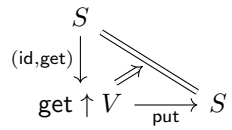
1. The algebra structure is a morphism



   in $\mathcal{O}\mathsf{rd} \downarrow V$. That is, $\mathsf{put}(s, v)$ is defined when $v \leq \mathsf{get}(s)$, and the colaxitor of the above square is the colax PutGet law: for all $v \leq \mathsf{get}(s)$,

$$v \leq \mathsf{get}(\mathsf{put}(s, v)).$$

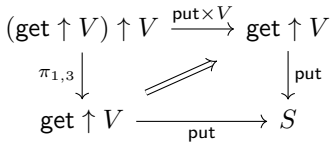
2. The unit law



   is the colax GetPut law: for all $s \in S$,

$$\mathsf{put}(s, \mathsf{get}(s)) \leq s.$$

3. The multiplication law



   is the colax PutPut law: for all $v_2 \leq v_1 \leq \mathsf{get}(s)$,

$$\mathsf{put}(s, v_2) \leq \mathsf{put}(\mathsf{put}(s, v_1), v_2).$$

If instead we use the 2-monad $(\mathsf{get} : S \to V) \mapsto (\pi_2 : S \times V \to V)$, then we get the same structure but without the constraint that $v \leq \mathsf{get}(s)$ in the put and the quantification for the PutPut law is over all v_2, v_1, and s with no relation between them.

We can now compare the notion of a colax lens with that of a ps-lens.

Theorem 4. *Let S and V be partial orders and $\mathsf{get} : S \to V$ and $\mathsf{put} : S \times V \to V$ be functions. Then*

1. *If these satisfy ps-consistency, then they satisfy the colax PutGet law; the converse holds if* get *is monotone.*
2. *If these satisfy ps-acceptability, then they satisfy the colax GetPut law; the converse holds if* put *is monotone in the second variable.*

Moreover, if get *and* put *satisfy both ps-consistency and ps-acceptability, then* get *is monotone and* put *is monotone in the second variable. Therefore,* get *and* put *satisfy both ps-consistency and ps-acceptibility if and only if* get *is monotonic,* put *is monotonic in the second variable, and they satisfy both colax PutGet and colax GetPut.*

Proof. 1. Suppose that get and put satisfy ps-consistency; then since $\mathsf{put}(s, v) \leq \mathsf{put}(s, v)$, we conclude that $v \leq \mathsf{get}(\mathsf{put}(s, v))$, which is the colax PutGet law. Conversely, suppose that get is monotone and that $\mathsf{put}(s, v) \leq s'$; colax PutGet says that $v \leq \mathsf{get}(\mathsf{put}(s, v))$, so we therefore conclude that $v \leq \mathsf{get}(s')$.

2. Suppose that get and put satisfy ps-acceptability; then since $\mathsf{get}(s) \leq \mathsf{get}(s)$, we conclude that $\mathsf{put}(s, \mathsf{get}(s)) \leq s$, which is the colax GetPut law. Conversely, suppose that put is monotone in the second variable and that $v \leq \mathsf{get}(s)$; then the colax GetPut law says that $\mathsf{put}(s, \mathsf{get}(s)) \leq s$, so we conclude $\mathsf{put}(s, v) \leq s$.

Suppose that get and put satisfy both ps-consistancy and ps-accetability.

- Suppose that $s \leq s'$. We know that $\mathsf{put}(s, \mathsf{get}(s)) \leq s'$ by ps-acceptability, and so $\mathsf{get}(s) \leq \mathsf{get}(s')$ by ps-consistency, proving that get is monotonic.
- Suppose that $v \leq v'$. Then $v' \leq \mathsf{get}(\mathsf{put}(s, v'))$ by ps-consistency, and so $\mathsf{put}(s, v) \leq \mathsf{put}(s, v')$ by ps-acceptability, showing that put is monotonic in the second variable.

□

I have not put in ps-stability and colax PutPut in the above theorem because they only *almost* correspond. Namely, setting $s' = \mathsf{put}(s, v)$ in ps-stability gives us

$$\frac{v_2 \leq v_1 \leq \mathsf{get}(s)}{\mathsf{put}(s, v_2) \leq \mathsf{put}(\mathsf{put}(s, v_2), v_1)}$$

whereas the colax PutPut rule is

$$\frac{v_2 \leq v_1 \leq \mathsf{get}(s)}{\mathsf{put}(s, v_2) \leq \mathsf{put}(\mathsf{put}(s, v_1), v_2)}$$

Note the change in order of v_1 and v_2 at the end. However, the main use of ps-stability is that it is compositional and that it implies the PutGetPut law:

$$\mathsf{put}(\mathsf{put}(s, v), \mathsf{get}(\mathsf{put}(s, v))) = \mathsf{put}(s, v).$$

However, this also follows from colax PutPut (with colax PutGet and colax GetPut) so long as put is monotone in the second variable. Colax PutGet implies that $v \leq \mathsf{get}(\mathsf{put}(s, v))$ so if put is monotone in the second variable we then conclude that $\mathsf{put}(s, v) \leq \mathsf{put}(\mathsf{put}(s, v), v) \leq \mathsf{put}(\mathsf{put}(s, v), \mathsf{get}(\mathsf{put}(s, v)))$. Conversely, by colax GetPut we have $\mathsf{put}(\mathsf{put}(s, v), \mathsf{get}(\mathsf{put}(s, v))) \leq \mathsf{put}(s, v)$. Therefore, so long as this condition is also composable, the colax PutPut condition will suffice to imply the same stability property as ps-stability.

We therefore turn our attention to composition of colax lenses. Let's make a definition of colax lens which is separate from the abstract discussion above.

Definition 5. A (very) well behaved *colax lens* $(\mathsf{get}, \mathsf{put}) : S \leftrightarrows V$ *between orders* S *and* V *consists of functions*

$$\begin{cases} \mathsf{get} : S \to V \\ \mathsf{put} : S \times V \to S \end{cases}$$

so that get *is monotonic,* put *is monotonic in the second variable, and*

1. *They satisfy the colax PutGet law:*

$$v \leq \mathsf{get}(\mathsf{put}(s, v)).$$

   Or, equivalently, ps-consistency.

2. *They satisfy the colax GetPut law:*

$$\mathsf{put}(s, \mathsf{get}(s)) \leq s.$$

   Or, equivalently, ps-acceptability.

3. *(If "very",) they satisfy the PutPut law: for* $v_2 \leq v_1 \leq \mathsf{get}(s)$,

$$\mathsf{put}(s, v_2) \leq \mathsf{put}(\mathsf{put}(s, v_1), v_2)$$

DSL for Programming with Delta Lenses

Kazutaka Matsuda and Vadim Zaytsev et al

This working group aimed to explore the design space for domain-specific languages (DSLs) for programming with (asymmetric) delta lenses [10]. To this end, we began by reviewing simple motivating examples that delta lenses address.

Example 6 (Projection)**.** Consider that the source is a set of name-age pairs, such as

$$s = \{(\texttt{"Alice"}, 25), (\texttt{"Bob"}, 37)\}$$

and that the view is obtained from the source by gathering only names:

$$v = \{\texttt{Alice}, \texttt{Bob}\}$$

When the view is changed to the view v'

$$v' = \{\texttt{Aurora}, \texttt{Bob}\}$$

there are several possibilities for how this change is made, including

(1) `Alice` is removed and a new person named `Aurora` is inserted; and

(2) `Alice` changes her name to `Aurora`.

Reasonable updated sources differ in (1) and (2): in (1), the source should be updated to $s' = \{(\texttt{"Aurora"}, 0), (\texttt{"Bob"}, 37)\}$, where 0 denotes an arbitrarily chosen default age; and in (2), the updated source should be $s'' = \{(\texttt{"Aurora"}, 25), (\texttt{"Bob"}, 37)\}$, keeping her age. □

Example 7 (Every second character)**.** The second example is more abstract but illustrates a similar concept to the first one. Consider that the source and view are both strings, where the view is obtained by extracting every second character from the source, such as `aaabbc` to `abc`. When the string `abc` is changed to `abbc`, there are several possibilities for how this change is made, such as the new `b` is inserted before or after the existing `b`. When we want to keep the correspondence of pairs of odd and even characters, expected view updating results differ depending on interpretations: when `b` is inserted before `b`, the updated source should be `aa␣babbc`, and when it is inserted after the existing `b`, the updated source should be `aaab␣bbc`, where ␣ is an arbitrarily chosen default character (which in this example can even be chosen to be `a` in order to keep the ambiguity if that is desirable; this is not always possible in general). □

When dealing only with states, we cannot distinguish these finer interpretations of updates. This demands operation-based formalisations, such as delta lenses. Also, often deltas are smaller than states, and thus mapping deltas tends to be more efficient than state mapping.

Then, we review existing formalisations for delta lenses and related concepts. We focus only on the backward delta propagation, which is the key to disambiguating update interpretations in the above examples. In this view, delta lenses correspond to update-update lenses [1]. That is, a delta lens is a

(certain) pair of a *functor* — mapping of forward deltas — and a *retrofunctor* [29, Chapter 7] (also known as a cofunctor) — mapping of backward deltas — while an update-update lens is modelled only by a retrofunctor [1, 6, 7]. Like a functor, a retrofunctor connects two categories C and D and maps objects in C into those in D, while, unlike a functor, it maps morphisms in D into C. More specifically, given an object A in C, it sends a morphism $u : FA \to B'$ in D for some object B to a morphism $F_A u \in A \to A'$ where $FA' = B'$, preserving composition and identities. Take C and D as categories where objects are states and morphisms are updates. Then, the object mapping of a retrofunctor corresponds to a *get* while the morphism mapping corresponds to a delta-based *put*. This formalisation is simple and elegant, but its implementation presents several technical challenges:

- A straightforward implementation requires dependent types [20], as the backward transformation maps deltas that start from the original source. However, dependent types are not currently available for mainstream (even for functional) programming languages such as Haskell and OCaml.
- In category theory, a morphism comes with its domain and codomain — in our case, the states before and after the update (represented by the morphism). This means that the straightforward implementation of delta-based *put* also involves state-based *put* (though with an additional delta parameter), neglecting its advantage in efficiency. Also, this treatment conflicts with the practical situation that an update can be valid for several states.
- Many primitive lenses do not require the original source to determine delta mapping. Thus, for efficiency concerns, the computation of such original states should be avoided as much as possible.

Thus, there are design spaces depending on how we deal with the issues. To explore such design spaces, this working group during the seminar investigated some existing implementations of delta lenses — the module `Generics.Pointless.DLenses` [30] and update-update lenses — the part of the file `Elmlens.hs` [35] in Haskell, and related techniques in the literature. For example, the edit lens [21] addresses the third issue in the simply-typed setting by using the complement set.

Digital Twins

Romina Eramo et al

Digital Twins (DTs) are virtual representations of physical systems that continuously receive data from their real-world counterparts to simulate, monitor, and predict behaviours. They are used across domains such as manufacturing, healthcare, and smart cities to enable data-driven decision-making. Bidirectional Transformations (BX) play a key role in ensuring consistency between the physical system and its digital representation, or among the various interdependent artefacts within the digital twin itself. BX frameworks allow changes in one model or data view (e.g., sensor data, control parameters) to be safely and automatically propagated to related views, maintaining synchronisation and coherence across the system.

While the discussion has focused on DTs, several systems (e.g., adaptive systems, cyber-physical systems, real-time systems) share a common foundation in the continuous interaction between computational and physical components. They all involve dynamic models that evolve in response to changes in the environment or internal state, often requiring synchronisation between different representations or subsystems. This makes them highly suitable for the application of BX, which enables the consistent propagation of changes across interconnected views or models. For instance, BX can support synchronisation between monitored data and system state (as in DTs), policy adaptation (in adaptive systems), coordination of software and physical layers (in CPS and control systems), and predictable updates under timing constraints (in real-time systems). Composition plays a central role in integrating BX within DT. It enables the coherent combination of multiple digital views — such as data streams, control models, and simulations — into a unified representation of the physical system.

Composition also applies to transformations themselves, allowing modular, reusable synchronisation mechanisms between related artefacts. In complex systems, it supports both horizontal integration (across different models) and vertical alignment (across abstraction levels), ensuring consistency between specifications and implementations. Additionally, composition is essential for structuring system behaviour and communication, facilitating reliable propagation of changes between the physical and digital domains. Overall, it provides the foundation for scalability, modularity, and maintainability in DT–BX integration.

The formalisation of DTs can greatly benefit from frameworks, like Dejima [3, 19], a framework originally designed to manage updatable, consistent bidirectional views in databases, that uses BX to manage consistency between interconnected models. Within a DT, many components — such as sensor data, simulation models, control logic, and system configurations — can be encoded as BX to ensure changes in one layer are accurately and consistently reflected across others. This is especially relevant in multi-view or hierarchical models, where BX enables coherence between abstractions and their implementations. The integration of such transformations supports modular evolution and reliable synchronization within the dynamic structure of a DT.

However, applying BX in DTs introduces new challenges, particularly regarding Quality of Service (QoS) and asynchronous updates. Traditional BX

frameworks do not address key DT requirements like latency constraints, data fidelity, or non-blocking synchronisation. Existing tools offer useful benchmarking support for model transformations, but further extensions are needed to simulate real-world DT environments.

Computation Models

Keisuke Nakano et al

Goal

This group discussed the possibility of a computation model for BX, especially *well-behaved asymmetric lenses*. A well-behaved asymmetric lens is a pair of two functions, $get : S \to V$ and $put : S \times V \to S$, which are required to satisfy the (GetPut) and (PutGet) lens laws. To facilitate users in describing lenses that satisfy these laws, many domain-specific languages, so-called BX languages, have been proposed. The BX languages are designed so that the lenses described by users always satisfy lens laws (or can be automatically verified to do so) by imposing restrictions on syntax, types, and primitives. Due to these restrictions, some well-behaved asymmetric lenses might exist which are impossible to describe in the chosen BX language. To the best of our knowledge, no BX languages have been proven to be 'complete' in the sense that every computable lens can be expressed in the language. In this group, we shared ideas on a computation model for well-behaved asymmetric lenses, which is expected to serve as a reference model for BX languages, much like Turing machines and lambda calculus do for programming languages. We considered a Turing machine as the basis of a computation model instead of lambda calculus and other models because it has a reversible variant that can be inverted easily.

Summary

The discussion began with an introduction to computation models for two kinds of functions. One is an involutory function, which is a function f that satisfies $f(f(x)) = x$ for all x in the domain of f. The computation model for involutory functions is *Time-Symmetric Turing Machine* (TSTM) [28]. Another is an idempotent function, which is a function f that satisfies $f(f(x)) = f(x)$ for all x in the domain of f. The computation model for idempotent functions is *Idempotent Turing Machine* (ITM) [27]. In the discussion, we reviewed how these two computation models were designed to compute all and only computable involutory functions and idempotent functions, respectively. The expressiveness of both computation models is guaranteed based on the property that they are closed by conjugation with injective functions, *i.e.*, for any involutory (resp. idempotent) function f and any injective function g, the function $g^{-1} \circ f \circ g$ is also an involutory (resp. idempotent) function. This property is crucial for the syntactic restrictions imposed on ordinal Turing machines to design the computation models, TSTM and ITM. It has been shown successfully that every computable involutory function (resp. idempotent function) can be computed by a TSTM (resp. ITM) using the fact that the function $swap(x, y) = (y, x)$ (resp. $copy(x, y) = (y, y)$), can be implemented by a TSTM (resp. ITM).

The discussion then shifted to the design of a computation model for BX. According to a recent result [16], a well-behaved asymmetric lens can be characterised by a *put* function satisfying two properties:

$$put(s, v) \downarrow s' \wedge put(s, v') \downarrow s' \Rightarrow v = v' \qquad \text{(PutInjectivity)}$$
$$put(s, v) \downarrow s' \Rightarrow put(s', v) \downarrow s' \qquad \text{(PutIdempotency)}$$

for any $s, s' \in S$ and $v, v' \in V$, where the second has also been called the (PutTwice) law in the literature. A *put* function forms a well-behaved asymmetric lens with a (uniquely determined) *get* function if and only if it satisfies those two lens laws. After the discussion, we found that a well-behaved asymmetric lens can be characterised by a class of functions that are closed under conjugation with injective functions. Every function in the class corresponds to a *put* function that satisfies the two above laws. A computation model for the class of functions is expected to be designed similarly to the ITM, with the specialised *copy* function as its core operation, where further observation is required on the expressiveness of syntactically restricted reversible TM.

References

[1] D. Ahman and T. Uustalu. Taking updates seriously. In R. Eramo and M. Johnson, editors, *Proceedings of the 6th International Workshop on Bidirectional Transformations co-located with The European Joint Conferences on Theory and Practice of Software, BX@ETAPS 2017, Uppsala, Sweden, April 29, 2017*, volume 1827 of *CEUR Workshop Proceedings*, pages 59–73. CEUR-WS.org, 2017. URL: https://ceur-ws.org/Vol-1827/paper11.pdf.

[2] A. Anjorin, Z. Diskin, M. Wang, and Y. Xiong, editors. *NII Shonan Meeting Report No. 091 (2016-13): Bidirectional Transformations (BX Shonan)*. National Institute of Informatics, 2016. URL: https://shonan.nii.ac.jp/docs/No-091.pdf.

[3] Y. Asano, Y. Cao, S. Hidaka, Z. Hu, Y. Ishihara, H. Kato, K. Nakano, M. Onizuka, Y. Sasaki, T. Shimizu, M. Takeichi, C. Xiao, and M. Yoshikawa. Bidirectional Collaborative Frameworks for Decentralized Data Management. In G. Fletcher, K. Nakano, and Y. Sasaki, editors, *Software Foundations for Data Interoperability*, pages 13–51, Cham, 2022. Springer. doi:10.1007/978-3-030-93849-9_2.

[4] E. C. Brady. Idris 2: Quantitative Type Theory in Practice. In A. Møller and M. Sridharan, editors, *Proceedings of the 35th European Conference on Object-Oriented Programming (ECOOP)*, volume 194 of *LIPIcs*, pages 9:1–9:26. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ECOOP.2021.9.

[5] M. Capucci and D. J. Myers. Contextads as Wreaths; Kleisli, Para, and Span Constructions as Wreath Products, 2024. arXiv:2410.21889.

[6] B. Clarke. Internal lenses as functors and cofunctors. In J. Baez and B. Coecke, editors, *Proceedings Applied Category Theory 2019, ACT 2019, University of Oxford, UK, 15-19 July 2019*, volume 323 of *EPTCS*, pages 183–195, 2020. doi:10.4204/EPTCS.323.13.

[7] B. Clarke. Delta lenses as coalgebras for a comonad. In L. Iovino and L. M. Kristensen, editors, *STAF 2021 Workshop Proceedings: 9th International Workshop on Bidirectional Transformations, Joint Workshop on Foundations and Practice of Visual Modeling and Data for Model-Driven Engineering, International workshop on MDE for Smart IoT Systems, 4th International Workshop on (Meta)Modeling for Healthcare Systems, and 20th International Workshop on OCL and Textual Modeling co-located with Software Technologies: Applications and Foundations, Federation of Conferences (STAF 2021), Virtual Event / Bergen, Norway, June 21-25, 2021*, volume 2999 of *CEUR Workshop Proceedings*, pages 18–27. CEUR-WS.org, 2021. URL: https://ceur-ws.org/Vol-2999/bxpaper2.pdf.

[8] A. Cleve, E. Kindler, P. Stevens, and V. Zaytsev, editors. *Report from Dagstuhl Seminar 18491 on Multidirectional Transformations and Synchronisations*. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2019. doi:10.4230/DagRep.8.12.1.

[9] K. Czarnecki, J. N. Foster, Z. Hu, R. Lämmel, A. Schürr, and J. R. Terwilliger. Bidirectional Transformations: A Cross-Discipline Perspective. In R. F. Paige, editor, *Theory and Practice of Model Transformations (ICMT)*, volume 5563 of *LNCS*, pages 260–283. Springer, 2009. doi:10.1007/978-3-642-02408-5_19.

[10] Z. Diskin, Y. Xiong, and K. Czarnecki. From state- to delta-based bidirectional model transformations: the asymmetric case. *J. Object Technol.*, 10:6: 1–25, 2011. URL: https://doi.org/10.5381/jot.2011.10.1.a6, doi:10.5381/JOT.2011.10.1.A6.

[11] S. Fischer, Z. Hu, and H. Pacheco. A Clear Picture of Lens Laws. In R. Hinze and J. Voigtländer, editors, *Mathematics of Program Construction*, pages 215–223, Cham, 2015. Springer. doi:10.1007/978-3-319-19797-5_10.

[12] L. Fritsche, J. Kosiol, A. Schürr, and G. Taentzer. Avoiding Unnecessary Information Loss: Correct and Efficient Model Synchronization based on Triple Graph Grammars. *International Journal on Software Tools for Technology Transfer, FASE 2019 Special Issue*, 23(3):335–368, 2021. doi:10.1007/S10009-020-00588-7.

[13] J. Gibbons, R. F. Paige, A. Schürr, J. F. Terwilliger, and J. Weber, editors. *Bi-directional transformations (bx) – Theory and Applications Across Disciplines*. Banff International Research Station (BIRS) for Mathematical Innovation and Discovery, 2013. URL: https://archive.birs.ca/files/2013/13w5115/report13w5115.pdf.

[14] H. Giese and S. Hildebrandt. Efficient Model Synchronization of Large-Scale Models. Technical Report 28, Hasso-Plattner-Institut, 2009. URL: https://publishup.uni-potsdam.de/frontdoor/index/index/docId/2883.

[15] H. Goldstein, S. Frohlich, M. Wang, and B. C. Pierce. Reflecting on Random Generation. *Proceedings of the ACM on Programming Languages*, 7(ICFP), Aug. 2023. doi:10.1145/3607842.

[16] K. Hashiba, K. Nakano, K. Asada, and K. Kikuchi. Characterizations of Partial Well-Behaved Lenses. In G. Allais and Y. A. Liu, editors, *Proceedings of the 2025 ACM SIGPLAN International Workshop on Partial Evaluation and Program Manipulation (PEPM)*, pages 43–53. ACM, 2025. doi:10.1145/3704253.3706139.

[17] X. He and T. Zan. BIT: A Template-based Approach to Incremental and Bidirectional Model-to-Text Transformation. *Journal of Systems and Software*, 216:112148, 2024. doi:10.1016/J.JSS.2024.112148.

[18] C. Heunen, R. Kaarsgaard, and M. Karvonen. Reversible Effects as Inverse Arrows. In S. Staton, editor, *Proceedings of the 34th Conference on the Mathematical Foundations of Programming Semantics (MFPS)*, volume 341 of *ENTCS*, pages 179–199. Elsevier, 2018. doi:10.1016/J.ENTCS.2018.11.009.

[19] S. Hidaka. *Evolutionary Framework and Conflict Resolution for Multidirectional Transformations*, pages 25–36. Springer, Singapore, 2025. doi:10.1007/978-981-97-6429-7_2.

[20] M. Hofmann. *Syntax and semantics of dependent types*, pages 13–54. Springer, London, 1997. doi:10.1007/978-1-4471-0963-1_2.

[21] M. Hofmann, B. C. Pierce, and D. Wagner. Edit lenses. In J. Field and M. Hicks, editors, *Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012, Philadelphia, Pennsylvania, USA, January 22-28, 2012*, pages 495–508. ACM, 2012. doi:10.1145/2103656.2103715.

[22] Z. Hu, A. Schürr, P. Stevens, and J. Terwilliger, editors. *Report from Dagstuhl Seminar 11031 on Bidirectional Transformations (bx)*. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2011. doi:10.4230/DagRep.1.1.42.

[23] R. P. James and A. Sabry. Information Effects. In J. Field and M. Hicks, editors, *Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 73–84. ACM, 2012. doi:10.1145/2103656.2103667.

[24] M. Johnson, R. Rosebrugh, and R. Wood. Algebras and Update Strategies. *Journal of Universal Computer Science*, 16(5):729–748, 2010. doi:10.3217/jucs-016-05-0729.

[25] S. Libkind and D. J. Myers. Towards a Double Operadic Theory of Systems, 2025. arXiv:2505.18329.

[26] K. Matsuda and M. Wang. Sparcl: A Language for Partially-Invertible Computation. *Proceedings of the ACM on Programming Languages*, 4(ICFP):118:1–118:31, 2020. doi:10.1145/3409000.

[27] K. Nakano. Idempotent Turing Machines. In F. Bonchi and S. J. Puglisi, editors, *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 202 of *LIPIcs*, pages 79:1–79:18. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.MFCS.2021.79.

[28] K. Nakano. Time-Symmetric Turing Machines for Computable Involutions. *Science of Computer Programming*, 215:102748, 2022. doi:10.1016/J.SCICO.2021.102748.

[29] N. Niu and D. I. Spivak. Polynomial functors: A mathematical theory of interaction, 2024. URL: https://arxiv.org/abs/2312.00990, arXiv:2312.00990.

[30] H. Pacheco and A. Cunha. pointless-lenses: Pointless lenses library. Hackage, https://hackage.haskell.org/package/pointless-lenses, 2012.

[31] P. Schultz, D. I. Spivak, and C. Vasilakopoulou. Dynamical Systems and Sheaves, 2019. arXiv:1609.08086.

[32] A. Schürr. Specification of Graph Translators with Triple Graph Grammars. In E. W. Mayr, G. Schmidt, and G. Tinhofer, editors, *Proceedings of the 20th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 903 of *LNCS*, pages 151–163. Springer, 1994. doi:10.1007/3-540-59071-4_45.

[33] A. Å. Thuné, K. Matsuda, and M. Wang. Reconciling Partial and Local Invertibility. In S. Weirich, editor, *Proceedings of the 33rd European Symposium on (ESOP), Part II*, volume 14577 of *LNCS*, pages 59–89. Springer, 2024. doi:10.1007/978-3-031-57267-8_3.

[34] Y. Wakuta, M. Mior, T. Zenmyo, Y. Sasaki, and M. Onizuka. NoSQL Schema Design for Time-Dependent Workloads, 2023. arXiv:2303.16577.

[35] Z. Ye. elmapp: An functional and compositional approach to web applications via bidirectional transformation. GitHub, https://github.com/peter-jerry-ye/elmapp, 2023.

[36] T. Yokoyama, H. B. Axelsen, and R. Glück. Principles of a Reversible Programming Language. In A. Ramírez, G. Bilardi, and M. Gschwind, editors, *Proceedings of the Fifth Conference on Computing Frontiers (CF)*, pages 43–54. ACM, 2008. doi:10.1145/1366230.1366239.

[37] R. Yoshida, Y. Sasaki, C. Xiao, Y. Ishihara, Y. Asano, and M. Onizuka. *Transaction Management in Collaborative Data Management*, pages 93–119. Springer, 2025. doi:10.1007/978-981-97-6429-7_5.

[38] X. Zhang, G. Guo, X. He, and Z. Hu. Bidirectional Object-Oriented Programming: Towards Programmatic and Direct Manipulation of Objects. *Proceedings of the ACM on Programming Languages*, 7(OOPSLA1):230–255, 2023. doi:10.1145/3586035.

[39] X. Zhang and Z. Hu. Towards Bidirectional Live Programming for Incomplete Programs. In *Proceedings of the 44th IEEE/ACM International Conference on Software Engineering (ICSE)*, pages 2154–2164. ACM, 2022. doi:10.1145/3510003.3510195.

[40] X. Zhang, R. Xie, G. Guo, X. He, T. Zan, and Z. Hu. Fusing Direct Manipulations into Functional Programs. *Proceedings of the ACM on Programming Languages*, 8(POPL):1211–1238, 2024. doi:10.1145/3632883.