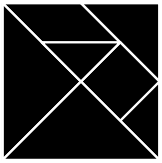
GSDLAB TECHNICAL REPORT

An algebraic semantics for bidirectional model synchronization

Zinovy Diskin

GSDLAB–TR 2014-04-01 August 2014



Generative Software
Development Lab

Generative Software Development Laboratory
University of Waterloo
200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1

WWW page: http://gsd.uwaterloo.ca/

A Product Line of Lax Lenses.

Zinovy Diskin[1,2]

Abstract

The goal of the present document is to support the taxonomy for bidirectional model synchronization developed in [1] with a formal semantics. The taxonomy is 3D so that each synchronization type is characterized by a triple of coordinates (x, y, z), in which x classifies the *organizational symmetry* of the case, y is for the *informational symmetry*, and z is for *incrementality* of the update propagation operations. Different types of delta lenses (algebraic structures modeling bx) can be classified by points on the YZ-plane. As for the x-coordinate, it says, roughly, whether update propagation is uni- or bi-directional, but as it was shown in [1], there are several important refinements of the two-valued uni, bi-classification so that actually axis X has four rather than two points.

A formal semantics for the enriched X-classification seems to be an entirely novel aspect for the (delta) lens literature. There are also several contributions for the very delta lens framework within the plane YZ. First of all, we build a product line of delta lenses so that each concrete delta lens structure is characterizes by two parameters. To this end, we will build a framework in which an asymmetric delta lens appears as a special case of the symmetric one, which is in a sense dual to the Johnson and Rosebrugh construction, in which a symmetric lens is presented as a span of asymmetric lenses [2]. The second novelty is more essential: we present lax versions of major lens laws of compositionality (the infamous PutPut), and invertibility.

Keywords: Synchronization Taxonomy, Formal semantics, Model Synchronization, Model Transformation, Model Driven Engineering

1. Introduction

The goal of the present document is to support the taxonomy for bidirectional model synchronization developed in [1] with a formal semantics. The taxonomy is 3D so that each synchronization type is characterized by a triple of coordinates (x, y, z), in which x classifies the *organizational symmetry* of the case, y is for the *informational symmetry*, and z is for *incrementality* of the update propagation operations. It is convenient to present the space as a product $X|timYZ$, where X is the org-symmetry axis and $YZ = Y \times Z$ is the plain of info-symmetry and incrementality, which actually classifies the computational framework (= a system of update propagation operations) supporting the synchronization. Different types of delta lenses (algebraic structures modeling bx) can be classified by points on this plane. As for the x-coordinate of the cases placed in axis X, roughly, it says whether update propagation is uni- or bi-directional, but as it was shown in [1], there are several important refinements of the two-valued uni, bi-classification so that actually axis X has four rather than two points.

A formal semantics for the enriched X-classification seems to be an entirely novel aspect for the (delta) lens literature. There are also several contributions for the very delta lens framework within the plane YZ. First of all, we build a product line of delta lenses so that each concrete delta lens structure is characterizes by two parameters. To this end, we will build a framework in which an asymmetric delta lens appears as a special case of the

Email address: zdiskin@gsd.uwaterloo.ca (Zinovy Diskin)
[1]University of Waterloo, Canada
[2]McMaster University, Canada

symmetric one, which is in a sense dual to the Johnson and Rosebrugh construction, in which a symmetric lens is presented as a span of asymmetric lenses [2]. The second novelty is more essential: we present lax versions of major lens laws of compositionality (the infamous Put-Put), and invertibility. The semantics is algebraic and essentially based on operations over models, updates, and intermodel correspondence mappings—all considered as abstract nodes and arrows.

We will define a family of algebraic structures using the following common pattern. Let the structure to be defined is S. We first define the carrier of S (as a rule, it will be a previously defined structure with, perhaps, new sets added), on which several new operations that S has to have are defined. To specify these operations, we give their arities, and, perhaps, some elementary technical equations (we say *laws*) they must satisfy (some of these laws will be skipped to save space). Then we introduce substantial laws capturing semantics of the operations, and say that S is *well-behaved* if its operations satisfy these laws. The presentation is necessarily abstract, although informal explanations and intuitive meaning of the constructs are provided. Concrete examples illustrating how these abstract constructs work can be found in papers [3, 4, 5]; some of the constructs were implemented with TGG [6, 7].

2. Formalizing Informational Symmetry: Alignment Frameworks with Consistency

2.1. Model spaces and alignment.

Definition 1 (Model spaces). A *model space* is a category M, whose objects are called *models*, and morphisms (arrows) are *directed deltas* or *updates*. In detail, a model space is a directed graph $\mathsf{M} = (\mathsf{M}_\bullet, \mathsf{M}_\Delta, \mathsf{s}, \mathsf{t})$ with a set $\mathsf{M}_\bullet$ of nodes called *models*, a set M_Δ of arrows called *deltas*, and functions $\mathsf{s}\colon \mathsf{M}_\Delta \to \mathsf{M}_\bullet$, $\mathsf{t}\colon \mathsf{M}_\Delta \to \mathsf{M}_\bullet$, which assign a source and a target model to every delta. For a delta a with $\mathsf{s}(a) = A$ and $\mathsf{t}(a) = A'$, we write $a\colon A \to A'$ or else $a \in \mathsf{M}_\Delta(A, A')$, where $\mathsf{M}_\Delta(A, A')$ is the set of all deltas from A to A'.

Being a category means that the graph is endowed with two additional structures. First, arrows can be sequentially composed: for any pair $a_1\colon A \to A'$ and $a_2\colon A' \to A''$, there is defined their composition $a_1; a_2\colon A \to A''$, and the associativity law holds: $(a_1; a_2); a_3 = a_1; (a_2; a_3)$. Second, every model $A \in \mathsf{M}_\bullet$ is assigned with a special delta $\mathsf{id}_A\colon A \to A$ called *identity*, and $\mathsf{id}_A; a = a = a; \mathsf{id}_{A'}$ for any $a\colon A \to A'$.

Intuitively, one may think about $\mathsf{M}_\bullet$ as the class of all models conforming to a fixed but implicit metamodel. Deltas from M_Δ can be thought of as either structural (mappings) or behavioral (edit sequences) specifications of updates; details and a thorough discussion can be found in [4]. Identity deltas can be seen as *idle* updates that do nothing. We will use terms delta and update interchangeably.

We will also assume the following *update (delta) division* condition. Given a pair of updates with a common source, $a_1\colon A \to A_1, a_2\colon A \to A_2$, let a_2/a_1 denotes the set $\{a\colon A_1 \to A_2\colon\ a_1; a = a_2\}$ of updates that continue update a_1 and result in update a_2. We require this set to be non-empty for any pair (a_1, a_2):

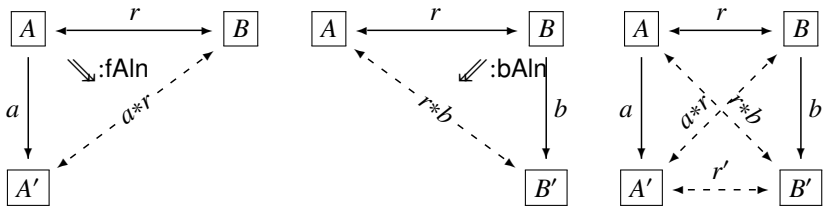
(UDiv). $\qquad a_2/a_1 \neq \varnothing$

Definition 2 (Alignment frameworks). An *alignment framework* is given by the following data.

(a) A pair of model spaces (M, N) called the *source* and the *target* space resp.

(b) A set R of *correspondence mappings* or just *corrs* between M- and N-models. That is, there are total functions $\mathsf{s}\colon \mathsf{M}_\bullet \leftarrow \mathsf{R}$ and $\mathsf{t}\colon \mathsf{R} \to \mathsf{N}_\bullet$, and we write $r\colon A \leftrightarrow B$ or $r \in \mathsf{R}(A, B)$ if $A = \mathsf{s}.r$ and $r.\mathsf{t} = B$. Note that we write the function symbol to the right or to the left of the argument to match the direction of function in our diagrams, in which space M will be on the left, and space N is on the right.

In our applications, a corr $r\colon A \leftrightarrow B$ is to be thought of as a set of bidirectional links from elements of model A to elements of model B so that if $e_1 \in A$ and $e_2 \in B$ are linked (then we could write $r(e_1, e_2)$), we consider them as two different representations of the same object. Moreover, one of the elements e_i, or even both, can be derived in its model by posing suitable queries against the model (see [8, 9] for details). However, in the present paper, elements of R are just abstract entities called *corrs* and denoted by bidirectional arrows between models in different spaces. Similarly, updates are abstract entities denoted by arrows between models in the same space. In our



(a) arities (b) (Assoc) law

Figure 1: Alignment operations and their laws

diagrams, update arrows will be always vertical whereas corrs are horizontal.

(c) A pair of operations over corrs and updates, fAln and bAln, which are called *forward* and *backward (re)alignment*. Their arities are shown in Fig. 1(a): operations' input nodes are framed, input arrows have solid bodies, and output arrows are dashed. We write $a * r$ for $\mathsf{fAln}(a, r)$ and $r * b$ for $\mathsf{bAln}(b, r)$.

We denote an alignment framework by a stroked arrow $\mathsf{R}\colon \mathsf{M} \nleftrightarrow \mathsf{N}$. Although the arrow is bidirectional to recall the symmetry of the notion, we still distinguish spaces M and N by their order in the line: the left space is the *source* space of the framework, and the right one is the *target* space.

Definition 3 (Well-behavedness). An alignment framework is called *well-behaved (wb)* if the following four laws are respected.

(a) Identity updates do not actually need realignment:

(IdAln) $\qquad \mathsf{id}A * r = r = r * \mathsf{id}B$

for any corr $r\colon A \leftrightarrow B$.

(b) The result of applying a sequence of interleaving forward and backward alignments does not depend on the order of application as shown in Fig. 1(b):

(Assoc) $\qquad (a * r) * b = a * (r * b)$

for any corr r and any updates a, b.

We will call diagrams like those shown in Fig. 1(a,b) *commutative* if the arrow at the respective operation output is indeed equal to that one computed by the operation. For example, diagram Fig. 1(b) is commutative if $r' = a * r * b$.

(c) Alignment is compositional: for any consecutive updates $a\colon A \to A'$, $a'\colon A' \to A''$, $b\colon B \to B'$, $b'\colon B' \to B''$, the following holds:

(AlnAln) $\quad (a; a') * r = a' * (a * r)$ and $r * (b; b') = (r * b) * b'$

(d) Corr division condition holds. Given a pair of corrs with a common source, $r_1 : A \leftrightarrow B_1$, $r_2 : A \leftrightarrow B_2$, let r_2/r_1 denotes the set $\{b\colon B_1 \to B_2 :\ r_1 * b = r_2\}$ of updates that map the first corr into the second via realignment. Similarly, $r_2 \setminus r_1$ is $\{a\colon A_1 \to A_2 :\ a * r_1 = r_2\}$ for a given pair of corrs $r_i\colon A_i \leftrightarrow B$. We require these sets to be non-empty:

(CDiv) $\qquad r_2/r_1 \neq \varnothing \neq r_2 \setminus r_1$

We will assume all our alignment frameworks are well-behaved by default. □

***Remark* 1 (Categorical treatment).** It is easy to check that conditions (a,b,c) of well-behavedness make an alignment framework a functor $\mathsf{R}\colon \mathsf{M} \times \mathsf{N} \to \textbf{\textit{Set}}$ or, equivalently, $\mathsf{R}\colon (\mathsf{M}^\circ)^\circ \times \mathsf{N} \to \textbf{\textit{Set}}$ (where $\_^\circ$ denotes dualization of a category, i.e., inversion of all its arrows). Such functors are called *profunctors* or *distributors* [**?**][Benabou], and usually denoted by stroked arrows $\mathsf{R} : \mathsf{M} \nrightarrow \mathsf{N}^\circ$. However, in our context it is more natural and convenient to work in a more symmetric setting of functor $\mathsf{R}\colon \mathsf{M} \times \mathsf{N} \to \textbf{\textit{Set}}$.

The following three conditions are equivalent:

(i) $\mathsf{R}\colon \mathsf{M} \nleftrightarrow \mathsf{N}$ is an alignment framework,

(ii) functor $\mathsf{R}\colon \mathsf{M}\times\mathsf{N} \to \textbf{\textit{Set}}$ satisfies (CDiv),

(iii) category Col(R) (with Col the *collage functor* $\mathsf{Col}\colon \textbf{\textit{Dist}} \to \textbf{\textit{Cat}}/\mathbf{2}$) satisfies the arrow division condition for any pair of its arrows with a common source, i.e., conditions (UDiv) and (CDiv).

Definition 4 (Consistency and private updates). A *consistency framework* is an alignment framework endowed with two extra structures modeling intermodel consistency. Formally, it is a triple of the following components.

(a) An alignment framework $\mathsf{R}\colon \mathsf{M} \leftrightarrows \mathsf{N}$ as defined above.

(b) A subset $\mathsf{K} \subset \mathsf{R}$ of corrs called *consistent*. In other words, for any pair of models $A \in \mathsf{M}_\bullet$ and $B \in \mathsf{N}_\bullet$ and the respective set of corrs $\mathsf{R}(A,B)$, there is a defined a subset of *consistent* corrs $\mathsf{K}(A,B) \subset \mathsf{R}(A,B)$ between A and B (i.e., $\mathsf{K}(A,B) = \mathsf{R}(A,B) \cap \mathsf{K}$). If a corr $r \in \mathsf{K}(A,B)$, we say that models A and B are *consistent via* r and write $A \sim_r B$.

We do not exclude the cases when $\mathsf{K}(A,B)$ has more than one corr, or is empty. In the latter case we write $A\#B$ and call the models *(entirely) inconsistent*. If, on the contrary, $\mathsf{K}(A,B) \neq \varnothing$, we say that the models are *potentially consistent* and write $A \sim B$. For $A \in \mathsf{M}_\bullet$, we write $A.\mathsf{K}_\bullet$ for the set $\{B \in \mathsf{N}_\bullet \colon A \sim B\}$. Similarly, ${}_\bullet\mathsf{K}.B$ denotes the set $\{A \in \mathsf{M}_\bullet \colon A \sim B\}$ (where we again write a function symbol to the right or to the left of the argument to suggest the direction of function in our diagrams).

We assume the following *totality* condition:

(Total) $\quad A.\mathsf{K}_\bullet \neq \varnothing \neq {}_\bullet\mathsf{K}.B$ for all $A \in \mathsf{M}_\bullet$, $B \in \mathsf{N}_\bullet$

That is, any model has at least one potentially consistent counterpart on the other side, and hence synchronization is always possible.

(c) Subspaces of *private updates* are given, $\mathsf{M}^{\mathrm{prv}} \subset \mathsf{M}$ and $\mathsf{N}^{\mathrm{prv}} \subset \mathsf{N}$, such that $\mathsf{M}^{\mathrm{prv}}_\bullet = \mathsf{M}_\bullet$ and $\mathsf{N}^{\mathrm{prv}}_\bullet = \mathsf{N}_\bullet$. That is, for any two models $A, A' \in \mathsf{M}_\bullet$, there is a set (perhaps, empty) of updates $\mathsf{M}^{\mathrm{prv}}_\Delta(A,A') \subseteq \mathsf{M}_\Delta(A,A')$ called *private*, composition of two private updates is private, idle updates are private, and divisions of private updates are private two: $a_2/a_1 \subset \mathsf{M}^{\mathrm{prv}}_\Delta$ as soon as $a_1, a_2 \in \mathsf{M}^{\mathrm{prv}}_\Delta$. Likewise for the N side.

Non-private updates are called *public*. Thus, we have partitions: $\mathsf{M}_\Delta = \mathsf{M}^{\mathrm{prv}}_\Delta \uplus \mathsf{M}^{\mathrm{pub}}_\Delta$ and $\mathsf{N}_\Delta = \mathsf{N}^{\mathrm{prv}}_\Delta \uplus \mathsf{N}^{\mathrm{pub}}_\Delta$.

Definition 5 (Well-behavedness). Informally, a consistency framework is *well-behaved*, if (the underlying alignment framework is wb and) the two facets of consistency, consistent corrs and private updates, fit together so that private updates do not affect consistency. Formally, we require the following two laws to hold: (a) For any corr $r\colon A \leftrightarrow B$ and any private updates $a\colon A \to A'$, $b\colon B \to B'$, we have $a * r \in \mathsf{K}$ iff $r \in \mathsf{K}$ iff $r * b \in \mathsf{K}$. (b) For any two consistent corrs with the same source, $r_1 \in \mathsf{K}(A, B_1)$ and $r_2 \in \mathsf{K}(A, B_2)$, we have $r_2/r_1 \subset \mathsf{N}^{\mathrm{prv}}_\Delta$. That is, two consistent N-counterparts of an M-model A only differ privately (but share the same public part determined by A). Similarly, $r_2 \backslash r_1 \subset \mathsf{M}^{\mathrm{prv}}_\Delta$ for any two corrs with the same target, $r_1 \in \mathsf{K}(A_1, B)$ and $r_2 \in \mathsf{K}(A_2, B)$. □

To unify notation, we will sometimes use symbol K polymorphically for denoting both consistent corrs and private updates. That is, we write $\mathsf{K}(X,Y)$ with $X, Y \in \mathsf{M}_\bullet \cup \mathsf{N}_\bullet$ meaning that

$\mathsf{K}(X,Y) \stackrel{\mathrm{def}}{=} \mathsf{M}^{\mathrm{prv}}_\Delta(X,Y)$ if both $X, Y \in \mathsf{M}_\bullet$,
$\mathsf{K}(X,Y) \stackrel{\mathrm{def}}{=} \mathsf{N}^{\mathrm{prv}}_\Delta(X,Y)$ if both $X, Y \in \mathsf{N}_\bullet$,
$\mathsf{K}(X,Y) \stackrel{\mathrm{def}}{=} \mathsf{K}(X,Y)$ if $X \in \mathsf{M}_\bullet$ and $Y \in \mathsf{N}_\bullet$,
$\mathsf{K}(X,Y) = \perp$ (not defined) if $X \in \mathsf{N}_\bullet$, $Y \in \mathsf{M}_\bullet$.[3]

Then we can say that a consistency framework is a pair (R, K) with R an alignment framework and K a polymorphic function as above, which satisfy all necessary conditions. We will denote a consistency framework as $\mathsf{K}\colon \mathsf{M} \leftrightarrows \mathsf{N}$ without explicit mentioning the underlying alignment framework $\mathsf{R}\colon \mathsf{M} \leftrightarrows \mathsf{N}$.

***Remark* 2 (Categorical treatment).** The polymorphic use of K described above is a known categorical construction of presenting a distributor by its *collage* category, actually a barrel. It can be proven that functor $\mathsf{Col}\colon \boldsymbol{Dist} \to \boldsymbol{Cat}/\boldsymbol{2}$ is an equivalence of categories (see Joyal's Catlab; http://nlab.mathforge.org/joyalscatlab/)

If to follow Leibniz's (formal) vs. Newtonian (physical) style of building mathematical models, then the definitions should ensure that K is a sub-profunctor with division of R. Correspondingly, category $\mathsf{Col}(\mathsf{K})$ is a subcategory of $\mathsf{Col}(\mathsf{R})$, which inherits the division property.

2.2. Informational asymmetry

Definition 6 (Info-asymmetry). Let $\mathsf{K}\colon \mathsf{M} \leftrightarrows \mathsf{N}$ be a wb consistency framework. We say that space M is *dominated by* N *via* K, and write $\mathsf{M} \leq_{\mathsf{K}} \mathsf{N}$ (or $\mathsf{N} \geq_{\mathsf{K}} \mathsf{M}$), if the only private updates on the M-side are identities.

Proposition 1. Let $\mathsf{M} \leq_{\mathsf{K}} \mathsf{N}$. Then for any $B \in \mathsf{N}_\bullet$, there is a uniquely defined model $A \in \mathsf{M}_\bullet$ called the *(abstract) view of* B, such that (a) $\mathsf{K}(A,B)$ is a singleton whereas (b) $\mathsf{K}(A',B) = \varnothing$ for any $A' \neq A$.

Proof. Let $r{\in}\mathsf{K}(A,B)$ and $r'{\in}\mathsf{K}(A',B)$. Then there is a private update $a_{rr'}: A \to A'$ such that $a_{rr'} * r = r'$ by the definition of consistency framework. However, $\mathsf{M} \leq_{\mathsf{K}} \mathsf{N}$ implies that a is identity, i.e., $A = A'$, and $r = r'$ by the IdAln law. □

Corollary 1. If $\mathsf{M} \leq_{\mathsf{K}} \mathsf{N}$, then function $f^{\mathsf{K}}: \mathsf{M}_\bullet \leftarrow \mathsf{N}_\bullet$ is defined such that (a) set $\mathsf{K}(f^{\mathsf{K}}.B, B)$ is a singleton for any model $B{\in}\mathsf{N}_\bullet$, while (b) $\mathsf{K}(A', B) = \varnothing$ for any $A' \neq f^{\mathsf{K}}.B$.

Model $f^{\mathsf{K}}.B$ is called the *view* of model B determined by consistency framework K, and f^{K} is the view computation function; following the terminological tradition of lenses, we call this function *get-the-view* and denote it by $\mathsf{get}^{\mathsf{K}}_\bullet$. Condition (b) formalizes the requirement that any non-idle view update makes the view inconsistent with its source. Condition (a) states that the view is given with its unique traceability mapping — the unique element of set $\mathsf{K}(\mathsf{get}^{\mathsf{K}}_\bullet.B, B)$, which we will denote by $\mathsf{get}^{\mathsf{K}}_{\leftrightarrow}.B$. As a rule, we will omit superindex K if it is clear from the context. Note also that function $\mathsf{get}_\bullet$ is surjective due to Totality condition in Definition 4.

Proposition 2. If $\mathsf{M}{\leq_{\mathsf{K}}}\mathsf{N}$ and $\mathsf{N}{\leq_{\mathsf{K}}}\mathsf{M}$, then the respective view computation functions $\mathsf{get}_{\bullet 1}: \mathsf{M}_\bullet \leftarrow \mathsf{N}_\bullet$ and $\mathsf{get}_{\bullet 2}: \mathsf{M}_\bullet \to \mathsf{N}_\bullet$ are mutually inverse: $(\mathsf{get}_{\bullet 1}.B).\mathsf{get}_{\bullet 2} = B$ for any $B{\in}\mathsf{N}_\bullet$, and $A = \mathsf{get}_{\bullet 1}.(A.\mathsf{get}_{\bullet 2})$ for any $A{\in}\mathsf{M}_\bullet$. Hence, both $\mathsf{get}_\bullet$-functions are bijections. Conversely, if $\mathsf{M} \leq_{\mathsf{K}} \mathsf{N}$ and the respective $\mathsf{get}_{\bullet 1}: \mathsf{M}_\bullet \leftarrow \mathsf{N}_\bullet$ is bijective, then $\mathsf{N} \leq_{\mathsf{K}} \mathsf{M}$ and the respective $\mathsf{get}_{\bullet 2}: \mathsf{M}_\bullet \to \mathsf{N}_\bullet$ is the inverse of $\mathsf{get}_{\bullet 1}$. □

2.3. Info-symmetry types

Our goal is to specify a set of properties of consistency frameworks, which would set their (complete and disjoint) taxonomy. We call these properties *info-symmetry types*. We will begin with setting some general terminology about types, then consider dualization, and finish this subsection with a formal definition of the taxonomy.

Let ***CFwk*** be the class of all consistency frameworks. If π is a property of consistency frameworks, then we write $\mathsf{K} \models \pi$ to say that a framework K satisfies the property, and we thus have a set $[\![\,\pi\,]\!] = \{\mathsf{K} \in \boldsymbol{CFwk}\colon\ \mathsf{K} \models \pi\}$ of all frameworks satisfying the property. If $\mathsf{K} \in [\![\,\pi\,]\!]$, i.e., $\mathsf{K} \models \pi$, we will say that framework K is an *instance of type* π, or K is *of type* π. Thus, we loosely use the term *type* both syntactically — as a synonym for the term *property*, and semantically — to refer to the corresponding set of instances.

Having a set of types/properties $\{\pi_1 \ldots \pi_n\}$, we can form a type $\pi = \pi_1 \vee \ldots \vee \pi_n$ so that $[\![\,\pi\,]\!] = [\![\,\pi_1\,]\!] \cup \ldots \cup [\![\,\pi_n\,]\!]$. Then we call type π *abstract* because instantiating this type means instantiation of one of the subtypes $[\![\,\pi_i\,]\!]$: $\mathsf{K} \in [\![\,\pi\,]\!]$ iff $\mathsf{K} \in [\![\,\pi_i\,]\!]$ for some i. We call a set of types $\{\pi_1 \ldots \pi_n\}$ *(taxonomically) complete* if $\boldsymbol{CFwk} = [\![\,\pi_1\,]\!] \cup \ldots \cup [\![\,\pi_n\,]\!]$. We call types *disjoint* if $\mathsf{K} \models \pi_i$ implies $\mathsf{K} \not\models \pi_j$ for all $j \neq i$; then sets $[\![\,\pi_i\,]\!]$ are disjoint.

Two types are logically equivalent, $\pi_1 \Leftrightarrow \pi_2$, iff they have the same extension $[\![\,\pi_1\,]\!] = [\![\,\pi_2\,]\!]$. □

For considering informational and organizational symmetries, the following notion will be central.

Definition 7 (Dualization). Any consistency framework $\mathsf{K}\colon \mathsf{M} \nleftrightarrow \mathsf{N}$ determines its dual framework $\mathsf{K}^\circ\colon \mathsf{N} \nleftrightarrow \mathsf{M}$ in the following way. We first dualize the underlying alignment framework: (i) the source space is N and the target space is M; (ii) for any corr r, $r.\mathsf{s}^\circ \stackrel{\text{def}}{=} r.\mathsf{t}$ and $\mathsf{t}^\circ.r \stackrel{\text{def}}{=} \mathsf{s}.r$; (iii) operations are $\mathsf{fAln}_{\mathsf{R}^\circ}(b, r) \stackrel{\text{def}}{=} \mathsf{bAln}_{\mathsf{R}}(b, r)$ and $\mathsf{bAln}_{\mathsf{R}^\circ}(a, r) \stackrel{\text{def}}{=} \mathsf{fAln}_{\mathsf{R}}(a, r)$. Then we dualize the consistency structure: $\mathsf{K}^\circ(B, A) \stackrel{\text{def}}{=} \mathsf{K}(A, B)$, $\mathsf{K}^\circ(B, B') = \mathsf{K}(B, B') = \mathsf{N}^{\text{prv}}_\Delta(B, B')$ and $\mathsf{K}^\circ(A, A') \stackrel{\text{def}}{=} \mathsf{K}(A, A') = \mathsf{M}^{\text{prv}}_\Delta(A, A')$.

Definition 8 (Symmetric and asymmetric types). Any type/property π gives rise to a *dual type* π° such that $\mathsf{K} \models \pi^\circ$ iff $\mathsf{K}^\circ \models \pi$. We denote type $\leq^\circ_{\mathsf{K}}$ (see Definition 6) by $\geq_{\mathsf{K}}$, and write $\mathsf{N} \geq_{\mathsf{K}} \mathsf{M}$ for $(\mathsf{K}\colon \mathsf{M} \nleftrightarrow \mathsf{N}) \models \leq^\circ$. A type π is called *almost concrete* if $\pi \Leftrightarrow \pi_1 \vee \pi_1^\circ$ (and hence $[\![\,\pi\,]\!] = [\![\,\pi_1\,]\!] \cup \left[\!\!\left[\,\pi_1^\circ\,\right]\!\!\right]$) for some concrete type π_1. A type is called *symmetric* if $\pi \Leftrightarrow \pi^\circ$. We call types π_1, π_2 *mutually dual* if $\pi_1 \Leftrightarrow \pi_2^\circ$ (and hence $\pi_1^\circ \Leftrightarrow \pi_2$).

Proposition 3. For any $\mathsf{K}\colon \mathsf{M} \nleftrightarrow \mathsf{N}$, $(\mathsf{K}^\circ)^\circ = \mathsf{K}$. For any type π, $(\pi^\circ)^\circ \Leftrightarrow \pi$.

Proposition 4 (Dualization and consistency). A consistency framework $\mathsf{K}\colon \mathsf{M} \nleftrightarrow \mathsf{N}$ is well-behaved(wb) iff its dual $\mathsf{K}^\circ\colon \mathsf{N} \nleftrightarrow \mathsf{M}$ is wb too. Thus, the notion of consistency is symmetric and not affected by dualization.

Definition 9 (Info-symmetries). Let $\mathsf{K}\colon \mathsf{M} \nleftrightarrow \mathsf{N}$ be a consistency framework.

(a1) If neither side has non-idle private updates, we call K *poorly info-symmetric* and write $\mathsf{M}{\approx_{\mathsf{K}}}\mathsf{N}$ or $\mathsf{K}_\approx\colon \mathsf{M} \nleftrightarrow \mathsf{N}$.

(a2) If both sides have non-idle private updates, we call K *richly info-symmetric*, and write $\mathsf{M}\!\!\succ\!\!\prec_{\mathsf{K}}\!\!\mathsf{N}$ or $\mathsf{K}_{\succ\prec}\colon \mathsf{M} \leftrightarrows \mathsf{N}$

We say a consistency framework K is *info-symmetric* if it is either poorly or richly symmetric.

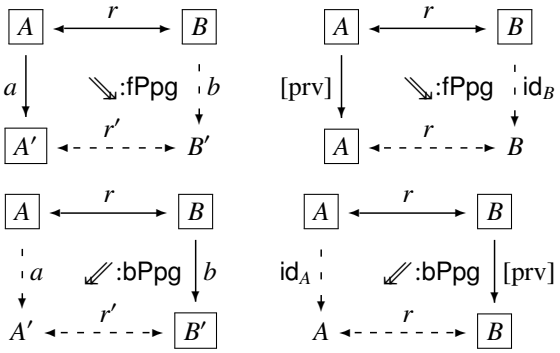
(b) If only one side has non-idle private updates, we we call the consistency framework *info-asymmetric*. If this side is the target space, we write $\mathsf{M} <_{\mathsf{K}} \mathsf{N}$ or $\mathsf{K}_{<}\colon \mathsf{M} \leftrightarrows \mathsf{N}$. If it is the source space, we write $\mathsf{M} >_{\mathsf{K}} \mathsf{N}$ or $\mathsf{K}_{>}\colon \mathsf{M} \leftrightarrows \mathsf{N}$.

Theorem 1 (Info-symmetry taxonomy). Let $\mathsf{K}\colon \mathsf{M} \leftrightarrows \mathsf{N}$ be a consistency framework. There are four mutually exclusive and jointly complete logical possibilities (concrete info-symmetry types): (a1) $\mathsf{M}\approx_{\mathsf{K}}\mathsf{N}$, (a2) $\mathsf{M}\!\!\succ\!\!\prec_{\mathsf{K}}\!\!\mathsf{N}$, (b1) $\mathsf{M} <_{\mathsf{K}} \mathsf{N}$, and (b2) $\mathsf{M} >_{\mathsf{K}} \mathsf{N}$, which can be grouped in two abstract types: info-symmetry (a) = (a1)∨(a2), and info-asymmetry (b)=(b1)∨(b2). The latter type is almost concrete as types (b1) and (b2) are mutually dual. □

Two important remarks are in order.

***Remark* 3 (Implicit metamodel mappings).** Each of the four relationships above is a property of a triple $(\mathsf{M}, \mathsf{K}, \mathsf{N})$ rather than a pair (M, N). When we talk about the info-symmetry relation between UML models and Java code, or between class diagrams and relational schemas, we implicitly assume some mapping between the metamodels is given, and this mapping determines a corresponding consistency framework between the model spaces (see [3, 10] for details). In this sense we can use a loose notation and write $\mathsf{M}\!\!\succ\!\!\prec_{\mathsf{inf}}\!\!\mathsf{N}$ meaning that we have a consistency framework K such that $\mathsf{M}\!\!\succ\!\!\prec_{\mathsf{K}}\!\!\mathsf{N}$.

***Remark* 4 (Spaces vs. models).** Info-symmetry is a relationship between model spaces rather than individual model states. Suppose, for example, that we have a richly symmetric situation $\mathsf{M}\!\!\succ\!\!\prec_{\mathsf{K}}\!\!\mathsf{N}$, and two synchronized models A and B evolving in their respective spaces (we will consider this in detail later in 4). A specific state A_0 of model A can lack private updates because A_0 is a poor model that does not have private data, and hence any update $a\colon A_0 \to A_1$ is public. But state A_1 (or consecutive states) can well have private data and hence private updates. When we have been writing $A\!\!\succ\!\!\prec_{\mathsf{inf}}\!\!B$ in the paper, we actually meant the respective relationship between model spaces defined by the metamodels of A and B (w.r.t. some implicit mapping between the metamodels as explained in Remark 3 above).



(a) arities (b) (IdPpg) laws

Figure 2: Operations of update propagation

3. Formalizing Incrementality: Delta Lenses

We will describe a family of algebraic structures called *(delta) lenses*, which model non-incremental and incremental model synchronization. Each such a structure will be defined over some underlying consistency framework $\mathsf{K}\colon \mathsf{M} \leftrightarrows \mathsf{N}$, and comprises two operations of forward (from M to N) and backward (from N to M) update propagation.

3.1. Incremental update propagation and lenses

Definition 10 (Lenses). Let $\mathsf{K}\colon \mathsf{M} \leftrightarrows \mathsf{N}$ be a consistency framework. An a *(delta) lens*) over K is a pair of operations over corrs and updates, fPpg and bPpg, which are called, resp., *forward* and *backward update propagation*. The arities of the operations are specified in Fig. 2(a) with output arrows dashed and output nodes not framed.

We will often use a linear notation and write $a.\mathsf{fPpg}(r) = b$ (read "update a is propagated over corr r to b") and $a = \mathsf{bPpg}(r).b$ ("a is obtained by backward propagation of b over r") for the cases specified in the diagrams. Similarly, we will write $r' = r.\mathsf{fPpg}(a)$ (read "r' is the result of re-alignment caused by update a") and $r' = \mathsf{bPpg}(b).r$ ("r' is re-alignment caused by b").

We denote a delta lens by a double arrow $\lambda\colon \mathsf{M} \leftleftharpoons \mathsf{N}$ to recall two operations.

Definition 11 (Dual lenses). Given a consistency framework $\mathsf{K}\colon \mathsf{M} \leftrightarrows \mathsf{N}$ and a lens $\lambda\colon \mathsf{M} \leftleftharpoons \mathsf{N}$ over it, the dual lens, $\lambda^{\circ}\colon \mathsf{N} \leftleftharpoons \mathsf{M}$ consist of the operations $(\mathsf{bPpg}, \mathsf{fPpg})$ over the dual consistency framework $\mathsf{K}^{\circ}\colon \mathsf{N} \leftrightarrows \mathsf{M}$.

Definition 12 (Well-behaved lenses). A lens $\lambda\colon \mathsf{M} \leftrightharpoons \mathsf{N}$ is called *well-behaved (wb)* if it satisfies several laws specified below.

Correctness. If $(r\colon A \leftrightarrow B) \in \mathsf{K}(A,B)$ and $(b,r') = \mathsf{fPpg}(a,r)$ for some update $a\colon A \to A'$, then $r' \in \mathsf{K}(A',B')$ where $b\colon B \to B'$. Similarly for operation bPpg. That is, update propagation ensures consistency restoration. □

Privacy no-op. For any update $a\colon A \to A'$ the following two conditions are equivalent: (i) a is private, (ii) $a.\mathsf{fPpg}(r) = \mathsf{id}_B$ for any consistent corrs $r\colon A \leftrightarrow B$. Similar equivalences are required for any update $b\colon B \to B'$ (the lower diagram). That is, private updates defined by the underlying consistency framework are exactly updates mapped to identity, i.e., in fact, not propagated to the other side. □

Recall that private updates determined by the underlying consistency framework are updates that do not affect consistency, and hence, need not be propagated. Stevens [11] called this property Hypocraticness. Hence, Privacy no-op implies Hippocraticness, but also requires the converse implication.

As idle updates are private, they are propagated to idle updates and we say that the lens is *stable*: if nothing changes on one side, nothing happens on the other side as well [5] (a very special case of Hippocraticness).

Proposition 5 (Conjecture). Is condition (ii) in Privact no-op equivalent to the following weak version: $a.\mathsf{fPpg}(r) = \mathsf{id}_B$ for some consistent corr $r\colon A \leftrightarrow B$ (see the upper diagram in Fig. 2(b))

Compatibility with alignment. If $a.\mathsf{fPpg}(r) = \mathsf{id}_B$ for $a\colon A \to A'$ and $r\colon A \leftrightarrow B$, then $a * r = r.\mathsf{fPpg}(a)$. Dually for bPpg. That is, a wb delta lens can re-align a corr if the change was caused by a *private* update. Re-alignment along a public update is, in general, not possible as the lens must first propagate, and only then re-align the update. □

Compositionality. Consider two consecutive updates $a_1\colon A \to A_1$, $a_2\colon A_1 \to A_2$ and their composition $a_{12} = a_1;a_2\colon A \to A_2$. In reasonable synchronization scenarios, if both updates a_i are insertions, or if both are deletions, then $a_{12}.\mathsf{fPpg}(r) = a_1.\mathsf{fPpg}(r);a_2.\mathsf{fPpg}(r_1)$ for any consistent corr $r\colon A \leftrightarrow B$ and $r_1 = r.\mathsf{fPpg}(a_1)$ (Fig. 3 shows how arrows fit together, but the marker :fPpg to be labeling the outer rectangle ABB_2A_2 is not shown).

In general, for a corr $r\colon A \leftrightarrow B$, the equality $a_{12}.\mathsf{fPpg}(r) = a_1.\mathsf{fPpg}(r);a_2.\mathsf{fPpg}(r_1)$ with $r_1 = r.\mathsf{fPpg}(a_1)$ does not hold if update a_1 involves deletion of some public data (say, A^-) of model A whereas update a_2 restores these data so that the composed update a does not actually change A. However, the story on the B-side is more complex.

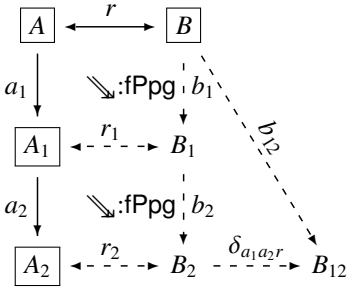


Figure 3: Compositionality for fPpg

Update $b_1 = a_1.\mathsf{fPpg}(r)$ must delete the r-image of data piece A^- in model B, , say, B^-, and the respective part of private data based on B^-, say, $B^{-\mathrm{prv}}$, as well. Thus, we have $b_1\colon B \to B_1$ with B^- and $B^{-\mathrm{prv}}$ deleted from B_1. Then update $b_2 = a_2.\mathsf{fPpg}(r_1)\colon B_1 \to B_2$ must restore data B^- in model B_2, but as the respective private component $B^{-\mathrm{prv}}$ is lost, b_2 replaces it with some standard minimal set of N-private data, say, B_0. Thus, model B_2 is built from B^- and B_0. In contrast, update $b_{12} = a_{12}.\mathsf{fPpg}(r)\colon B \to B_{12}$ keeps data B^- unchanged in B_{12}, and hence the latter is built from B^- and $B^{-\mathrm{prv}}$. Minimality of B_0 is specified by a uniquely determined delta $\delta_0^-\colon B_0 \to B^-$, and then we should have a delta $\delta\colon B_2 \to B_{12}$ obtained by pairing id_{B^-} and δ_0^- such that $b_1;b_2;\delta = a_{12}.\mathsf{fPpg}(r)$. This delta is private (as its public component is idle). These considerations motivate the following formal law: For any two consecutive updates, $a_1 : A \to A_1$, $a_2 : A_1 \to A_2$ and a corr $r\colon A \leftrightarrow B$, there is a *private* delta $\delta_{a_1a_2r}\colon B_2 \to B_{12}$ such that $b_1;b_2;\delta_{a_1a_2r} = (a_1;a_2).\mathsf{fPpg}(r)$ as shown in Fig. 3

If both updates are insertions, or both are deletions, then $\delta_{a_1a_2r}$ is required to be an identity, and, hence, $a_1.\mathsf{fPpg}(r);a_2.\mathsf{fPpg}(r_1) = (a_1;a_2).\mathsf{fPpg}(r)$. Moreover, if at least one of the updates is private, then $\delta_{a_1a_2r}$ is required to be an identity too.

A similar law is stated for a corr and two consecutive updates on the N-side. □

Invertibility. We begin with an informal discussion. Given an update $a\colon A \to A'$ and a consis-

tent corr $r: A \leftrightarrow B$, let $a_{rr}: A \to A'_{rr}$ denotes the update $a.\mathsf{fPpg}(r).\mathsf{bPpg}(r)$ resulting from forward and then backward propagation (see the left and central squares in diagram Fig. 4a). In general, $a_{rr} \neq a$ because when a is propagated to the N-side, the private part of a ia lost and cannot be restored. In more detail, we can consider update $a: A \to A'$ as a pair $(a^{\text{prv}}, a^{\text{pub}})$, whose components update, resp. the private, say, A^{prv}, and the public, A^{pub}, parts of A. Then update a_{rr} is also a pair $(\mathsf{id}_{A^{\text{prv}}}, a^{\text{pub}})$, and there should be a delta $\delta_{ar} = (a^{\text{prv}}, id_{A^{\text{pub}}}): A'_{rr} \to A'$ so that $a_{rr}; \delta_{ar} = a$. These considerations motivate the following formal law: for any $a: A \to A'$ and consistent corr $r: A \leftrightarrow B$, there is defined a *private* delta $\delta_{ar}: A'_{rr} \to A'$ such that $a_{rr}; \delta_{ar} = a$ with symbols A'_{rr} and a_{rr} explained in Fig. 4(a). Dually, for any $b: B \to B'$ and consistent corr $r: A \leftrightarrow B$, there is defined a private update δ_{br} such that $b; \delta_{br} = b_{rr}$ as shown in diagram Fig. 4(b). We call δ_{ar} and δ_{br} *invertibility* deltas.

Corollary 2. An immediate consequence of the existence of invertibility deltas are the following laws: $a.\mathsf{fPpg}(r).\mathsf{bPpg}(r).\mathsf{fPpg}(r) = a.\mathsf{fPpg}(r)$ and $b.\mathsf{bPpg}(r).\mathsf{fPpg}(r).\mathsf{bPpg}(r) = b.\mathsf{bPpg}(r)$ described by Fig. 4(a')(b') and called *weak invertibility* in [5], Thus, invertibility deltas imply weak invertibility. □

We present two results about lenses.

Theorem 2. For a wb-lens, if update $a: A \to A'$ is public, $a \in \mathsf{M}^{\text{pub}}_{\Delta}$, then for any consistent corr $r: A \leftrightarrow B$, update $a.\mathsf{fPpg}(r)$ is also public and belongs to $\mathsf{N}^{\text{pub}}_{\Delta}$. Dually, if $b: B \to B'$ is public, then $\mathsf{bPpg}(r).b$ is also public.

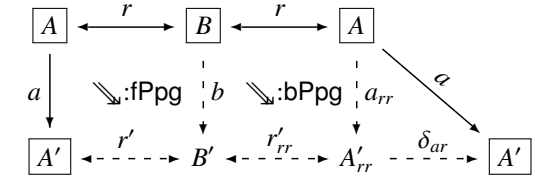
Proof. Follows from the definition of privacy/publicity and the privacy of invertibility deltas. □

Theorem 3. A lens $\lambda: \mathsf{M} \leftrightharpoons \mathsf{N}$ is wb iff its dual $\lambda^{\circ}: \mathsf{M} \leftrightharpoons \mathsf{N}$ is wb too. That is, well-behavedness is a symmetric notion. □

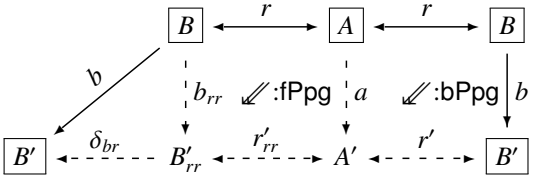
3.2. Info-Asymmetry and Lenses

The consistency framework underlying a lens influences its properties.

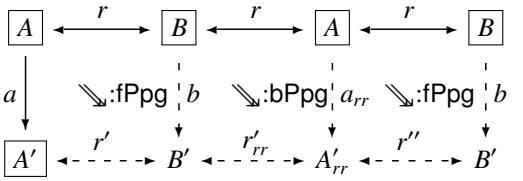
Definition 13 (Asymmetric lenses). [ZD: insert the def from the jot11 paper [4].] We denote an asymmetric lens by $\lambda^{\leq}: \mathsf{M} \leftrightharpoons \mathsf{N}$ with the superscript pointing to the dominated side (the view).



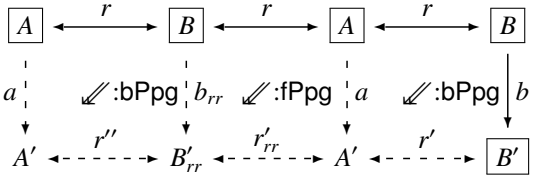
(a) fbPpg invertibility delta



(b) bfPpg invertibility delta

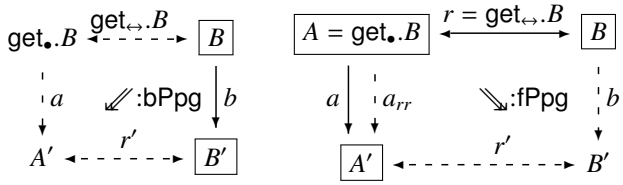


(a') fbfPpg law



(b') bfbPpg law

Figure 4: Round-tripping laws. (a) and (b) are Invertibility deltas. (a') and (b') are weak invertibility. Scenario in diagrams (b,b') "run" from the right to the left.



(a) defining get_Δ (b) defining put

Figure 5: Extracting an asymmetric lens for info-asymmetric triple space

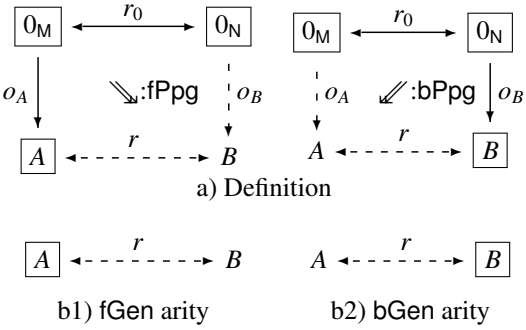


Figure 6: Non-incremental propagation

Theorem 4 (Info-asymmetry and lenses). Any delta lens $\lambda\colon \mathsf{M} \leftrightharpoons \mathsf{N}$ over an asymmetric alignment framework $\mathsf{K}^{\leq}\colon \mathsf{M} \nleftrightarrow \mathsf{N}$ (i.e., $\mathsf{M} \leq_{\mathsf{inf}} \mathsf{N}$ holds), gives rise to an asymmetric delta lens as defined in [4].

Proof. We will employ functions $\mathsf{get}^{\mathsf{K}}_{\bullet}\colon \mathsf{M} \leftarrow \mathsf{N}$ and $\mathsf{get}^{\mathsf{K}}_{\leftrightarrow}\colon \mathsf{R} \leftarrow \mathsf{N}_{\bullet}$ provided by the asymmetry of $\mathsf{K}\colon \mathsf{M} \nleftrightarrow \mathsf{N}$, and show how they can be used to define functions get_Δ and put such that the PutGet law holds.

Diagram Fig. 5(a) shows how to define get_Δ for a given update $b\colon B \to B'$. We first compute $A = \mathsf{get}^{\mathsf{K}}_{\bullet}.B$ and $r = \mathsf{get}^{\mathsf{K}}_{\leftrightarrow}.B$, then apply bPpg and obtain arrows a and r' as shown in the diagram. Because ${}_{\bullet}\mathsf{K}.B$ is a singleton by the info-asymmetry condition, we necessarily have $A' = \mathsf{get}^{\mathsf{K}}_{\bullet}.B'$ and $r' = \mathsf{get}^{\mathsf{K}}_{\leftrightarrow}.B'$. We now define $\mathsf{get}_\Delta(b) = a$. It preserves identities as bPpg does.

Definition of put is shown by diagram Fig. 5(b). Recall that $\mathsf{get}^{\mathsf{K}}_{\bullet}$ is surjective; having an update a and model B such that $A = \mathsf{get}^{\mathsf{K}}_{\bullet}.B$ as shown in the diagram (b), we compute $r = \mathsf{get}^{\mathsf{K}}_{\leftrightarrow}(B)$ and apply fPpg, which produces arrows b and r' as shown. We define $\mathsf{put}(a, B) \stackrel{\text{def}}{=} b$. Because ${}_{\bullet}\mathsf{K}.B'$ is a singleton, we conclude that $A' = \mathsf{get}^{\mathsf{K}}_{\bullet}.B'$, and it remains to prove the PutGet law. Let $a_{rr} \stackrel{\text{def}}{=} \mathsf{get}_\Delta.b = \mathsf{bPpg}(r).(a.\mathsf{fPpg}(r))$ (see diagram Fig. 4(a)), and δ_{ar} is the respective invertibility delta. The latter is always private, and by asymmetry of the alignment framework, δ_{ar} must be an identity, hence, $a = a_{rr}$. □.

Theorem 5 (Info-bijectivity and lenses). Let $\lambda\colon \mathsf{M} \leftrightharpoons \mathsf{N}$ be a lens over a bijective consistency framework $\mathsf{K}^{\cong}\colon \mathsf{M} \nleftrightarrow \mathsf{N}$ Then categories M and N are isomorphic via mutually inverse functors $\mathsf{get}_1\colon \mathsf{M} \leftarrow \mathsf{N}$ and $\mathsf{get}_2\colon \mathsf{M} \to \mathsf{N}$.

3.3. Non-incremental update propagation.

We will begin with an important addition to the notion of model space.

Definition 14 (Initial models). Let M be a model space. A model 0_{M} is called *initial*, if for any model $A \in \mathsf{M}_{\bullet}$ there is a unique delta $o_A\colon 0_{\mathsf{M}} \to A$. It can be proven by standard categorical means that all initial models are isomorphic.

Intuitively, the initial model comprises all data that must be in any M-model and so is the minimal model possible; delta o_A embeds model 0_{M} into A. For many structural model spaces, e.g., class diagrams, the initial model is empty. In contrast, behavior models often must have certain states and transitions initializing the behavior, e.g., initial states.

Definition 15 (Well-behaved lenses and initial models). A consistency framework $\mathsf{K}\colon \mathsf{M} \nleftrightarrow \mathsf{N}$ is called *wb*, if in addition to laws stated in Definition 5, $\mathsf{K}(0_{\mathsf{M}}, 0_{\mathsf{N}})$ is a singleton $\{r_0\}$.

A lens $\lambda\colon \mathsf{M} \leftrightharpoons \mathsf{N}$ is *wb*, if, in addition to laws stated in Definition 12, the following holds: for any $A \in \mathsf{M}_{\bullet}$, $o_A.\mathsf{fPpg}(r_0) = o_B$ for some $B \in \mathsf{N}_{\bullet}$, and similarly, for any $B \in \mathsf{N}_{\bullet}$, $o_A = \mathsf{bPpg}(r_0).o_b$ for some $A \in \mathsf{M}_{\bullet}$. □

Now we note that a wb lens $\lambda\colon \mathsf{M} \leftrightharpoons \mathsf{N}$ provides also non-incremental propagation operations defined as specified in Fig. 6(a). What we have defined are two operations, *forward*, fGen, and *backward*, bGen, *generation*, whose arities are specified in Fig. 6(b): $A.\mathsf{fGen} = (r, B)$ and $(A, r) = \mathsf{bGen}.B$

Each of these operations actually comprises two ordinary operations: fGen consists of $\mathsf{fGen}_{\bullet}\colon \mathsf{M}_{\bullet} \to \mathsf{N}_{\bullet}$

and $\mathsf{fGen}_{\leftrightarrow}: \mathsf{M}_\bullet \to \mathsf{R}$; and bGen consists of $_\bullet\mathsf{bGen}: \mathsf{M}_\bullet \leftarrow \mathsf{N}_\bullet$ and $_{\leftrightarrow}\mathsf{bGen}: \mathsf{R} \leftarrow \mathsf{N}_\bullet$. For example, in diagram Fig. 6(b1), we have $A.\mathsf{fGen}_{\leftrightarrow} = r$ and $A.\mathsf{fGen}_\bullet = B$.

The following result shows how non- and incremental update propagation are related (see [12] for details).

Theorem 6. Let $\lambda: \mathsf{M} \leftrightharpoons \mathsf{N}$ be a wb lens, $a: A \to A'$ an update, and $r: A \leftrightarrow B$ a consistent corr. Let $(a, r).\mathsf{fPpg} = (r', b)$ and $A.\mathsf{fGen} = (r_{||}, B_{||})$. Then there is a unique private delta $\delta_{ar}: B_{||} \to B$ such that $r_{||}; \delta_{ar} = r'$. □

Thus, although a lens provides incremental update propagation in both directions, we can choose to make change propagation in one or both directions non-incremental by employing operations fGen and bGen defined above. In this way a lens determines several *computational frameworks* depending on which (if any) directions of update propagation are chosen to be non-incremental.

Definition 16 (Incremental specilaization). Let $\lambda: \mathsf{M} \leftrightharpoons \mathsf{N}$ be a wb lens. Its *incremental specialization* is an algebraic structure $\lambda^{\mathsf{inc}}: \mathsf{M} \leftrightharpoons \mathsf{N}$ comprising two operations $(\mathsf{Op}_1, \mathsf{Op}_2)$ defined as follows.

(i) If $\mathsf{Op}_1 = \mathsf{fGen}$ and $\mathsf{Op}_2 = \mathsf{bGen}$, i.e., neither direction is incremental, we call the specialization *non-incremental* and write $\lambda^{||}: \mathsf{M} \leftrightharpoons \mathsf{N}$.

(ii) If $\mathsf{Op}_1 = \mathsf{fPpg}$ and $\mathsf{Op}_2 = \mathsf{bPpg}$, i.e., both directions are incremental, we call the specialization *fully incremental*, and write $\lambda^{\asymp}: \mathsf{M} \leftrightharpoons \mathsf{N}$.

(iii) If only one direction is incremental, specialization is *semi-incremental*. If the incremental direction is from the source to the target, i.e., $\mathsf{Op}_1 = \mathsf{fPpg}$ while $\mathsf{Op}_2 = \mathsf{bGen}$, we write $\lambda^{>}: \mathsf{M} \leftrightharpoons \mathsf{N}$. If the incremental direction is from the target to the source, i.e., $\mathsf{Op}_1 = \mathsf{fGen}$ while $\mathsf{Op}_2 = \mathsf{bPpg}$, we write $\lambda^{<}: \mathsf{M} \leftrightharpoons \mathsf{N}$.

We will often call an incremental specialization $\lambda^{\mathsf{inc}}: \mathsf{M} \leftrightharpoons \mathsf{N}$ of a lens just a lens.

Theorem 7 (Incrementality taxonomy). Let $\lambda: \mathsf{M} \leftrightharpoons \mathsf{N}$ be a wb lens. There are four mutually exclusive and jointly complete logical possibilities (concrete incrementality types) for the incremental specialization $\lambda^{\mathsf{inc}}: \mathsf{M} \leftrightharpoons \mathsf{N}$ with $\mathsf{inc} \in \{||, \asymp, >, <\}$. The last two types are mutually dual, and can be grouped into almost concrete type of semi-incrementality. □

3.4. Lenses, incrementality and info-symmetry

Our work above demonstrates that the type of a lens $\lambda: \mathsf{M} \leftrightharpoons \mathsf{N}$ is determined by two indexes: one is its info-symmetry type $\mathsf{inf} \in \{\approx, \asymp, >, <\}$ (determined by the info-symmetry type of the underlying consistency framework), and the other is the type of its incremental specialization $\mathsf{inc} \in \{||, \asymp, >, <\}$. We will write a lens with double-indexing $\lambda^{\mathsf{inc}}_{\mathsf{inf}}$, and call each of so double-indexed specialization a *computational frameworks*. The total is 16 types of computational frameworks. However, amongst these 16 types, several are basically the same up to permutation of the source and the target, e.g., lenses $\lambda^{<}_{<}$ and $\lambda^{>}_{>}$ possess the same properties and dualization mutually converts them each into the other. The same is true for lenses of types $\lambda^{<}_{>}$ and $\lambda^{>}_{<}$. However, lenses $\lambda^{>}_{<}$ and $\lambda^{<}_{<}$ are essentially different. The lower inf-index shows that both lenses support view maintenance. The upper inc-index of lens $\lambda^{<}_{>}$ describes it as a pair (fPpg, bGen) that supports non-incrementally computed (by bGen) and updatable (by fPpg) view. In contrast, the inc-index of lens $\lambda^{>}_{>}$ describes it as a pair (fGen, bPpg) that supports incremental view computation but non-incremental view update propagation; the latter makes this computational framework non-applicable for databases (but it can still be used for model compilation and incremental reverse engineering). As Fig. **??** shows, there are 10 really different lens types.

4. Formalizing Organizational symmetry: Synchronization Cases and Types

Organizational symmetry is about change propagation, and in this section we first introduce a key notion of a changing model as a *trajectory* in the respective model space. Then we define a synchronization case as a pair of trajectories satisfying certain mutual synchronization conditions. Finally, we define organizational polices as special constraints on synchronization cases, which can be classified by their symmetry w.r.t. the source and the target models.

4.1. Models as trajectories

We consider a changing model as a *trajectory* in the respective model space. Let M be a metamodel, and $\mathsf{M} = (\mathsf{M}_\bullet, \mathsf{M}_\Delta)$ be the space of its instances and deltas as described above. A *model-as-trajector* is a mapping

$A: I \to \mathsf{M}_\bullet$, whose domain I is a finite linearly ordered set $\{i_0 < i_1 < \ldots < i_n\}$ of *version numbers* or *indexes*. Thus, A appears as the model's immutable identity whereas its state $A(i)$ changes as index i runs over I. To simplify notation, below we will write A_i for $A(i)$, and by the abuse of terminology often call model's states just models.

To traverse set I, we will use operations $i{-}1, i{-}2, ...$ and $i{+}1, i{+}2, ...$ with the evident meaning ($i_0 - 1$ and $i_n + 1$ are not defined).

If $i{\in}I$ is a version number and $i{-}1$ is its predecessor, we have a (directed) delta $a_i: A_{i-1} \to A_i$ specifying the change. We may consider the pair $(i{-}1, i)$ as an arrow from $i{-}1$ to i, and delta a_i as the A-image of this arrow in M_Δ. This makes I a directed graph, and A a graph mapping. Moreover, we can make I a category I with nodes $\mathsf{I}_\bullet = I$ and arrows $\mathsf{I}_\Delta = \{(i_1 i_2) \in I{\times}I: \ i_1 < i_2\}$, arrow composition $(i_1 i_2);(i_2 i_3) = (i_1 i_3)$ and identities ii. Then a model trajectory is a functor (a mapping of categories) $A: \mathsf{I} \to \mathsf{M}$, i.e., a graph mapping such that $A(i_1 i_3) = A(i_1 i_2); A(i_2 i_3)$ and $A(ii) = \mathsf{id}_{A_i}$. For any non-initial index i, we write a_i for delta $A(i{-}1, i): A_{i-1} \to A_i$ to be read "the update that created model A_i".

4.2. Synchronization cases

Synchronization of two models is about maintaining certain correspondences between two trajectories, say, $A: \mathsf{I} \to \mathsf{M}$ and $B: \mathsf{J} \to \mathsf{N}$, in two computationally related model spaces. We will assume the spaces are related by a wb delta lens $\lambda: \mathsf{M} \leftrightharpoons \mathsf{N}$ comprising operations of forward and backward delta propagation as described in Def. 10.

Partitioning of model deltas into private and public determines a similar partitioning of indexes $I = I^{\mathrm{prv}} \uplus I^{\mathrm{pub}}$ with

$$I^{\mathrm{prv}} = \left\{i \in I: \ a_i \in \mathsf{M}_\Delta^{\mathrm{prv}}\right\}, \text{ and } I^{\mathrm{pub}} = \left\{i \in I: \ a_i \in \mathsf{M}_\Delta^{\mathrm{pub}}\right\},$$

and similarly $J = J^{\mathrm{prv}} \uplus J^{\mathrm{pub}}$. Thus, $i{\in}I^{\mathrm{prv}}$ (or $i{\in}I^{\mathrm{pub}}$) means that model A_i is the result of a private (resp. public) update $a_i: A_{i-1} \to A_i$.

Since public updates destroy consistency, as soon as a public update is committed on one side, it must be propagated to the other side to restore consistency. According to Theorem 2, the propagated update is also public, but we call it *passive*, whereas the original update is *active*.

For example, suppose that we have registered a public update $a_i: A_{i-1} \to A_i \in \mathsf{M}_\Delta^{\mathrm{pub}}$ on the A-side, and hence $i \in I^{\mathrm{pub}}$. There are two possibilities for the case.

(a) Update a_i was initiated on the A side (we say a_i is *active*) and then was propagated to the B side. This means that there is a version number $i{\triangleright} \in J$ and update $b_{i\triangleright}: B_{(i\triangleright)-1} \to B_{i\triangleright}$ produced by this propagation (we then say that $b_{i\triangleright}$ is *passive*), so that the corr resulted from this propagation, $r_{i,i\triangleright}: A_i \leftrightarrow B_{i\triangleright}$, restores consistency. Considering all such updates gives us an order preserving bijection $\_{\triangleright}: I^{\mathrm{act}} \to J$ for some ordered subset $I^{\mathrm{act}} \subseteq I^{\mathrm{pub}}$.

(b) Update a_i is the result of propagation from the other side (now a is *passive*) of some (active) update $b_{i\blacktriangleright}: B_{(i\blacktriangleright)-1} \to B_{i\blacktriangleright}$ for some version number $i{\blacktriangleright} \in J$, and we again have a consistent corr $r_{i,i\blacktriangleright}: A_i \leftrightarrow B_{i\blacktriangleright}$ resulted from this backward propagation. Considering all such updates gives us an order preserving bijection $\_{\blacktriangleright}: I^{\mathrm{pas}} \to J$ for some ordered subset $I^{\mathrm{pas}} \subseteq I^{\mathrm{pub}}$. Clearly, sets I^{act} and I^{pas} are disjoint and their union is I^{pub}, $I^{\mathrm{pub}} = I^{\mathrm{act}} \uplus I^{\mathrm{pas}}$.

Viewing the same pair of trajectories from the B-side, gives us partitioning $J^{\mathrm{pub}} = J^{\mathrm{act}} \uplus J^{\mathrm{pas}}$ and order-preserving bijections $\triangleleft\_: I \leftarrow J^{\mathrm{act}}$ and $\blacktriangleleft\_: I \leftarrow J^{\mathrm{pas}}$. Moreover, for a correct synchronization case, we should have bijections $\triangleright$ and $\blacktriangleleft$ to be mutually inverse and set an isomorphism $I^{\mathrm{act}} \cong J^{\mathrm{pas}}$. Similarly, bijections $\blacktriangleright$ and $\triangleleft$ are to be mutually inverse as well, and $I^{\mathrm{pas}} \cong J^{\mathrm{act}}$.

Thus, for a pair of correctly synchronized trajectories as above, we have a partitioning $I^{\mathrm{pub}} = I^{\mathrm{act}} \uplus I^{\mathrm{pas}}$, a partitioning $J^{\mathrm{pub}} = J^{\mathrm{act}} \uplus J^{\mathrm{pas}}$, and two pairs of order-preserving bijections,

$$\_{\triangleright}: I^{\mathrm{act}} \to J^{\mathrm{pas}} \text{ and } \_{\blacktriangleright}: I^{\mathrm{pas}} \to J^{\mathrm{act}},$$

and

$$\triangleleft\_: I^{\mathrm{pas}} \leftarrow J^{\mathrm{act}} \text{ and } \blacktriangleleft\_: I^{\mathrm{act}} \leftarrow J^{\mathrm{pas}},$$

such that $\triangleright = \blacktriangleleft^{-1}$ and $\blacktriangleright^{-1} = \triangleleft$. Hence, we have an isomorphisms $\bowtie \subset I^{\mathrm{pub}}{\times}J^{\mathrm{pub}}$ (we could denote it by $\blacktriangleright\!\!\blacktriangleleft$ as well) such that for any pair $i{\triangleright\!\!\triangleleft}j \in I^{\mathrm{pub}}{\times}J^{\mathrm{pub}}$, models A_i and B_j are consistent via a corr $r_{i\triangleright\!\!\triangleleft j}$ computed during update propagation.

Figure 7 provides more details for the mechanism. Suppose that models A_i and B_j have been synchronized, and after that each of the models evolved with its own sequence of private updates (labeled with [prv] in the figure). Suppose that a public update $a_{i\oplus 1}$, where $i{\oplus}1$ denotes

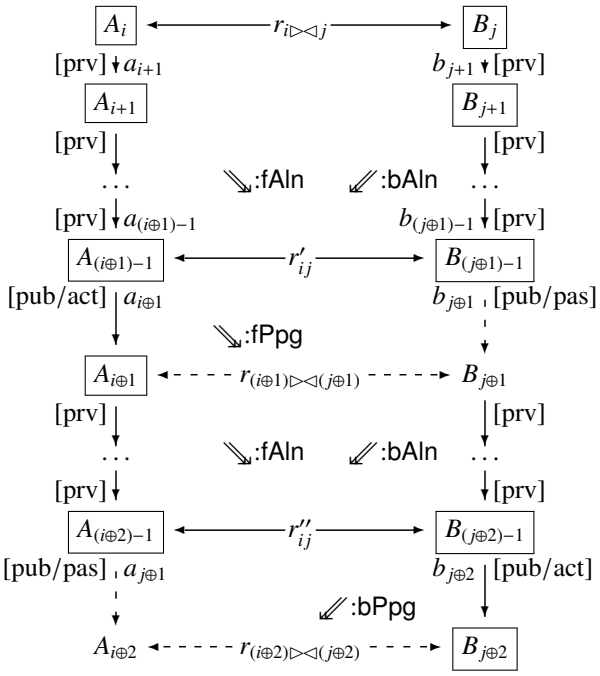


Figure 7: A general sync case

the next index in the set I^{pub} (which is different from the next index, i+1, in the entire set I,) was committed on the A-side and destroyed consistency. Hence, this update was propagated to side B to restore consistency. In order to preserve the results of private changes on the B side, this propagation has been done for the last corr r'_{ij} before the public update $a_{i\oplus1}$ as shown in the figure. This corr r'_{ij} is computed by formula (1)

$$[a_{i+1}; a_{i+2}; ...; a_{(i\oplus1)-1}] * r_{i\triangleright\triangleleft j} * [b_{j+1}; b_{j+2}; ...; b_{(j\oplus1)-1}] \quad (1)$$

where $j\oplus1$ denotes the immediate successor of j in J^{pub}, and * denotes the realignment operation specified in Def. 2 and 3. Note that the set of I-indexes from i to $i\oplus1$ and the set of J-indexes from j to $j\oplus1$ can be of different length. It is also possible that, e.g., $(i\oplus1) - 1 = i$, which means there were no private updates on the left side preceding the public one, in which case the left term in square brackets above is an identity update. The same holds for the right term in square brackets.

Now update $a_{i\oplus1}$ can be propagated to the right side by applying operation fPpg to the pair $(a_{i\oplus1}, r'_{ij})$, which results in update $b_{j\oplus1}$ and the new corr $r_{(i\oplus1)\triangleright\triangleleft(j\oplus1)}$. The latter is consistent because r'_{ij} is consistent (since previously only private update were made on both sides), and a correct propagation preserves consistency.

In the next synchronization step of the case, public update $b_{j\oplus2}$was first committed on the B-side, and hence propagated to the left side with operation bPpg. The input corr for the operation was again provided by the alignment framework, again computed by formula (1) with i and j replaced by $i\oplus1$ and $j\oplus1$ (and, respectively, $i\oplus1$ and $j\oplus1$ in formula (1) are replaced by $i\oplus2$ and $j\oplus2$). The result is a new pair of synchronized models $r_{(i\oplus2)\triangleright\triangleleft(j\oplus2)}$, and the system is ready to yet another synchronization step as above.

The next definition gives an accurate formalization of the discussion above. As mappings denoted by black triangles are derived from those denoted by blank triangles, we can specify the required correspondences by only using blank triangle mappings.

Definition 17 (Synchronization case). Given a fully incremental delta lens $\lambda^{\asymp}\colon \mathsf{M} \leftrightharpoons \mathsf{N}$, a *(consistent) synchronization case* is a pairs of trajectories, $A\colon \mathsf{I} \to \mathsf{M}$ and $B\colon \mathsf{J} \to \mathsf{N}$, with the following additional structure.

(a) Sets I^{pub} and J^{pub} are further partitioned, $I^{\text{pub}} = I^{\text{act}} \uplus I^{\text{pas}}$ and $J^{\text{pub}} = J^{\text{act}} \uplus J^{\text{pas}}$, and two isomorphisms are given: $\_\triangleright\colon I^{\text{act}} \to J^{\text{pas}}$ and $\triangleleft\_\colon I^{\text{pas}} \leftarrow J^{\text{act}}$. This establishes an isomorphism $\triangleright\!\triangleleft \subset I^{\text{pub}} \times J^{\text{pub}}$, and for each pair of corresponding indexes $i\triangleright\!\triangleleft j$, there is a consistent corr $r_{i\triangleright\triangleleft j}\colon A_i \leftrightarrow B_j$. Particularly, for the initial indexes we have $i_0\triangleright\!\triangleleft j_0$ and $r_0 = r_{i_0\triangleright\triangleleft j_0}$.

(b) Let $i\oplus1$ denotes the next index in the linearly ordered set I^{pub}, which, in general, may be greater than the next index i+1 in set $I \supset I^{\text{pub}}$ due to several private update indexes following i before $i\oplus1$. Similarly, $j\oplus1$ is the next index in J^{pub}.

If $i\triangleright\!\triangleleft j$ and $i\oplus1 \in I^{\text{act}}$, then $j\oplus1 \in J^{\text{pas}}$ and

$$a_{i\oplus1}.\mathsf{fPpg}(r'_{ij}) = b_{j\oplus1} \text{ and } r_{(i\oplus1)\triangleright\triangleleft(j\oplus1)} = r'_{ij}.\mathsf{fPpg}(a_{i\oplus1}),$$

where r'_{ij} is computed by (1).

If $i\triangleright\!\triangleleft j$ and $i\oplus1 \in I^{\text{pas}}$, then $j\oplus1 \in J^{\text{act}}$ and

$$a_{i\oplus1} = \mathsf{bPpg}(r'_{ij}).b_{j\oplus1} \text{ and } r_{(i\oplus1)\triangleright\triangleleft(j\oplus1)} = \mathsf{bPpg}(b_{j\oplus1}).r'_{ij}.$$

where r'_{ij} is again computed by the same formula (1).

Modifications of this definition for the cases of a non-incremental lens $\lambda^{\shortparallel}: \mathsf{M} \leftrightharpoons \mathsf{N}$ and a semi-incremental lens $\lambda^{>}: \mathsf{M} \leftrightharpoons \mathsf{N}$ or $\lambda^{<}: \mathsf{M} \leftrightharpoons \mathsf{N}$ are straightforward. □

We will denote a synchronization case as defined above by a double arrow $\sigma: A \rightleftarrows B$, and the set of all such cases by $\boldsymbol{Sync}_\lambda(A, B)$ with the subscript referring to the underlying lens providing the computational framework. Given a synchronization case σ, we will sometime subscript its constituting elements and write $A_\sigma: I_\sigma \to \mathsf{M}$ etc.

4.3. Organizational Polices and Types

Class $\boldsymbol{Sync}_\lambda(A, B)$ encompasses all possible synchronizations of models A and B. If a special *organizational* policy between the models is assumed, some synchronization cases can be a priori prohibited. For example, we can prohibit update propagation from A to B and thus make A an entirely passive receiver of changes from side B, if we require $I^{\text{act}}_\sigma = \varnothing$ for any case σ allowed by the policy. Dually, we make B a passive receiver of changes A by requiring $J^{\text{act}}_\sigma = \varnothing$ for any legal case σ. To establish a more refined organizational policy, we can prohibit propagating some, but not all, updates from either side so that both sets I^{act} and J^{act} are not empty. Thus, an org-policy specifies a special subclass of class $\boldsymbol{Sync}_\lambda(A, B)$.

Below we make these ideas formal.

Definition 18 (Org-policy). Let $\lambda: \mathsf{M} \leftrightharpoons \mathsf{N}$ be a wb lens.

(i) An *org-policy* is a pair of sets $\Pi = (P, Q)$ with $P \subseteq \mathsf{M}^{\text{pub}}_\Delta$ and $Q \subseteq \mathsf{N}^{\text{pub}}_\Delta$, whose elements are called *propagatable* deltas. We require Π to be *complete* in the following sense. Let $P.\lambda$ denotes all target updates propagated from P with lens λ, $P.\lambda \stackrel{\text{def}}{=} \{a.\mathsf{fPpg}(r)\colon\ a{\in}P, r{\in}\mathsf{K}\}$, and, dually, $\lambda.Q$ denotes the set of all source updates propagated from Q, i.e., the set $\{\mathsf{bPpg}(r).b\colon\ b{\in}Q, r{\in}\mathsf{K}\}$. Then we require $\mathsf{M}^{\text{pub}}_\Delta = P \cup \lambda.Q$ and $P.\lambda \cup Q = \mathsf{N}^{\text{pub}}_\Delta$.

(ii) We say a synchronization case $\sigma: A \rightleftarrows B$ *conforms* to policy $\Pi = (P, Q)$ and write $\sigma \models \Pi$, if $a_i \in P$ for all $i \in I^{\text{act}}_\sigma$, and $b_j \in Q$ for all $j \in J^{\text{act}}_\sigma$. In other words, in order to claim conformance $\sigma \models \Pi$, we require $\{a_i{\in}\mathsf{M}_\Delta\colon\ i \in I^{\text{act}}_\sigma\} \subset P$ and $\left\{b_j{\in}\mathsf{N}_\Delta\colon\ j \in J^{\text{act}}_\sigma\right\} \subset Q$.

In this way, an org-policy Π determines a corresponding synchronization *type*, i.e., a class of synchronization cases $[\![\, \Pi \,]\!] = \{\sigma{\in}\boldsymbol{Sync}_\lambda(A, B)\colon\ \sigma \models \Pi\}$ conforming to the policy.

***Remark* 5 (Organization vs. technology).** Importantly, sets P and Q constituting a policy are determined organizationally rather than technologically in the sense that propagation operations can be defined for non-propagatable deltas. For example, when code is generated from a UML model by a forward operation fPpg or fGen, the backward operation bPpg is defined for all code deltas, and is important for checking correctness of code generated from the model wrt. its conformance to the model as prescribed by the invertibility law. However, only some (or none) of code deltas are allowed to be propagated back to the model.

On the other hand, if some updates are not propagatable by technical reasons, e.g., in the database context, there is no a reasonable update propagation policy ensuring the uniqueness of the source database, these updates obviously cannot be organizationally allowed. That is, an organizational policy is built within the boundaries of the technical restrictions.

Definition 19 (Org-symmetries). Let $\Pi = (P, Q)$ is an organizational policy as defined above.

(a1) If both directions propagate all possible updates: $P = \mathsf{M}_\Delta$, $Q = \mathsf{N}_\Delta$, we call the policy *richly org-symmetric* and write $\Pi_{\asymp}: A \rightleftarrows B$ or $A \asymp_\Pi B$.

(a2) If both directions only propagate some of the possible updates: $\varnothing \neq P \subsetneq \mathsf{M}_\Delta$, $\varnothing \neq Q \subsetneq \mathsf{N}_\Delta$, we call the policy *poorly org-symmetric* and write $\Pi_{\gtrsim}: A \rightleftarrows B$ or $A \gtrsim_\Pi B$.

We call the policy *org-symmetric* if it is poorly or richly symmetric.

(b) If one direction propagates all, and the other some, of the updates, we call the policy *org-semi-symmetric*. That is, either (b1) $P = \mathsf{M}_\Delta$ and $\varnothing \neq Q \subsetneq \mathsf{N}_\Delta$, in which case we write $\Pi_{\gtrsim}: A \rightleftarrows B$ or $A \gtrsim_\Pi B$, or (b2) $\varnothing \neq P \subsetneq \mathsf{M}_\Delta$ and $Q = \mathsf{N}_\Delta$, and we write $\Pi_{\lesssim}: A \rightleftarrows B$ or $A \lesssim_\Pi B$.

(c) If one direction propagates all, and the other none, of the updates, we call the policy *(strictly) org-asymmetric*. That is, either (c1) $P = \mathsf{M}_\Delta$ and $Q = \varnothing$, in which case we write $\Pi_{>}: A \rightleftarrows B$ or $A >_\Pi B$, or (c2) $P = \varnothing$ and $Q = \mathsf{N}_\Delta$, and we write $\Pi_{<}: A \rightleftarrows B$ or $A <_\Pi B$.

Theorem 8 (Org-symmetry taxonomy). There are six mutually exclusive and jointly complete logical possibilities (concrete org-symmetry types): (a1) $A \asymp_\Pi B$, (a2)

$A \prec_\Pi B$; (b1) $A \preceq_\Pi B$, (b2) $A \succeq_\Pi B$; (c1) $A \succ_\Pi B$ and (c2) $A \prec_\Pi B$. They can be grouped in three abstract types: org-symmetry (a)=(a1)∨(a2), org-semi-symmetry (b)= (b1)∨(b2), and org-asymmetry (c)=(c1)∨(c2). Types (b) and (c) are almost concrete as types (b1) and (b2), as well as (c1) and (c2) are mutually dual. □

***Remark* 6.** Each of the six types is a property of a triple (A, Π, B) rather than a pair (A, B). However, we may use a loose notation and write, say, $A \prec_{\mathsf{org}} B$ meaning that we have a synchronization $\Pi_\prec : A \rightleftarrows B$ providing $A \prec_\Pi B$. This loose notation is used in [1].

***Remark* 7.** A very special subtype of this type is *interleaving*, for which, in addition, the following holds: $I^{\mathrm{prv}} = \varnothing = J^{\mathrm{prv}}$. In other words, the two models actually share the same set of version indexes, and all changes on either sides are at once propagated to the other side in the interleaving mode.

5. Related Work

A majority of work on semantic foundations for model transformations and bx assumes an operational rule-based semantics, e.g., Maude's term rewriting rules for ATL [13], transformations of symbolic graphs for QVT-R (the check-only mode) [14] and for general inter-modeling patterns in [15], or TGG as a general bx engine [16]. A much more declarative approach to bx semantics, in which update propagation procedures are considered as abstract algebraic operations (along the lines of the *lens* approach to the view update problem [17]), was proposed by Stevens in her seminal papers [18, 11] and developed in [19]. The original *state-based* lenses were later modified and subsumed by asymmetric [4] and symmetric [5] *delta* lenses; implementation of these delta lenses via TGG is described in, resp., [7] and [6].

The formal semantics presented above is a major development of the delta lens framework. We enrich the notion ofconsistency framework with the constructs of private and public updates, and show that asymmetry defined in terms of private updates coincides with asymmetry defined in terms of a view computation function. Also, we refine compositionality and invertibility laws for delta lenses by making them lax, and present some simple results about them. Accurate definitions of symmetry and duality of delta lenses are novel, and organization of the variety of delta lenses into a two-dimensional space is novel too.

The org-symmetry dimension has been discussed in the literature as unidirectional vs. bidirectional transformations [20, 16, 21]. We present a more fine-grained taxonomy by introducing organizational semi-symmetry, and give the dimension a formal semantics via the notion of a synchronization case.

References

[1] Z. Diskin, H. Gholizadeh, A. Wider, K. Czarnecki, A three-dimensional taxonomy for bidirectional model synchronization, Journal of System and Software.

[2] M. Johnson, R. D. Rosebrugh, Spans of lenses, in: K. S. Candan, S. Amer-Yahia, N. Schweikardt, V. Christophides, V. Leroy (Eds.), Proceedings of the Workshops of the EDBT/ICDT 2014 Joint Conference (EDBT/ICDT 2014), Athens, Greece, March 28, 2014., Vol. 1133 of CEUR Workshop Proceedings, CEUR-WS.org, 2014, pp. 112–118.
URL `http://ceur-ws.org/Vol-1133/paper-18.pdf`

[3] Z. Diskin, Model Synchronization: Mappings, Tiles, and Categories, in: J. M. Fernandes, R. Lämmel, J. Visser, J. Saraiva (Eds.), GTTSE, Vol. 6491 of Lecture Notes in Computer Science, Springer, 2009, pp. 92–165.

[4] Z. Diskin, Y. Xiong, K. Czarnecki, From State- to Delta-Based Bidirectional Model Transformations: the Asymmetric Case, Journal of Object Technology 10 (2011) 6: 1–25.

[5] Z. Diskin, Y. Xiong, K. Czarnecki, H. Ehrig, F. Hermann, F. Orejas, From state-to delta-based bidirectional model transformations: the symmetric case, in: MODELS, Springer, 2011, pp. 304–318.

[6] F. Hermann, H. Ehrig, F. Orejas, K. Czarnecki, Z. Diskin, Y. Xiong, Correctness of model synchronization based on triple graph grammars, in: MODELS, Springer, 2011, pp. 668–682.

[7] A. Anjorin, S. Rose, F. Deckwerth, A. Schürr, Efficient model synchronization with view triple graph grammars, in: Modelling Foundations and Applications, Springer, 2014, pp. 1–17.

[8] Z. Diskin, Model synchronization: mappings, tile algebra, and categories, in: R. Lämmel et al. (Ed.), Postproceedings GTTSE 2009, LNCS#6491, Springer, 2011.

[9] Z. Diskin, T. Maibaum, K. Czarnecki, Intermodeling, queries, and kleisli categories, in: Fundamental Approaches to Software Engineering, Springer, 2012, pp. 163–177.

[10] Z. Diskin, S. Kokaly, T. Maibaum, Mapping-aware megamodeling: Design patterns and laws, in: Software Language Engineering, Springer, 2013, pp. 322–343.

[11] P. Stevens, Bidirectional model transformations in QVT: semantic issues and open questions, Software and System Modeling 9 (1) (2010) 7–20.

[12] Z. Diskin, An algebraic semantics for bidirectiomnal model synchornization, Tech. Rep. GSDLab-TR 2014-04-01, University of Waterloo (2014).
URL `http://gsd.uwaterloo.ca/node/588`

[13] J. Troya, A. Vallecillo, A rewriting logic semantics for ATL, Journal of Object Technology 10 (2011) 5: 1–29.

[14] E. Guerra, J. de Lara, An algebraic semantics for qvt-relations check-only transformations, Fundam. Inform. 114 (1) (2012) 73–101.

[15] E. Guerra, J. de Lara, F. Orejas, Inter-modelling with patterns, Software and System Modeling 12 (1) (2013) 145–174.

[16] A. Schürr, F. Klar, 15 Years of Triple Graph Grammars, in: ICGT, 2008, pp. 411–425.

[17] J. N. Foster, M. B. Greenwald, J. T. Moore, B. C. Pierce, A. Schmitt, Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem, ACM Transactions on Programming Languages and Systems (TOPLAS) 29 (3) (2007) 17.

[18] P. Stevens, Bidirectional model transformations in qvt: Semantic issues and open questions, in: Model Driven Engineering Languages and Systems, Springer, 2007, pp. 1–15.

[19] Z. Diskin, Algebraic Models for Bidirectional Model Synchronization, in: MoDELS, Vol. 5301 of LNCS, Springer, 2008, pp. 21–36.

[20] M. Antkiewicz, K. Czarnecki, Design Space of Heterogeneous Synchronization, in: R. Lämmel, J. Visser, J. Saraiva (Eds.), GTTSE, Vol. 5235 of Lecture Notes in Computer Science, Springer, 2007, pp. 3–46.

[21] K. Czarnecki, J. N. Foster, Z. Hu, R. Lämmel, A. Schürr, J. Terwilliger, Bidirectional Transformations: A Cross-Discipline Perspective, in: ICMT, 2009.