Algebra of bidirectional model synchronization[⋆]

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

Zinovy Diskin

zdiskin@cs.toronto.edu

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

1 Introduction

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

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

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

many models n such that $m \succ a \prec n$ or $m \prec b \succ n$, and symmetrically for n. It implies that synchronization is, in general, a pair of *multi-valued* mapping between the classes of models. To make these mappings single-valued, some *synchronization policy* has to be adopted.

The second class of synchronization scenarios appears in model transformations (MT). Consider, for example, refinements of high-level models by more detailed ones, or implementation of UML models by Java programs. Synchronization procedures should be provided for both directions because either of the models, the source or the target one, can be modified. Consistency between models m and n now means that n is a possible correct transform/implemetation of m, and it is again a multi-valued rather than functional relation in both directions (e.g., the same UML model can be implemented differently, and different models can have the same implementation). Hence, some synchronization policy is needed again.

As a rule, a reasonable policy is based on information contained in the old (before updates) models. Suppose we begin with two consistent models m and n, and then model n is modified to n'. To synchronize models and compute m', we may need (besides model n') information contained in models m and n, or perhaps only in m, or perhaps n' alone would suffice. Thus, we can distinguish *triple-model-*, *di-model-* or *single-model-based* synchronizations. In more detail, synchronization essentially depends on what data structures the models are (trees, XML documents, relational tables,...), and what operations over them are used in view definitions (hidden in symbols $\prec$ and $\succ$ above) or in model transformations. The richer these operations are, the more complex synchronization policies are. The issue is well known in the database community as the *view update problem*, and has been studied there since early 80s (see [8] for a brief survey). More recently, it has also been studied in the bi-directional programming community [12], and in replica synchronization, most notably, in work of the Harmony group [8]. The issue is also well-known to MDD practitioners, but its theoretical exploration in the modeling community only recently began [14, 1]. Particularly, it is shown in [14] that model synchronization is tightly connected to the QVT-style of model transformation [13].

The goal of the present paper is to analyze a variety of synchronization procedures and patterns in an abstract semantics framework. We follow the *syntax-free* (or *schema/metamodel-free*) style of formal modeling of model management procedures developed in [11, 8, 3, 14, 1] and also ignore metamodels and their mappings. In this approach, having a metamodel amounts to having a class of models M. Then, given two classes M and N, a forward triple-model-based synchronization policy appears as a function from $M \times M \times N$ to N, which takes a source model m, its updated version m' and the N-counterpart of m, n, as the arguments, and returns the updated version n' of model n. The type of backward synchronization policies is a symmetric mapping $N \times N \times M \to M$ returning the updated model m' for a triple (n, n', m). Somewhat surprisingly, interesting observations on the nature of synchronization can be made, and useful concepts emerge, even in this simple algebraic framework. Nevertheless, having the diversity and complexity of concrete data structures involved in the synchronization scenarios, one may doubt the usefulness of abstract semantic considerations. These concerns are quite reasonable but not indisputable. The following quote from [14] explains the position:

> ... it may seem foolhardy to write a paper which approaches semantic issues in bidirectional model transformations from first principles. However, there is currently a wide gap between what is desired for the success of MDD and what is known to be soundly supportable by graph transformations; the use of QVT-style bidirectional transformations has not spread fast, despite the early availability of a few tools, partly (we think) because of uncertainty among users over fundamental semantic issues.

An important pro-semantics argument is the success of synchronization tools developed by the Harmony group: they have designed complex and effective procedures for replica synchronization, which are based on firm semantic foundations [8]. In the paper we take the semantic argument for granted, and pursue the "foolhardy" syntax-free approach even further by analyzing and scrutinizing the first principles stated in [14]. We will see that there are two essentially different types of synchronization, selective and incremental: the former is most naturally formalized with di-model-based mappings while the latter is triple-model-based. Moreover, the two synchronization types obey different algebraic laws. We argue that the Check-then-enforce law required by QVT (it was called Hippocraticness in [14]) is natural for selective synchronization but may be too restrictive for incremental one. On the other hand, the History Ignorance law and its immediate consequence, Undoability, are natural for incremental synchronization but do not generally hold for selective one. Nevertheless, it will be proven that in the case of functional consistency relations, the difference between selective and incremental synchronization collapses and we have trivially Hippocratic systems (which may enjoy but not necessarily history ignorance and undoability).

Related work and contributions of the paper. A long-standing tradition in modeling model synchronization can be referred to as *algebraic semantics* because synchronization procedures are considered as algebraic operations regulated by algebraic equations (laws). It can be traced back to early work on the view update problem (VU), particularly to seminal papers by Bancilhon and Spyratos [2] and Dayal and Bernstein [4]. This algebraic style was continued by Meertens in [11] in the context of constraint maintenance (CM) far more general than VU, and more recently was further elaborated in the works of the Harmony group on so-called lenses [8] (but again within the VU-perspective). An adaptation of the CM-approach for bi-directional MT is developed by Stevens in [14]. A much broader and less formal perspective on model synchronization is presented by Antkiewicz and Czarnecki in [1]. The paper aims to classify a wide spectrum of synchronization procedures and tools within a unified framework. Updates are modeled as functions over model spaces and single-valuedness of synchronizers is achieved through so-called decision functions. Algebraic laws/equations are not considered because [1] focuses on types (signatures) of the synchronization procedures rather than their algebra. The paper describes more than a dozen of different model synchronization types, including those that we will consider in the paper, but their precise algebraic treatment needs further work. As mentioned above, this algebraic tradition focuses exclusively on semantics and is syntax-free (metamodeling-free). In contrast, the work [5] presents an algebraic framework encompassing both syntax and semantics of MT but in the particular case of functional (rather than relational) transformations; model synchronization is not considered. Another approach also encompassing both syntax and semantics is based on graph transformations, in which triple graph grammars [10] are especially relevant for model synchronization and transformation [7].

The present paper is within the relational syntax-free algebraic approach, and its contributions are as follows. The ideas of Meertens and Stevens are adapted for modeling synchronization of overlapping models (the first class of scenarios mentioned above), and analyzed within the context of selective-vs-incremental synchronization (Sect. 2.2 and 2.3).The notion of coherent transformation system [14] is refined by (a) considering partial synchronizing functions, (b) introducing conditions of history Ignorance and new version of Undoability that is more practically applicable, and (c) considering weak invertibility of synchronizers as a possible substitute for Hippocraticness (Sect. 3.2).[1] An important result of the paper is Theorem 5 on p.10 that specifies relationships between the formalisms. Also, a novel algebraic model of updates and update translation is presented, and its abstract version – updatability – is motivated as well (Sect. 2.3.1-2). On this base, a new *trigonal* model of incremental synchronization is built (Sect. 3.3). Finally, two new ways of composing elementary synchronization systems are introduced (Sect. 4), which seem to be widely applicable in MDD practice. To make the paper self-contained, the view update problem is sketched in Sect. 2.1.

2 Types of synchronization and Undoability

2.1 The view update problem

We consider a toy example illustrating the issue. Suppose that we have a database storing a triple of rational numbers, $m = (x, y, z)$, and a view to it storing the first and third components, $n = (x, z)$. If the view is updated to $n' = (x', z')$, we can easily translate it back and define the updated database state to be $m' = (x', y, z')$. However, if the view definition involves operations with records, the situation becomes much more complicated. For example, suppose that the view is defined by setting $n = (s, z)$ with $s = x + y$ an operation (query) over the database state. If the view is updated to $n' = (s', z')$, computing the updated database is problematic: we know that $x' + y' = s'$ but it is not enough to define x' and y'. To make the latter certain, we need one more condition for x' and y'. For example, we may assume that $x' = x$ and then $y' = s' - x$, or that $x' = x + \Delta s/2$, $y' = y + \Delta s/2$ where $\Delta s = s' - s$, or that $x'/y' = x/y$. Each of these *update policies* uniquely determines an updated database state. Thus, if M and N denote the spaces of database states and view states respectively, the choice of an update policy means that we have a function $put\colon N \times M \to M$ (where *put* stands for "put back") while the very view definition amounts to a function $get\colon M \to N$ ("get" the view value). Of course, getting the view of the updated database should produce the updated view, $get(put(n', m)) = n'$. This is the (PutGet) law in [8].

[1] We also introduce a new notion of undoable lens (Sect. 3.1).

Putting back the original (not updated) view should not change the database, $put(get(m), m) = m$; this is the (GetPut) law in [8].

In applications, it is often more convenient to consider the space of view states bigger (or equal) than the range of get, $get(M) \subseteq N$. It may be even necessary like in our example of two-views system $M \xrightarrow{get_M} A \xleftarrow{get_N} N$ considered on page 1, if the ranges of get functions are different. By the (PutGet) law, non-surjectivity of get implies partiality of put because otherwise, for any $n' \in N$ we take any $m \in M$ and find that $n' = get(put(n', m))$. Thus, for any fixed database state m, $Dom[put(-, m)] \subseteq get(M)$.[2] The exact equality would be an additional yet reasonable requirement: any real view can be propagated back. In this way we come to the notion of an *updatable* or *dynamic* view: it is a view definition (function get) equipped with a certain partial policy of update translation (put), which satisfy the equalities above. The Harmony group calls such a pair a *lens* (whose top and bottom edges are functions get and put).

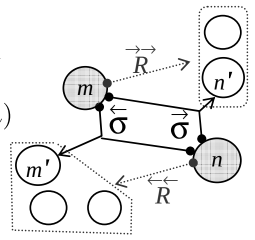
2.2 Selective synchronization

Let M be a class of models (say, in UML), N a class of programs (say, in Java) and predicate mRn means that model m is implemented by program n. Importantly, relation R is not functional: there may be many implementations of the same model and different models can be implemented by the same program. In other words, mappings $\overset{\twoheadrightarrow}{R}\colon M \twoheadrightarrow N$, $\overset{\twoheadrightarrow}{R}(m) \overset{\text{def}}{=} \{n \in N \mid mRn\}$ and $\overset{\twoheadleftarrow}{R}\colon N \twoheadrightarrow M$, $\overset{\twoheadleftarrow}{R}(n) \overset{\text{def}}{=} \{m \in M \mid mRn\}$, are generally multi-valued. This is a typical situation for many cases of MT (see [14]. Another aspect is that mappings $\overset{\twoheadrightarrow}{R}$, $\overset{\twoheadleftarrow}{R}$ may result in empty sets (are partially defined), which is equivalent to having their inverses $\overset{\twoheadleftarrow}{R}$, $\overset{\twoheadrightarrow}{R}$ non-surjective. For example, not every Java program can be reverse engineered into UML due to the lack of expressiveness of the latter (e.g., [6] shows this for class diagrams). Thus, in general, the *(model) consistency* relation R is neither total nor surjective nor single-valued in either of the directions. Arguably, this schema is assumed by the QVT Standard [13] (see [14] for a detailed discussion).

The following scenario is typical. Given first a pair of consistent models mRn, one of the models, say n, changes to n'. The goal is to synchronize models and find a suitable model $m' \in M$ consistent with n', i.e., $m'Rn'$. The set of all M-models consistent with n' is given by $\overset{\twoheadleftarrow}{R}(n')$, that is, we need to select a unique model $m' \in \overset{\twoheadleftarrow}{R}(n')$; hence the name *selective* synchronization. Some selection policy is required and a natural idea is to employ the model m that we already have. For example, we may try to introduce some metrics in the space M and look for the model $m' \in \overset{\twoheadleftarrow}{R}(n')$ closest to m. In this way we obtain a function $M \xleftarrow{\overleftarrow{\sigma}} N \times M$ called the *backward synchronizer*. Symmetrically, we have also a *forward*

[2] Given a binary function $f\colon X \times Y \to Z$ and an element $y \in Y$, we denote by f_y the function from X to Z defined by $f_y(x) \overset{\text{def}}{=} f(x, y)$. We also write $f(-, y)$ for f_y.

synchronizer $M \times N \xrightarrow{\overrightarrow{\sigma}} N$ as illustrated by the diagram on the right). In this diagram, arrows denote applications of mappings to their arguments. For example, $n' = \overrightarrow{\sigma}(m, n)$ and $m' = \overleftarrow{\sigma}(n, m)$. Also, the two-model set in the right-upper corner is $\overset{\twoheadrightarrow}{R}(m)$, and the three-model set in the left-lower corner is $\overset{\twoheadleftarrow}{R}(n)$.



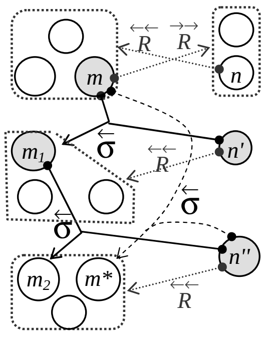
$\overset{\twoheadrightarrow}{R}$
m
n'
$\overleftarrow{\sigma}$
$\overrightarrow{\sigma}$
m'
n
$\overset{\twoheadleftarrow}{R}$

This framework was proposed in [11], where many theoretical results and examples can be found, and then adopted for MT in [14]. We will now revise the basic assumptions. A fundamental law of any synchronization is *correctness*: synchronization procedure must return consistent models, $m'R[\overrightarrow{\sigma}(m', n)]$ and $[\overleftarrow{\sigma}(n', m)]Rn'$ hold for all $m, m' \in M$,$n, n' \in N$. We call these conditions *range correctness*.

Since we want to consider partial consistency relations ($\overset{\twoheadrightarrow}{R}$ and $\overset{\twoheadleftarrow}{R}$ may result in empty sets), range correctness implies that synchronizers must be only partially defined, otherwise we come to a contradiction: $[\overleftarrow{\sigma}(n', m)]Rn'$ means that $n' \in \overset{\twoheadrightarrow}{R}(M)$ for any $n' \in N$. Thus, to keep partiality of R in the direction from N to M, i.e., to have $\overset{\twoheadrightarrow}{R}(M) \subsetneqq N$, we must keep $Dom[\overleftarrow{\sigma}(-, m)] \subseteq \overset{\twoheadrightarrow}{R}(M)$. It is reasonable to require the exact equality here: if $n' \in \overset{\twoheadrightarrow}{R}(M)$ and hence $\overset{\twoheadleftarrow}{R}(n') \neq \emptyset$, a suitable $m' = \overleftarrow{\sigma}(n, m)$ can be found. In other words, if a model n ***can*** be synchronized, then the backward synchronization procedure is defined for n (and provides a correct synchronization by the range correctness). Thus, we postulate $Dom[\overleftarrow{\sigma}(-, m)] = \overset{\twoheadrightarrow}{R}(M)$ and symmetrically $Dom[\overrightarrow{\sigma}(-, n)] = \overset{\twoheadleftarrow}{R}(N)$ (*domain correctness*).

Another basic law of selective synchronization is that if the old model happens to be in the new selection set, $m \in \overset{\twoheadleftarrow}{R}(n')$, then synchronization does nothing, $m' = m$. That is, $\overleftarrow{\sigma}(n', m) = m$ and, symmetrically, $\overrightarrow{\sigma}(m', n) = n$ if $m'Rn$. In other words, synchronization does not affect ("harm") models that are already consistent, hence the name *Hippocraticness* adopted in [14] for this law.

In [14], one more synchronization law, *undoability*, was proposed. It says that if all changes to the model were undone, for example, if we have started from a consistent pair mRn, then updated the target to n' and then come back to n, the synchronous counterpart in the space M must come back to m too. Undoability is a direct consequence of a stronger property of *history ignorance*. Let mRn and $n_0 = n$, $n_1 = n'$, $n_2 = n''$, $\ldots n_k = n^{(k)}$ is a sequence of updates on the target side, and $m_0 = m$, $m_1 = \overleftarrow{\sigma}(n_1, m_0)$, $m_2 = \overleftarrow{\sigma}(n_2, m_1)$,$\ldots m_k = \overleftarrow{\sigma}(n_k, m_{k-1})$ is the corresponding sequence of synchronized source models. History ignorance means that $m_k = \overleftarrow{\sigma}(n_k, m_0)$: it is the final result/state of model n that matters rather than the way of evolution. Specifically, when $n_k = n_0$, history ignorance (plus Hippocraticness) imply *undoability*. Undoability seems to be a very natural requirement yet (surprisingly) selective synchronization should *not* obey Ignorance nor Undoability laws. The reasons are demonstrated by diagram on the right, in which it is assumed that selection metrics is given by the geometrical distance between the points representing models. Then in the set $\overset{\leftleftarrows}{R}(n'')$, model $m_2 = \overleftharpoonup{\sigma}(n'', m_1)$ is the closest to m_1, but the model closest to $m = m_0$ is $m^* = \overleftharpoonup{\sigma}(n'', m)$ rather than m_2. This has happened because in the first synchronization (m and n'), a special configuration of the set $\overset{\leftleftarrows}{R}(n')$ "deviated" the model $m_1 = \overleftharpoonup{\sigma}(n', m)$ from "the vertical" m-m^*.


$\overset{\leftleftarrows}{R}$
$\overset{\rightrightarrows}{R}$
m
n
m_1
$\overleftharpoonup{\sigma}$
n'
$\overset{\leftleftarrows}{R}$
$\overleftharpoonup{\sigma}$
$\overleftharpoonup{\sigma}$
n''
m_2
m^*
$\overset{\leftleftarrows}{R}$

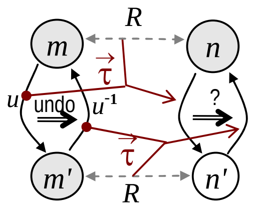
2.3 Incremental synchronization

2.3.1 Update translation. We consider the same synchronization scenario as above. Given consistent models mRn, model n changes to n' and we aim at finding its consistent counterpart m'. The idea of incremental synchronization is that if the changes from n to n' are not too significant, it is not reasonable to recompute the model $m' = \overleftharpoonup{\sigma}(n', m)$ from scratch. Rather, having an update (an "edit sequence") v such that $n' = n.v$, we translate it back to the source side to obtain a source model update $u = \overleftharpoonup{\tau}(v)$ to be applied to the old source model, $m' = m.u$. In general, the update translation is multi-valued and we employ the previous version of the model m to make it certain. Thus, the update translation functions have the types $\overleftharpoonup{\tau} : V \times M \to M$, $\overrightharpoonup{\tau} : U \times N \to N$, where V, U are the spaces of updates on the target and the source sides respectively.

In the context of incremental synchronization, Hippocraticness does not seem obligatory. Indeed, a model can contain some information that does not affect its consistency yet is useful. Hence, if even the updated model n' is consistent with "old" m, we may still want to update the latter to m' to make a better match with n'. On the other hand, history ignorance and undoability, which are somewhat foreign for selective synchronization, appear as quite natural requirements for incremental one. We will consider them in more detail but first we need to make the notion of update more certain.

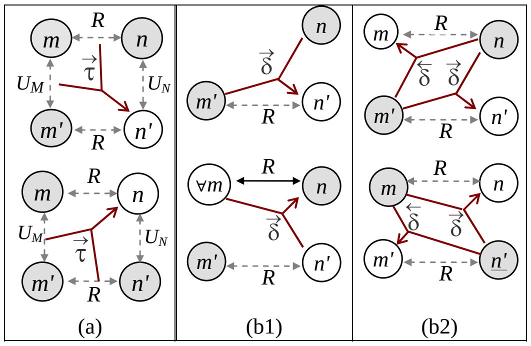
2.3.2 Modeling updates. There are a few approaches in the literature: functional (in which updates are functions/procedures over the model spaces [4, 9, 1]), history-based (updates are edit sequences), state-based (an update is just a new state [8]). In the paper we present a new model, which can be classified as an abstract history-based one. We first consider a space of models M as a directed graph, whose nodes are models and arrows are updates. The latter can be thought of as edit sequences. Because there may be different updates leading to the same result, it would be a multi-graph (many arrows between the same nodes are allowed). Evidently, arrows-updates could be composed (just concatenate the sequences), and each node should have a loop arrow denoting the trivial "do-nothing" update (the empty sequence). The possibility to undo changes means that updates are invertible: for any arrow $u\colon m \to m'$ there is an arrow $u^{-1}\colon m' \to m$ such that $u; u^{-1} = 1_m$, where ; denotes update composition and 1_m denotes the "do-nothing" update of the node m. Similarly, $u^{-1}; u = 1_{m'}$.

Compatibility of update translation with undoing, i.e., undoability, is demonstrated by diagram on the right. Given an update $u\colon m \to m'$ and a model n consistent with m, we translate u into an update $v = \overrightarrow{\tau}(u,n)\colon n \to n'$ on the target side, while the inverse u^{-1} of u is translated, in general, into $w = \overrightarrow{\tau}(u^{-1},n')\colon n' \to n''$. Undoability of translation means that $w = v^{-1}$, that is, $n'' = n$ and $v; w = 1_n$. Evidently, if translation τ is compatible with (i) update composition, $\overrightarrow{\tau}(u; u^{-1}, n) = \overrightarrow{\tau}(u,n); \overrightarrow{\tau}(u^{-1},n')$, and (ii) "do-nothing", $\overrightarrow{\tau}(1_m, n) = 1_n$, then τ is undoable.



The two latter conditions were studied in [9] in the setting of functional updates. Our model is more adequate to practice than functional: an edit sequences is an artifact attached to a particular model – the source model of the sequence, rather than a function over the entire space of models. It is also technically simpler to work with. Yet we leave this model for a future work and in the present paper consider an even simpler notion closer to the framework of lenses [8] and coherent transformations of [14]. To wit, rather than considering updates themselves, we will deal with a binary *updatability* relation $U_M \subset M \times M$: $(m, m') \in U_M$ iff there is an update $u\colon m \to m'$. Relation U_M is reflexive (owing to "do-nothing" updates), symmetric (undoing) and transitive (update composition). Hence, updatability is an equivalence relations, and the model space is partitioned into classes of mutually updatable models. Assumption that there is more than one such class means that *not* any two model can be updated to each other. Thinking extensionally, this assumption may seem counter-intuitive: just entirely delete one model and then build the other one. However, if models embody some contextual information (e.g., authorship and authorization), the existence of mutually non-updatable models becomes reasonable. Thus, a model space is a pair (M, U_M) with M a set and U_M an equivalence relation over M. Update translation is now modeled by a ternary function $\overrightarrow{\tau}\colon M \times M \times N \to N$ as shown by the upper diagram (a) below, and undoability means that if $n' = \overrightarrow{\tau}(m, m', n)$ then $n = \overrightarrow{\tau}(m', m, n')$ (consider both diagrams (a) together).

2.3.3 Diagonal schema revisited. We return to the two-model-based (diagonal) synchronization considered on p.5 but now with the incremental interpretation and with focus on undoability. Specifying the latter needs three models as demonstrated by diagrams (a) below while in the diagonal schema we have only two – the "old" model on the source side, m, is absent (the upper diagram in (b1)). To overcome this obstacle, it was proposed in [14] to consider any model consistent with n as a possible "old" model m. Then undoability is specified by the following implication: if $n' = \overrightarrow{\delta}(m', n)$ and $R(m,n)$ then $\overrightarrow{\delta}(m, n') = n$ as shown by the pair of diagrams (b1). However, applicability of this condition to many possible m makes undoability a rather strong imposition that, as is noted in [14], may be hard to meet.



We propose another version of the condition specified by diagrams (b2). We consider the only "old" source model provided by the pair (m', n), namely, that one given by the backward synchronization: if $n' = \overrightarrow{\delta}(m', n)$ and $m = \overleftarrow{\delta}(n, m')$ then $\overrightarrow{\delta}(m, n') = n$ (undo on the target side) and $\overleftarrow{\delta}(m, n') = m'$ (undo on the source side). Note that in this specification of Undoability, it appears as an integral property of the pair of synchronizers rather than a semi-synchronizer's individual property.

3 Formal semantic models and their relationships

In this section we transform semi-formal descriptions above into formal definitions. Let M, N be two spaces of data structures (trees, lists, models, databases).

3.1 View updates

Definition 1 (Dynamic views or lenses)**.** A *lens* from M to N is a pair $\lambda = (get_\lambda, put_\lambda)$ of functions: total $get_\lambda\colon M \to N$ and partial $put_\lambda\colon N \times M \to M$ (diagram (a) in Table 1). A lens is called *well-behaved* if conditions (Dom), (GetPut) and (PutGet) in Table 1 below are satisfied. If condition (PutPut) also holds (diagram (c)), the lens is called *very* well-behaved or *history ignorant*. A lens is called *undoable* if conditions (PutGet) and (Undo) hold (consider diagrams (a) and (b) together). Finally, a lens λ is called *total* if put_λ is a total function.

Below we omit the label λ if it is clear from the context. The class of all well-behaved lenses over $M \times N$ will be denoted by $\boldsymbol{Lens}(M, N)$.

Remark 1. Definitions of lens and (very) well-behaved lens (but without condition (Dom)) were introduced by the Harmony group. The notion of undoable lens is new.

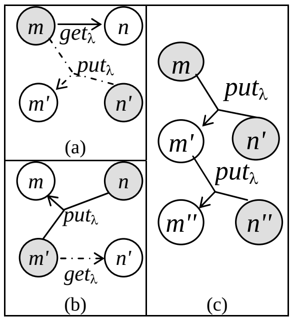
Proposition 1. Conditions (PutPut) and (GetPut) imply (Undo). Hence, a very well-behaved lens is undoable.

Proof. Write (PutPut) for $n'' = m.get$ and apply (GetPut) to the right part. □

Example 1. Each of update polices described in our toy example in section 2.1 determines a very well-behaved (hence, undoable) lens. (For the policy $x'/y' = x/y$,

Table 1: Conditions for lenses. All formulas are universally quantified for all their variables ($\forall m, m', m'' \in M, \forall n, n', n'' \in N$). Symbol $\Downarrow$ means definability.

Condition:	View def. $get_\lambda : M \to N$	Backward translation $put_\lambda : N \times M \to M$
(Dom)	$put_\lambda(m'.get_\lambda, m) \Downarrow$	
(GetPut)	$m = put_\lambda(m.get_\lambda, m)$	
(PutGet)	$put_\lambda(n', m).get_\lambda = n'$	
(Undo)	$m = put_\lambda(m.get_\lambda, put_\lambda(n', m))$	
history Ignorance		
(GetGet)	trivial	N/A
(PutPut)	N/A	let $m' = put_\lambda(n', m)$ then $put_\lambda(n'', m') = put_\lambda(n'', m)$

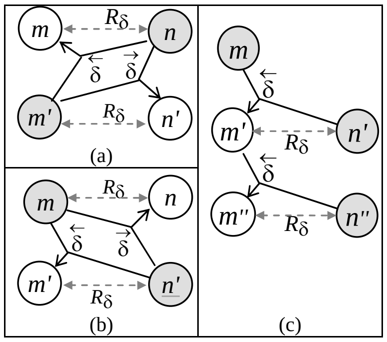


it follows from the distributivity laws). However, if for the database $m = (x, y, z)$ and the view $n = (x, y)$, we define the update policy $put(m, n') = (x', y', z + 1)$ so that the third component serves as a counter, then neither (PutPut) nor (Undo) hold.

3.2 Diagonal (two-model-based) synchronization

As it was noted in [14], a bi-directional MT program is a specification that can be read and interpreted in three different ways: as a specification of some consistency relation between the models, as a forward synchronizer and as a backward one.

Definition 2. A *diagonal synchronization system* (*di-system* in short) over $M \times N$ is a triple $\delta = (R_\delta, \overrightarrow{\delta}, \overleftarrow{\delta})$ with $R_\delta \subset M \times N$ a binary *consistency relation*, and $\overrightarrow{\delta} : M \times N \to N$ and $\overleftarrow{\delta} : N \times M \to M$ are two partial functions called the *forward* and *backward synchronizers* respectively (see diagram (a) below on the right). We call a di-system *well-behaved* if conditions (Corr) and (wInv) in Table 2 are satisfied. It is *very* well-behaved or *history ignorant* if, in addition, condition (hIgno) in the table holds (diagram (c)). The system is *undoable* if two (Undo) equations (with 's/t' standing for Undo on the source/target sides) hold (see diagrams (a,b) together). A di-system is *total* if both synchronizers are total functions.



Remark 2. The idea of diagonal synchronization was proposed by Meertens [11] in the context of user interface design. What he called a constraint maintainer is a total di-system satisfying (rgCorr) and (Hipp). For model transformation, this notion was adopted by Stevens [14] and augmented with a version of the undoability condition stronger than our (Undo) (see section 2.3.3). Precise relationships between the notions are described in Theorem 5 below.

Table 2: Conditions for diagonal synchronization systems. All formulas are universally quantified for all their free variables To ease reading formulas, we write $[m,n\rangle$ and $\langle m,n]$ for $\overrightarrow{\delta}(m,n)$ and $\overleftarrow{\delta}(n,m)$

Condition: name and label	Forward synch. $\overrightarrow{\delta} = [*,*\rangle\colon M\times N\to N$	Backward synch. $M \leftarrow N\times M : \langle *,*] = \overleftarrow{\delta}$
Correctness (Corr):	// see discussion in sect. 2.3	
Domain (dmCorr)	$[m,n\rangle\Downarrow$ iff $m\in\overleftarrow{R}_\delta(N)$	$\langle m,n]\Downarrow$ iff $n\in\overrightarrow{R}_\delta(M)$
Range (rgCorr)	$mR_\delta[m,n\rangle$	$\langle m,n]R_\delta n$
weakInvertibility (wInv):	// see diagram (a) above	
(wFwdBck) or (wFB)	$m' = \langle m', [m',n\rangle]$	
(wBckFwd) or (wBF)	$[\langle m',n],n\rangle = n$	
hist. Ignorance (hIgno) //consider diagram (c)	$\dfrac{\text{let } [m',n\rangle = n'}{\text{then } [m'',n'\rangle = [m'',n\rangle}$	$\dfrac{\text{let } m' = \langle m,n']}{\text{then } \langle m',n''] = \langle m,n'']}$
Undoability (Undo)	// consider diagrams (a,b) together	
(tUndo)	$[\langle m',n],[m',n\rangle\rangle = n$	
(sUndo)	$m' = \langle\langle m',n],[m',n\rangle]$	
Hippocraticness (Hipp) or Check-then-enforce	$mR_\delta n \Rightarrow [m,n\rangle = n$	$mR_\delta n \Rightarrow m = \langle m,n]$

Proposition 2. The following implications hold:
(i) (hIgno) and (wInv) imply (Undo)
(ii) (Corr) and (Hipp) imply (wInv)

Proof. (i) To prove (sUndo), write $\overleftarrow{\text{(hIgno)}}$ for $n'' = [m,n'\rangle$ and use (wFB). Symmetrically for (tUndo). (ii) is straightforward and observed in [14] □

The next result, though proved in a straightforward way, is important.

Lemma 3. Suppose that the consistency relation R_δ is functional from M to N, that is, there is a total function $f\colon M\to N$ such that $mR_\delta n$ iff $m.f = n$. Then for any $m\in M$, $n\in N$, $[m,n\rangle = m.f$ and a lens $\lambda(\delta) = (f,\langle *,*])$ is defined. The diagonal axioms specified in Table 2 then become lens axiom as specified in Table on the right. Conversely, any lens λ determines a functional di-system $\delta(\lambda)$ as described above.

Corollary 4. Given a di-system δ with functional consistency relation, the lens $\lambda(\delta)$ is (very) well-behaved/undoable iff δ is (very) well-behaved/undoable. Similarly, given a lens λ, di-system $\delta(\lambda)$ is (very) well-behaved/undoable iff λ is such.

Condition of diagonal system δ	Corresponding lens $\lambda(\delta)$
Correctness	
$(\text{dmCorr})^{\rightarrow}$ / $(\text{dmCorr})^{\leftarrow}$	trivial/(Dom)
$(\text{rgCorr})^{\rightarrow}$ / $(\text{rgCorr})^{\leftarrow}$	trivial/(PutGet)
weak Invertibility	
(wFB) / (wBF)	(GetPut)/(PutGet)
history Ignorance	
$(\text{hIgno})^{\rightarrow}$ / $(\text{hIgno})^{\leftarrow}$	trivial/(PutPut)
(tUndo) / (sUndo)	(PutGet)/(Undo)
Hippocraticness	
$(\text{Hipp})^{\rightarrow}$/ $(\text{Hipp})^{\leftarrow}$	trivial/(GetPut)

Our results in this section are summarized as follows.

Theorem 5. Different synchronization models are inter-relatd as shown by the class diagram in Fig. 1.

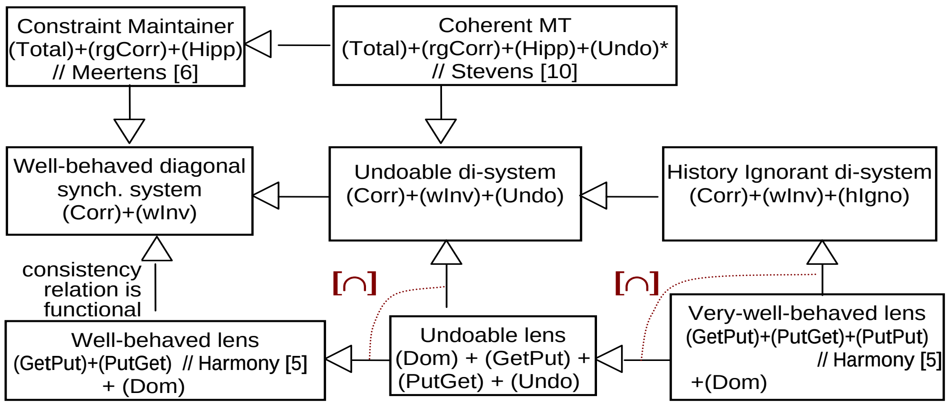


Fig. 1: Relationships between formal models. $\cap$ means intersection

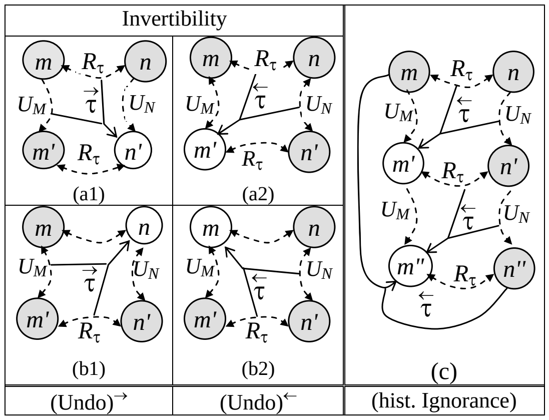
3.3 Incremental synchronization

Let (M, U_M) and (N, U_N) be two model spaces with updatability relations.

Definition 3. A *well-behaved trigonal synchronization system* over M, N (*tri-system* in short) is a triple $\tau = (R_\tau, \overrightarrow{\tau}, \overleftarrow{\tau})$ with $R_\tau \subset M \times N$ a binary *consistency relation*, and $\overrightarrow{\tau}\colon M \times M \times N \to N$ and $\overleftarrow{\tau}\colon N \times N \times M \to M$ are two partial functions called the *forward* and *backward synchronizers* resp., such that conditions (dmCorr), (rgCorr) and (Idnt) in Table 3 are satisfied.

To ease reading formulas, for arguments $m, m' \in M$, $n, n' \in N$, we write $[m, m', n\rangle$ and $\langle m, n', n]$ for $\overrightarrow{\tau}(m, m', n)$ and $\overleftarrow{\tau}(n, n', m)$.

A synchronization system is called *very well-behaved*/undoable if, in addition, conditions (hIgno)/(Undo) hold. If synchronizers are mutually inverse in the sense of conditions (Inv), the synchronization is called *invertible* – consider the pair of diagrams (a1)+(a2). Note that conditions (hIgno) and (Idnt) are tri-system counterparts of update translation compatibility with update composition and "do-nothing" updates (Sect.2.3.2)



Proposition 6. The following implications hold:
(i) (Hipp) implies (Idnt) *//straightforward*
(ii) (hIgno) and (Idnt) imply (Undo) *//assume* $m'' = m$ *in* $\overrightarrow{(hIgno)}$ *and* $n'' = n$ *in* $\overleftarrow{(hIgno)}$

Table 3: Conditions for trigonal synchronization systems. All formulas are universally quantified for all their variables ($\forall m, m', m'' \in M, \forall n, n', n'' \in N$). Expression $f(x) \Downarrow$ for a partial function f means that $f(x)$ is defined.

Condition: name and label	Forward increm. synch. $[*,*,*\rangle : M \times M \times N \to N$	Backward increm. synch. $M \leftarrow N \times N \times M : \langle *,*,*]$
Correctness: domain (dmCorr)	$[m, m', n\rangle \Downarrow$ iff $(mR_\tau n)\,\&\,(mU_M m')$	$\langle m, n', n] \Downarrow$ iff $(mR_\tau n)\,\&\,(nU_N n')$
range (rgCorr)	$\dfrac{\text{let } [m, m', n\rangle = n'}{\text{then } (m'R_\tau n')\,\&\,(nU_N n')}$	$\dfrac{\text{let } m' = \langle m, n', n]}{\text{then } (m'R_\tau n')\,\&\,(mU_M m')}$
Identity (Idnt)	$[m, m, n\rangle = n$	$\langle m, n, n] = m$
history Ignorance (hIgno)	$[m', m'', [m, m', n\rangle\rangle = [m, m'', n\rangle$	$\langle\langle m, n', n], n'', n'] = \langle m, n'', n]$
Undoability (Undo)	$[m', m, [m, m', n\rangle\rangle = n$	$m = \langle\langle m, n', n], n, n']$
Hippocraticness (Hipp)	$\dfrac{\text{if } m'R_\tau n}{\text{then } [m, m', n\rangle = n}$	$\dfrac{\text{if } mR_\tau n'}{\text{then } m = \langle m, n', n]}$
Invertibility (Inv): (BckFwd)	$[m, \langle m, n', n], n\rangle = n'$	
(FwdBck)	$m' = \langle m, [m, n, m'\rangle, n]$	

Example 2. Suppose that R_τ, U_M, U_N are always true, i.e., $R_\tau = M \times N$, $U_M = M \times M$, $U_N = N \times N$, and synchronizers do not actually react to updates: $[m, m', n\rangle = n$, $\langle m, n', n] = m$. It is very well-behaved but not invertible. It is trivially Hippocratic because it does nothing. Thus, in contrast to di-systems, in tri-synchronization Hippocraticness does not imply invertibility.

Example 3. Suppose that consistency relation is determined by a bijection $f\colon M \to N$ compatible with updatability: $mU_M m' \Rightarrow m.fU_N m'.f$. We define $R_f(m, n)$ iff $n = m.f$, $(m, m', n)\overrightarrow{\tau_f} = m'.f$ and $\overleftarrow{\tau_f}(n, n', m) = f^{-1}(n')$. Then the triple $\tau(f) = (R_f, \overrightarrow{\tau_f}, \overleftarrow{\tau_f})$ is very well-behaved (hence undoable) and invertible. It is also trivially Hippocratic because any update destroys consistency.

Example 4. Let $M = N = \mathbf{R} \times \mathbf{R}$ with $\mathbf{R}$ denoting the set of real numbers. Then model are pairs of numbers, which we will write as ab for M and xy for N. We define updatability and consistency relation by $U_M = M \times M$, $U_N = N \times N$ and $abRxy$ iff $a = x$. Synchronizers are defined as follows: $[ab, a'b', xy\rangle = (x + \Delta a)(y + \Delta b)$ where $\Delta a = a' - a$, $\Delta b = b' - b$, and $\langle ab, x'y', xy] = (a + \Delta x)(b + \Delta y)$ where $\Delta x = x' - x$, $\Delta y = y' - y$. It is easy to check that this system is very well-behaved and invertible yet it is not Hippocratic. If $\Delta a = 0$ but $\Delta b \neq 0$, we have $a'b'Rxy$ yet $[ab, a'b', xy\rangle = x(y + \Delta b) \neq xy$. The cause of non-Hippocraticness is that model synchronization uses all information embodied in the models while their consistency is based on only a part of it.

The examples demonstrate that (very) well-behavedness, invertibility and Hippocraticness are orthogonal concepts in trigonal synchronization. Particularly, non-Hippocraticness is quite natural for incremental systems. On the other hand, (Idnt) laws can be regarded as a weak form of Hippocraticness, and they are always assumed to hold.

Lemma 7. Suppose that for a correct trigonal system τ the consistency relation R_τ is functional from M to N: there is a total function $f\colon M \to N$ such that $mR_\tau n$ iff $m.f = n$. Then for any $n, n' \in N$, $nU_N n'$ iff both $n, n' \in f(M)$, and for any $m, m' \in M$, , $[m, m', n\rangle = m'.f$ and$\langle m, n', n] = \langle m, n', m.f]$. This defines a lens $\lambda(\tau)$ with $get_{\lambda(\tau)} = f$ and $put_{\lambda(\tau)}(n', m) = \langle m, n', m.f]$. The trigonal axioms specified in Table 3 then become lens axiom as specified in Table on the right. Conversely, any lens λ determines a functional tri-system $\tau(\lambda)$ as defined above.

Condition of trigonal system δ	Corresponding lens $\lambda(\tau)$
Correctness	
(dmCor)$^\rightarrow$/(dmCor)$^\leftarrow$	trivial/(Dom)
(rgCorr)$^\rightarrow$/(rgCorr)$^\leftarrow$	trivial/(PutGet)
Identity	
(Idnt)$^\rightarrow$/(Idnt)$^\leftarrow$	trivial/(GetPut)
history Ignorance	
(hIgno)$^\rightarrow$ / (hIgno)$^\leftarrow$	trivial/(PutPut)
(tUndo) / (sUndo)	(PutGet)/(Undo)
Invertibility	
(FwdBck) / (BckFwd)	(GetPut)/(PutGet)
Hippocraticness	
(Hipp)$^\rightarrow$/ (Hipp)$^\leftarrow$	trivial/(GetPut)

Corollary 8. Given a correct tri-system τ with functional consistency relation, the lens $\lambda(\tau)$ is well-behaved (and undoable) iff τ is well-behaved (and undoable). Similarly, given λ, tri-system $\tau(\lambda)$ is very well-behaved/undoable iff λ is such.

This result together with Corollary 4 shows that when consistency relation is functional, both di- and tri-synchronization systems degenerate to the same construct - a lens.

3.4 Summary

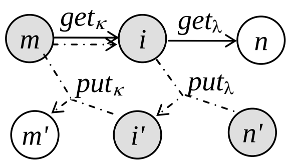
We have considered three families of formal models for model synchronization: lenses, di- and tri-systems, and analyzed four groups of algebraic laws representing four major synchronization concepts: (a) correctness (or well-behavedness), (b) Hippocraticness, (c) history ignorance (very well-behavedness) and undoability, and (d) invertibility. Among these structures, tri-systems present the most adequate and detailed model of incremental synchronization, and our analysis shows that they are also the most technically manageable: the four concepts are directly modeled in trigonal systems and, in fact, are mutually orthogonal within the trigonal framework. In contrast, for diagonal systems, history Ignorance, undoability and invertibility are modeled indirectly and there is some overlap between them. For lenses, the situation is even more complicated and the same pair of laws, (GetPut) and (PutGet) plays multiple roles (see tables in Lemmas 3,7).

4 Building synchronization systems from lenses

We consider three ways of lens composition: sequential one is known, the other two seem to be new (see discussion on p.1 for motivation).

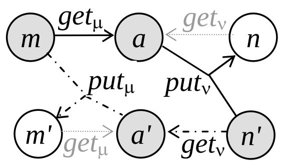
Construction 1 (Sequential composition of lenses)**.** Let $\kappa \in \boldsymbol{Lens}(M, I)$,

$\lambda \in \boldsymbol{Lens}(I, N)$ be two lenses. Their *sequential composition* $\kappa; \lambda \in \boldsymbol{Lens}(M, N)$ is defined as follows: for any $m \in M$, $n' \in N$, we set $m.get_{\kappa;\lambda} \overset{\text{def}}{=} m.get_\kappa.get_\lambda$ and $put_{\kappa;\lambda}(n', m) \overset{\text{def}}{=} put_\lambda(n', m.get_\kappa)$



Proposition 9. The composition $(\kappa; \lambda)$ is(very) well-behaved/undoable as soon as both lenses are such.

Construction 2 (*Co-targetial* composition of lenses)**.** Let μ be a lens from M to some space A and ν be a lens from N to the same space A. It makes A a space of common abstractions for M and N structures. Their *co-targetial composition* $\mu \bowtie \nu$ is a di-system over $M \times N$ defined as follows.



Consistency relation:: $R_{\mu\bowtie\nu}(m, n)$ iff $m.get_\mu = n.get_\nu$.
Forward synchronizer, $\overrightarrow{\tau_{\mu\bowtie\nu}}\colon M \times N \to N$::

$\overrightarrow{\tau_{\mu\bowtie\nu}}(m, n') \overset{\text{def}}{=} put_\nu(m.get_\mu, n')$.

Backward synchronizer, $\overleftarrow{\tau_{\mu\bowtie\nu}}\colon N \times M \to M$:: $\overleftarrow{\tau_{\mu\bowtie\nu}}(n', m) \overset{\text{def}}{=} put_\mu(n'.get_\nu, m)$.

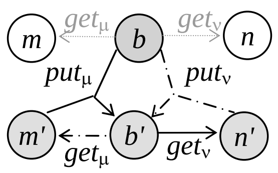
Proposition 10. The co-targetial composition $(\mu \bowtie \nu)$ of lenses is a (very) well-behaved/undoable diagonal system as soon as both lenses are such. In addition, the system is Hippocratic.

Construction 3 (*Co-sourcial* composition of lenses)**.** Let μ be a lens from some space B to M and ν be a lens from the same B to N. It makes M and N two different views to the same structure. Their *co-sourcial* composition $\mu \triangleleft\triangleright \nu$ is the following synchronization system. Consistency relation:: $R_{\mu\triangleleft\triangleright\nu}(m, n)$ iff there is $b \in B$ s.t.$m = b.get_\mu$, $n = b.get_\nu$.
Forward synchronizer, $\overrightarrow{\tau_{\mu\triangleleft\triangleright\nu}}\colon M \times B \to N$::

$\overrightarrow{\tau_{\mu\triangleleft\triangleright\nu}}(m', b) \overset{\text{def}}{=} (m', b)put_\mu.get_\nu$,

Backward synchronizer, $\overleftarrow{\tau_{\mu\triangleleft\triangleright\nu}}\colon N \times B \to M$::

$\overleftarrow{\tau_{\mu\triangleleft\triangleright\nu}}(n', b) \overset{\text{def}}{=} (n', b)put_\nu.get_\mu$, where we write *put* function symbols to the right of the arguments.



5 Conclusion

The paper presents an algebraic framework, in which semantics of different types of bi-directional model synchronization and transformation can be formally specified and analyzed. We have discussed semantics of bi-directional model synchronization in algebraic terms from the very first principles, including the notions of update and update translation, and the difference between selective and incremental synchronization as well. The goal was to find formal definitions and constructs that would be adequate to practice on one hand and technically manageable on the other. The four main synchronization concepts are distinguished (correctness, Hippocraticness, history ignorance/undoability and invertibility) and algebraically specified by equations. The behavior of these notions in different formalisms is also explored. Particularly, history ignorance and undoability appear to be very special requirements for selective synchronization but are natural for incremental synchronization while Hippocraticness, which seems quite natural for the former type of synchronization, is not necessary and may be too restrictive for the latter. It is also shown that trigonal synchronization structures present the most adequate and technically simple algebraic model of incremental synchronization, in which the four concepts are directly modeled and mutually independent.

References

[1] M. Antkiewicz and K. Czarnecki. Design space of heterogeneous synchronization. In *Generative and Transformational Techniques in Software Engineering*, 2008.

[2] F. Bancilhon and N. Spyratos. Update semantics of relational views. *TODS*, 6(4):557–575, 1981.

[3] G. Brunet, M. Chechik, S. Easterbrook, S. Nejati, N. Niu, and M. Sabetzadeh. A manifesto for model merging. In *GaMMa '06: Proc. Int. workshop on Global integrated model management.* ACM Press, 2006.

[4] U. Dayal and P. Bernstein. On the correct translation of update operations on relational views. *TODS*, 7(3):381–416, 1982.

[5] Z. Diskin and J. Diengel. A metamodel independent framework for model transformation: Towards generic model management patterns in reverse engineering. In J-M. Favre et al., editor, *3rd Int. Workshop on Metamodels, Schemas, Grammas and Ontologies for Reverse Engineering*, 2006.

[6] Z. Diskin, S. Easterbrook, and J. Dingel. Engineering associations: from models to code and back through semantics. In R. Paige and B. Meyer, editors, *TOOLS Europe 2008*, number 11 in LNBIP 11. Springer, 2008.

[7] H. Ehrig, K. Ehrig, C. Ermel, F. Hermann, and G. Taentzer. Information preserving bidirectional model transformations. In *FASE*, pages 72–86, 2007.

[8] J. Foster, M. Greenwald, J. Moore, B. Pierce, and A. Schmitt. Combinators for bi-directional tree transformations: a linguistic approach to the view update problem. In *POPL*, pages 233–246, 2005.

[9] G. Gottlob, P. Paolini, and R. Zicari. Properties and update semantics of consistent views. *ACM TODS*, 13(4):486–524, 1988.

[10] A. Königs and A. Schürr. Tool integration with triple graph grammars - A survey. *ENTCS*, 148(1):113–150, 2006.

[11] L. Meertens. Designing constraint maintainers for user interaction. available from http://www.kestrel.edu/home/people/meertens/, 1998.

[12] S. Mu, Z. Hu, and M. Takeichi. An algebraic approach to bi-directional updating. In W.-N. Chin, editor, *APLAS 2004*, volume 3302 of *LNCS*, 2004.

[13] OMG, `http://www.omg.org/docs/ptc`. MOF QVT Final Adopted Specification. Formal/05-11-01, 2005.

[14] P. Stevens. Bidirectional model transformation in QVT: Semantic issues and open questions. In *Models-07*. Springer LNCS#4735, 2007.