Electronic Communications of the EASST
Volume 57 (2013)

Proceedings of the Second International Workshop on Bidirectional Transformations (BX 2013)

Composing Least-change Lenses

Nuno Macedo, Hugo Pacheco, Alcino Cunha and José N. Oliveira

19 pages

Guest Editors: Perdita Stevens, James F. Terwilliger
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Composing Least-change Lenses

Nuno Macedo[1], Hugo Pacheco[2], Alcino Cunha[3] and José N. Oliveira[4]

[1] nfmmacedo@di.uminho.pt
[2] hpacheco@di.uminho.pt
[3] alcino@di.uminho.pt
[4] jno@di.uminho.pt
HASLab
INESC TEC / Universidade do Minho, Braga, Portugal

Abstract:

Non-trivial bidirectional transformations (BXs) are inherently ambiguous, as there are in general many different ways to consistently translate an update from one side to the other. Existing BX languages and frameworks typically satisfy fundamental first principles which ensure acceptable and stable (well-behaved) translation. Unfortunately, these give little insight about how a particular update translation is chosen among the myriad possible. From the user perspective, such unpredictability may hinder the adoption of BX frameworks.

The problem can be remedied by imposing a "principle of least change" which, in a state-based framework, amounts to translating each update in a way such that its result is as close as possible to the original state, according to some distance measure.

Starting by formalizing such BXs focusing on the particular framework of lenses, this paper discusses whether such *least-change lenses* can be defined by composition, an essential construct of BX frameworks. For sequential composition, two (dual) update translation alternatives are presented: a classical deterministic one and a nondeterministic. A key ingredient of the approach is the elegant formalization of the main concepts in relation algebra, which exposes several similarities and dualities.

Keywords: Bidirectional transformations, Principle of least change, Compositionality, Relational calculus

1 Introduction

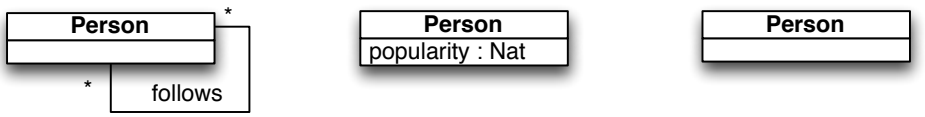
Consider the bidirectional transformation (BX) framework of *lenses* [FGM+07][1].

Definition 1 (Lens) A lens $l : S \trianglerighteq V$ between a source schema S and a view schema V consists of a total function $\mathsf{get}_l : S \to V$ and a partial function $\mathsf{put}_l : S \to V \nrightarrow S$. The lens is said to be well-behaved if it satisfies two BX laws:

$$s' \in \mathsf{put}_l\ s\ v \Rightarrow v = \mathsf{get}_l\ s' \qquad \textsc{PutGet}$$

$$v = \mathsf{get}_l\ s \Rightarrow s \in \mathsf{put}_l\ s\ v \qquad \textsc{GetPut}$$

[1] Note that we see lenses as a general bidirectional framework, including the particular lens language of [FGM+07].



(a) Model $\mathscr{M}_1$. (b) Model $\mathscr{M}_2$. (c) Model $\mathscr{M}_3$.

Figure 1: Model ($\mathscr{M}_1$) and respective views ($\mathscr{M}_2, \mathscr{M}_3$) of a simplified Twitter.

Here $s' \in \mathsf{put}_l\ s\ v$ means that $\mathsf{put}_l\ s\ v$ is defined and that $s' = \mathsf{put}_l\ s\ v$. The PUTGET law entails that a lens is *acceptable*, i.e., view updates are translated exactly by put (information in the view is not ignored). Essentially, this law imposes an upper bound on the behavior of put (admissible behaviors). The GETPUT law entails that a lens is *stable*, i.e., if the view is not changed, then it must be "put back" to the same source. Dually, this law imposes a lower bound on the behavior of put (mandatory behaviors).

The classic lens framework is compositional: writing a BX amounts to writing the get transformation in a domain-specific language that includes a set of primitive BXs, and combinators that allow us to compose those into complex "correct-by-construction" BXs. This contrasts with other "bidirectionalization" frameworks, that perform syntactic or semantic inversion of a transformation written in a general purpose language, or that derive suitable forward and backward transformations from a declarative consistency relation specified between source and target values.

In general, the above BX laws allow multiple valid backward transformations. For instance, consider the class diagram $\mathscr{M}_1$ in Figure 1a that models a simplified Twitter: we have people (*Person*) who may *follow* other people. We could define a model transformation T_1 that keeps the popularity of each person and erases the *follows* associations, resulting in the model $\mathscr{M}_2$ of Figure 1b, with the constraint that popularity cannot be higher than the number of people. A second model transformation T_2 could also remove the popularity of each person, ending up with $\mathscr{M}_3$ of Figure 1c, where only people that have followers are represented. To satisfy GETPUT, a put for T_1 must return the same source when the view is not changed. If the view is changed, then it must satisfy PUTGET, which essentially only requires popularity preservation. For example, should somebody be added to a view with a given popularity, put has freedom to determine who should follow her or him. Unfortunately, it is free to rearrange the followers of all other people too, as long as their popularity remains the same.

The lens laws should be taken as first principles: for example, GETPUT "only provides a relatively loose constraint on the behavior of lenses", as originally remarked by the authors of the lens framework [Fos09]. To understand the behavior of a lens, a user cannot rely solely on the laws and must directly inspect the definition of put. Therefore, laws providing more guarantees about put would provide a better account of its behavior to users: instead of requiring only that the source remains unchanged when the view is unchanged, they could require also that an update translation should be as small as possible.

This "principle of least change" has in fact already been proposed by Meertens [Mee98] for the BX framework of *constraint maintainers*. When applied to lenses, this principle will allow us to tighten the bounds imposed by the traditional laws, thus making the behavior of put more

predictable. Returning to our example, and assuming that the "distance" between two source instances increases with the number of people and *follow* associations not shared by both, the behavior of a put function that when adding a person to the target rearranges the followers of the remaining persons would no longer be *acceptable* (but different puts can still be defined, reflecting the different choices of followers to be added to the new person).

In this paper, we introduce *least-change lenses* by applying Meertens' [Mee98] "principle of least change" to regular lenses. Given a preorder (a reflexive and transitive binary relation) on sources that captures the distance to a given source, least-change lenses will require put to return minimal updates according to that preorder. We also discuss whether a compositional approach is viable to build such lenses. Indeed, Meertens did not investigate the composition of least-change maintainers and many authors [Ste10, Dis08, HPW11] view the lack of compositionality as a drawback.

Our fundamental (and novel) research question is: *given two least-change lenses, is their sequential composition also a least-change lens?* Suppose, for instance, that T_1 is a least-change lens minimizing updates according to the distance presented above. Can it be safely composed with another least-change lens? In part, the answer depends on the distance measure used for the target model, and we will present some criteria involving this metric that enable us to answer positively the above question.

Unfortunately, the framework of deterministic lenses (where put returns at most one source) is too restrictive to allow composition in general. In our example, if the distance on target models ignores popularity, any transformation performed after our transformation (for example, throwing away the popularity field and leaving only people) is free to assign a very high popularity to a new person, which will obviously lead to a non-minimal update of the composed global transformation (T_2 after T_1). In such situations, some sort of compositionality can sometimes be achieved if we allow put to be nondeterministic and require it to generate all sources at the minimal distance. This will allow the put of the first transformation to consider the different alternatives, and maybe return the desired least-changed sources. This nondeterministic composition is not ideal: when composing two lenses f and g, put_f needs to search which intermediate candidate results of put_g will generate minimal sources according to its preorder. Nevertheless, it is still far better than trying to discover the least-changed sources of the composed transformation directly, since for reasonable orders the guidance from the inner transformations will greatly limit the search space (indeed, it allows the construction of a search tree by composition).

Paper structure. The next section will introduce *least-change lenses*, both deterministic and nondeterministic. In Section 3 we provide a brief account of the relational calculus, just enough for allowing us to elegantly formalize such lenses. In particular, it will become clear that the traditional lens laws just impose bounds on the behavior of put, which will be tightened by least-change lenses. It will also expose the two variants (deterministic and nondeterministic) as duals of each other, paving the way to easily transcribing properties found about one of them to the other. Section 4 then presents some necessary conditions to attain (sequential) compositionality. Section 5 presents some related work, and Section 6 concludes and discusses several possible directions for future work on this topic, about which this paper aims only to make a small but principled contribution.

2 Least-change lenses

The concept of least changes requires the definition of a family of total preorders comparing sources relative to their "distance" to some other source[2]. For any source value $s \in S$ we also require the respective total preorder on S, denoted by $\preceq_s$, to be *stable*.

Definition 2 (Stable preorder) For any source $s \in S$, a preorder $\preceq_s$ is said to be *stable* if s is its unique minimal value, i.e.,

$$s_1 \preceq_s s \Leftrightarrow s_1 = s \qquad \textsc{OrdStable}$$

Condition OrdStable ensures that the source s is the single closest value to itself, i.e., s is the universal lower-bound of $\preceq_s$. One possible way to construct such total preorder $\preceq_s$ is to rely on a distance measure function $\mathsf{dist} : S \to S \to \mathbb{N}$ on sources S, such that [Mee98]

$$s_1 \preceq_s s_2 \Leftrightarrow (\mathsf{dist}\ s\ s_1) \leqslant (\mathsf{dist}\ s\ s_2)$$

where $\leqslant : \mathbb{N} \to \mathbb{N}$ is the standard order on natural numbers[3]. The order is stable if the distance is stable, i.e., the closest value to a source s is itself[4]:

$$\mathsf{dist}\ s\ s' = 0 \Leftrightarrow s = s' \qquad \textsc{DistStable}$$

An instance of $\mathscr{M}_1$ of our example consists of a tuple (W, F), where W is a set of people and F a *follows* relation from people to people (a set of pairs $F \subseteq W \times W$). A suitable distance measure for $\mathscr{M}_1$ is the size of their symmetric differences:

$$\mathsf{dist}_1\ (W,F)\ (W',F') = \#(W \,\Delta\, W') + \#(F \,\Delta\, F')$$
$$\quad \textbf{where}\ X \,\Delta\, Y = (X - Y) \cup (Y - X)$$

An instance of $\mathscr{M}_2$ can be seen as a tuple (W, P), where W is again a set of people and P is a function from people to popularity $P : W \to \mathbb{N}$, which is equivalent to a multi-set of people. Again, a suitable distance measure could be the size of their symmetric differences (generalized to multi-sets with the obvious definition):

$$\mathsf{dist}_2\ (W,P)\ (W',P') = \#(W \,\Delta\, W') + \#(P \,\Delta\, P')$$

We could however ignore the popularity of people not contained in both instances:

$$\mathsf{dist}'_2\ (W,P)\ (W',P') = \#(W \,\Delta\, W') + \#((P \otimes (W \cap W')) \,\Delta\, (P' \otimes (W \cap W')))$$
$$\quad \textbf{where}\ (X \otimes Y)\ x = \textbf{if}\ x \in Y\ \textbf{then}\ X\ x\ \textbf{else}\ 0$$

[2] By *total* preorder we mean a preorder $\preceq$ such that, for any points x, y, either $x \preceq y$ or $y \preceq x$ holds. Anti-symmetry is not required because any two points at the same distance will be comparable in either way and they need not be the same.

[3] Instead of natural numbers any well ordered set could be used in the range of the distance function, providing more flexibility when computing distances. Notice also that if, for a given s, $\mathsf{dist}\ s$ is an injective distance function, then the induced preorder $\preceq$ will be anti-symmetric (i.e. a partial order).

[4] In a metric space, this clause is known as *identity of indiscernibles*. In fact, it could be reasonable to assume that (S, dist) is a metric space, satisfying other properties such as *symmetry* or *triangle inequality*, although these are orthogonal to the properties of least-change lenses.

2.1 Deterministic least-change lenses

Definition 3 (Least-change lens) Given a family of stable total preorders $\preceq_s$ for every $s \in S$, a *least-change* lens $l : S_{\preceq} \trianglerighteq V$ consists of a forward total transformation $\mathsf{get}_l : S \to V$, and a partial backward transformation $\mathsf{put}_l : S \to V \nrightarrow S$. A least-change lens is well-behaved if it satisfies GETPUT and

$$s' \in \mathsf{put}_l\ s\ v \Rightarrow v = \mathsf{get}_l\ s' \wedge (\forall\, s_1\ .\ v = \mathsf{get}_l\ s_1 \Rightarrow s' \preceq_s s_1) \qquad \text{LEASTPUTGET}$$

LEASTPUTGET is equivalent to the original least change principle proposed in [Mee98]. It is a refinement of PUTGET, in the sense that the resulting source value is required to be not only acceptable, but also one of closest to the original s among the sources that share the same view v. LEASTPUTGET thus lowers the upper bound for put and reduces the range of valid updates.

Although the backward transformation is more restricted, there may still be more than one valid put for the same get and preorder $\preceq_s$, since there may be multiple sources at the same distance. Consider our example transformation T_1 with the preorder induced by the distance measure dist_1. If we add a person p to the view with popularity n, put is free to choose who will be her followers, since any instance with n followers for p is at the same distance from the original source. However, it is no longer free to change the followers of other people, since this would result in more distant instances. Concerning transformation T_2, if we use the preorder induced by dist_2, then when a new person is added to the view put must assign him popularity 0. However, if we use the preorder induced by dist'_2 instead, put is free to assign the new person any valid popularity. In both cases, put must preserve the popularity of the previously existing people.

Obviously, the preorder may not discriminate much, and at one end of the spectrum we may have a preorder induced by the measure that considers all different values at the same distance[5]:

$$\mathsf{dist}\ s\ s' = \textbf{if}\ s = s'\ \textbf{then}\ 0\ \textbf{else}\ 1$$

In this case, LEASTPUTGET degenerates into the original PUTGET, allowing the same backward transformations as the regular lens laws, only recovering the original source when the view is not modified. The other extreme case occurs when the preorder is refined to the point where minimal values are unique and there is a single valid put. This will occur when dist s is injective (resulting in a total order $\preceq_s$), since the minimal source will always be unique.

2.2 Nondeterministic least-change lenses

As discussed in the introduction, for many pragmatic examples a deterministic put that commits to a particular minimal update will be too restrictive to allow compositionality. Therefore, we will introduce a variant of least-change lenses where put is nondeterministic, and enumerates all possible minimal updates. To distinguish it from the deterministic case it will be denoted as Put.

Definition 4 (Nondeterministic least-change lens) Given a family of stable total preorders $\preceq_s$ for every $s \in S$, a *nondeterministic least-change lens* $l : S_{\preceq} \blacktriangleright V$ consists of a forward total

[5] In metric space terminology, this is known as the *discrete metric* on a set.

transformation $\mathsf{get}_l : S \to V$, and a nondeterministic backward transformation $\mathsf{Put}_l : S \to V \to \mathscr{P}(S)$ The lens is said to be well-behaved if it satisfies PUTGET and

$$v = \mathsf{get}_l\ s' \wedge (\forall\ s_1\ .\ v = \mathsf{get}_l\ s_1 \Rightarrow s' \preceq_s s_1) \Rightarrow s' \in \mathsf{Put}_l\ s\ v \qquad \text{LEASTGETPUT}$$

In nondeterministic least-change lenses, the LEASTGETPUT law replaces GETPUT as the lower bound of Put and PUTGET is kept as the upper bound. In fact, LEASTGETPUT is the dual of LEASTPUTGET, as it states that, if a source s' (with view v) is one of the closest to s, then it must be returned by $\mathsf{Put}\ s\ v$. This duality will become clear in the next section.

Returning to our transformation T_2 with distance measure dist'_2, when a new person is added to the view, a well-behaved Put will return all instances with all possible popularities for this person.

3 Formalizing least-change lenses in relational algebra

In the sequel we use relational algebra [BM97] to formally specify and reason about least-change lenses. Each set-valued function $f : A \to \mathscr{P}(B)$ is in 1-to-1 correspondence with a relation $R \subseteq B \times A$ in the sense that $b \in f\ a$ is equivalent to $(b,a) \in R$, or $b\ R\ a$ using infix notation, as in $2 \leqslant 3$. Infix notation suggests writing $R : B \leftarrow A$ (or $R : A \to B$) instead of $R \subseteq B \times A$ as the *type* of relation R, where the arrow captures the flow of information. This makes it easy to type the main operation of relation algebra, sequential *composition*, as arrow *chaining*: given relations $R : A \to B$ and $S : B \to C$, their composition, denoted by $S \cdot R : A \to C$, such that $c\ (S \cdot R)\ a$ means that there is a b such that $c\ S\ b$ and $b\ R\ a$. Arrow notation is also meaningful in typing the *converse* of a relation $R : A \to B$, $R^\circ : B \to A$, which is such that $a\ R^\circ\ b$ means the same as $b\ R\ a$. A function $f : A \to B$ is a special case of relation, in that it is deterministic and totally defined. Thus both $b\ f\ a$ and $a\ f^\circ\ b$ mean $b = f\ a$. For instance, expression $f^\circ \cdot R \cdot g$ denotes a preorder if R is another preorder. It compares a and a' by comparing their images R, $(f\ a)\ R\ (g\ a')$.

The relational calculus enables a *point-free* (or quantifier-free) formalization of first-order logic, where atomic formulas are inequations $R \subseteq S$, meaning $b\ R\ a \Rightarrow b\ S\ a$ for all a and b. To obtain the expressive power of first-order logic we also need the *split* (or *fork*) of relations $R : A \to B$ and $S : A \to C$, denoted by $R \vartriangle S : A \to B \times C$, $(b,c)\ (R \vartriangle S)\ a$ holding whenever $b\ R\ a$ and $c\ S\ a$ hold. The notation $\underline{a} : B \to A$ is used to denote the constant function that returns the value $a \in A$ for every input. Finally, we also use the *division* combinator between relations $R : C \to A$ and $S : C \to B$ denoted $S \,/\, R : A \to B$ (with $b\ (S \,/\, R)\ a$ holding whenever $a\ R\ c$ implies $b\ S\ c$, for any possible c). The domain and range of a relation $R : A \to B$ will be denoted by $\delta R : A \to A$ ($a\ \delta R\ a \Leftrightarrow \exists\ b\ .\ bRa$) and $\rho R : B \to B$ ($b\ \rho R\ b \Leftrightarrow \exists\ a\ .\ bRa$) respectively. These belong to a special kind of relation called *coreflexive* which, being subsets of the identity, act as a kind of filter.

It can be easily checked, for instance, that inequality $\mathsf{put}\ s \subseteq \mathsf{get}^\circ$ means the same as the PUTGET law: $s' \in \mathsf{put}\ s\ v \Rightarrow v = \mathsf{get}\ s'$. Similar re-statement of PUTGET and GETPUT in relational algebra makes it clear that they establish lower and upper bounds for put,

$$\mathsf{put}\ s \subseteq \mathsf{get}^\circ \qquad \text{PUTGET}$$

$$\rho \underline{s} \cdot \mathsf{get}^\circ \subseteq \mathsf{put}\ s \qquad \text{GETPUT}$$

where $\rho \underline{s} = \{(s,s)\}$ filters out values other than s, restricting the output of get° to s. As $\rho \underline{s}$ is at most the identity, such bounds are consistent – the lower bound is smaller than the upper bound.

The principle of least change, formalized by the LEASTPUTGET and LEASTGETPUT laws, can be encoded in terms of the so-called *shrink* operator $R \upharpoonright S : A \to B$ [MO11] that minimizes the output of a relation $R : A \to B$ in regard to a relation $S : B \to B$, and defined as follows:

$$b\ (R \upharpoonright S)\ a \Leftrightarrow b\ R\ a \wedge (\forall\, x\ .\ b\ R\ x \Rightarrow a\ S\ x)$$

In relational notation this equals defining $R \upharpoonright S = R \cap S \,/\, R^\circ$. So $R \upharpoonright S$ is *at most* R and its output is a *minimum* in regard to S. Using *shrink*, the least-change laws can be specified as follows:

$$\mathsf{put}\ s \subseteq \mathsf{get}^\circ \upharpoonright \preceq_s \qquad \text{LEASTPUTGET}$$
$$\mathsf{get}^\circ \upharpoonright \preceq_s \subseteq \mathsf{Put}\ s \qquad \text{LEASTGETPUT}$$

Clearly, these laws are dual of each other. Moreover, it can be shown that they refine the original laws in the sense that LEASTPUTGET lowers the upper-bound imposed by PUTGET and LEASTGETPUT raises the lower-bound imposed by GETPUT.

Proposition 1 *Given a family of stable total preorders $\preceq_s : S \to S$, we have*

$$\rho \underline{s} \cdot \mathsf{get}^\circ \subseteq \mathsf{get}^\circ \upharpoonright \preceq_s \subseteq \mathsf{get}^\circ$$

Proof. The upper bound is trivial since $\mathsf{get}^\circ \cap \preceq_s / \mathsf{get} \subseteq \mathsf{get}^\circ$; for the lower bound we have

$$\begin{array}{ll}
& \rho \underline{s} \cdot \mathsf{get}^\circ \subseteq \mathsf{get}^\circ \upharpoonright \preceq_s \\
\equiv & \{\text{-shrink definition -}\} \\
& \rho \underline{s} \cdot \mathsf{get}^\circ \subseteq \mathsf{get}^\circ \cap \preceq_s / \mathsf{get} \\
\equiv & \{\text{-universal-}\cap\text{ ; division -}\} \\
& \rho \underline{s} \cdot \mathsf{get}^\circ \subseteq \mathsf{get}^\circ \wedge \rho \underline{s} \cdot \mathsf{get}^\circ \cdot \mathsf{get} \subseteq \preceq_s \\
\equiv & \{\text{-}\rho \underline{s} = \underline{s} \cdot \underline{s}^\circ \text{ ; shunting -}\} \\
& \underline{s}^\circ \cdot \mathsf{get}^\circ \cdot \mathsf{get} \subseteq \underline{s}^\circ \cdot \preceq_s \\
\equiv & \{\text{-ORDSTABLE thus } \underline{s}^\circ \cdot \preceq_s = \top \text{ -}\} \\
& \underline{s}^\circ \cdot \mathsf{get}^\circ \cdot \mathsf{get} \subseteq \top
\end{array}$$

□

Note how ORDSTABLE is required for this proof. This was expected, since the lower bound is only restricting the result of put for the original source s, and ORDSTABLE states precisely that it is a minimum of the preorder. This proposition also shows that LEASTGETPUT and PUTGET are consistent bounds.

Wherever dist s is injective ($\preceq_s$ is antisymmetric in this case) or get is injective, $\mathsf{get}^\circ \upharpoonright \preceq_s$ will be deterministic[6] and put s will also be deterministic. Moreover:

Proposition 2 *For* get *surjective and* $\mathsf{get}^\circ \upharpoonright \preceq_s$ *deterministic,* $\mathsf{put}\ s = \mathsf{get}^\circ \upharpoonright \preceq_s$ *is the only total backward transformation that gives rise to a well-behaved deterministic least-change lens. It is also the smallest* Put *that gives rise to a well-behaved nondeterministic least-change lens.*

[6] Since antisymmetric shrinking criteria ensure determinism [MO11].

Proof. Trivial, since $\preceq_s$ is reflexive and shrinking deterministic relations by reflexive orderings has no effect [MO11]. □

We can also prove the expected relationship between regular and least-change lenses.

Proposition 3 *Every well-behaved deterministic least-change lens $l : S_{\preceq} \trianglerighteq V$ is a well-behaved regular lens $l : S \trianglerighteq V$.*

Proof. GETPUT is a requirement for least-change lenses and PUTGET follows from Proposition 1. □

Proposition 4 *Every well-behaved regular lens $l : S \trianglerighteq V$ is a well-behaved deterministic least-change lens $l : S_{\preceq} \trianglerighteq V$.*

Proof. Take the total preorder induced by $\mathsf{dist}\ s\ s' = \textbf{if}\ s = s'\ \textbf{then}\ 0\ \textbf{else}\ 1$. Then LEASTPUTGET degenerates into PUTGET. □

Finally, any well-behaved lens can also be seen as a well-behaved nondeterministic least-change lens by inferring a preorder such that $\mathsf{get}^{\circ} \upharpoonright \preceq_s$ is deterministic and equal to put.

Proposition 5 *Every well-behaved regular lens $l : S \trianglerighteq V$ is a well-behaved nondeterministic least-change lens $l : S_{\preceq} \blacktriangleright V$.*

Proof. For the total preorder

$$s'' \preceq_s s' \Leftrightarrow (s' \in \mathsf{Put}_l\ s\ (\mathsf{get}_l\ s') \Rightarrow s'' \in \mathsf{Put}_l\ s\ (\mathsf{get}_l\ s''))$$

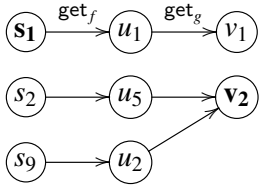
$\mathsf{get}_l^{\circ} \upharpoonright \preceq_s$ is deterministic and equal to the only possible put. LEASTGETPUT follows easily. □

4 Criteria for composing least-change lenses

Given two well-behaved least-change lenses $f : S_{\preceq} \trianglerighteq U$ and $g : U_{\sqsubseteq} \trianglerighteq V$, we will now discuss when can they be composed in order to obtain a well-behaved least-change lens $g \bullet f : S_{\preceq} \trianglerighteq V$ using the lens composition combinator $(\bullet)$ [FGM+07]:

$$\mathsf{get}_{g \bullet f}\ s = (\mathsf{get}_g \bullet \mathsf{get}_f)\ s$$
$$\mathsf{put}_{g \bullet f}\ s = (\mathsf{put}_f\ s) \bullet (\mathsf{put}_g\ (\mathsf{get}_f\ s))$$

In general this is not true, because $\mathsf{put}_{g \bullet f}$ is not guaranteed to return minimal values. For instance, consider the following example depicting the forward transformations get_f and get_g, and where $\mathsf{dist}\ x_i\ x_j = |j - i|$ for both lenses:



Consider the backward transformation of the view v_2 over the original source s_1 of the $g \bullet f$ lens: first, $\mathsf{put}_g\ v_2\ (\mathsf{get}_f\ s_1) = u_2$, since u_2 is closer to u_1 (the original intermediate U-view) than u_5; then, $\mathsf{put}_f\ s_1\ u_2 = s_9$, since it is the only choice for u_2. However, this is not the minimal result for the least-change lens $g \bullet f$, since the source closest to s_1 with view v_2 is s_2 rather than s_9. Such issues occur because the U-view generated by the concrete put_g selected from the range of valid ones may not result in a minimum S-view when passed through put_f.

We will see that in order to preserve LeastPutGet, the forward transformations will need to somehow preserve the preorders. To be more precise, the goal of this section is to analyze under which conditions LeastPutGet is preserved by composition, i.e.:

$$\forall\, s\,.\ \mathsf{put}_f\ s\ \subseteq\ \mathsf{get}_f^{\circ} \restriction \preceq_s \wedge \forall\, u\,.\ \mathsf{put}_g\ u\ \subseteq\ \mathsf{get}_g^{\circ} \restriction \sqsubseteq_u \Rightarrow \forall\, s\,.\ \mathsf{put}_{g \bullet f}\ s\ \subseteq\ (\mathsf{get}_g \bullet \mathsf{get}_f)^{\circ} \restriction \preceq_s$$

The simplest condition for this to hold is for get_g to be injective: since get_g° is deterministic, there is a unique valid put_g, and thus there is no ambiguity in the choice.

Proposition 6 *If $f : S_{\preceq} \trianglerighteq U$ and $g : U_{\sqsubseteq} \trianglerighteq V$ are well-behaved least-change lenses and get_g is injective, then $g \bullet f : S \trianglerighteq_{\preceq} V$ is a well-behaved least-change lens.*

However, as the above counter-example also shows, having an injective get_f is not enough to preserve LeastPutGet. Still, a sufficient condition on get_f is for it to be strictly increasing between the preorders $\preceq_s$ and $\sqsubseteq_{\mathsf{get}_f\ s}$:

$$s_1 \prec_s s_2 \Rightarrow (\mathsf{get}_f\ s_1) \sqsubset_{\mathsf{get}_f\ s} (\mathsf{get}_f\ s_2)$$

A strictly increasing get_f reads: if a source s_1 is smaller than another source s_2 (in distance to the original source s), then its view $\mathsf{get}_f\ s_1$ shall be smaller than the view of s_2 (in distance to the original view $\mathsf{get}_f\ s$). In point-free notation this can be expressed as follows:

$$\mathsf{get}_f \bullet \prec_s\ \subseteq\ \sqsubset_{\mathsf{get}_f\ s} \bullet \mathsf{get}_f \qquad \textsc{StrictInc}$$

In total preorders, we have $s_1 \prec_s s_2 \Leftrightarrow \neg\,(s_2 \preceq_s s_1)$. As such, an equivalent formulation is:

$$(\mathsf{get}_f\ s_1) \sqsubseteq_{\mathsf{get}_f\ s} (\mathsf{get}_f\ s_2) \Rightarrow s_1 \preceq_s s_2$$

Proposition 7 *If $f : S_{\preceq} \trianglerighteq U$ and $g : U_{\sqsubseteq} \trianglerighteq V$ are well-behaved least-change lenses and get_f is strictly increasing between $\preceq_s$ and $\sqsubseteq_{\mathsf{get}_f\ s}$ for every $s \in S$, then $g \bullet f : S \trianglerighteq_{\preceq} V$ is a well-behaved least-change lens.*

Unfortunately this requirement is too restrictive, as it forces get_f to be injective, meaning that it cannot abstract information from the source. To help understand why composition succeeds when get_f is strictly increasing, Figure 2 shows a possible configuration under this assumption. On the left subfigure, several values of type U spread over concentric circles according to their distance to a hypothetical center denoting $\mathsf{get}_f\ s$ (and represented by a star). Similarly for the right subfigure, for values of type S and center s. x^{-1} denotes the pre-image of x under get_f. In this case, we have a one-to-one correspondence since get_f, being strictly increasing, must be injective. Another consequence of this property is that all values of U that are at the same distance from

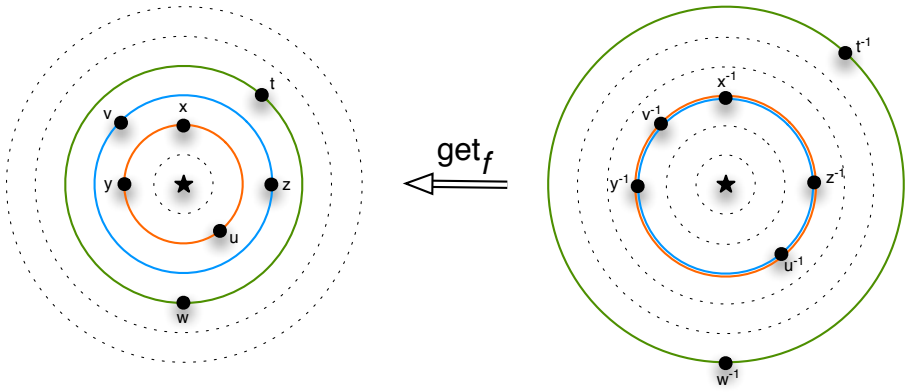


Figure 2: Strictly increasing transformation

$\mathsf{get}_f\ s$ must have their pre-images at the same distance from s. Moreover, the relative distances to the center are preserved, in the sense that, for example, since $y \sqsubseteq_{\mathsf{get}_f\ s} z$ then $y^{-1} \preceq_s z^{-1}$. Now, consider the backward transformation, when putting back a given target in the original source s. Being well-behaved, put_g will return one of the values that is closest to $\mathsf{get}_f\ s$, in this case either x, y or u; put_f must return one of their pre-images which, due to the above assumptions, is guaranteed to be one of the closest to s, whatever choice is made by put_g.

While a strictly increasing get_f suffices to guarantee that any composition $g \cdot f$ is also a well-behaved least-change lens, in order to attain a proper compositional language and be able to compose any sequence of transformations, this property must also be preserved by composition.

Proposition 8 *If two transformations* $\mathsf{get}_f : S_\preceq \to U$ *and* $\mathsf{get}_g : U_\sqsubseteq \to V$ *are strictly increasing, then* $\mathsf{get}_g \cdot \mathsf{get}_f : S_\preceq \to V$ *is also strictly increasing.*

A more interesting requirement is the following more relaxed version of StrictInc, that does not imply injectivity:

$$(\forall\ s_3\ .\ \mathsf{get}_f\ s_2 = \mathsf{get}_f\ s_3 \Rightarrow s_1 \prec_s s_3) \Rightarrow (\mathsf{get}_f\ s_1) \sqsubset_{\mathsf{get}_f\ s} (\mathsf{get}_f\ s_2)$$

In point-free notation this is written

$$(\prec_s / \mathsf{get}_f) \cdot \rho\mathsf{get}_f \subseteq \mathsf{get}_f^\circ \cdot \sqsubset_{\mathsf{get}_f\ s} \qquad \text{QuasiStrictInc}$$

Now a view $\mathsf{get}_f\ s_1$ is required to be smaller than another view $\mathsf{get}_f\ s_2$ only when s_1 is smaller than all sources that map to the view of s_2. An alternative formulation using the less than or equal orders is

$$(\mathsf{get}_f\ s_1) \sqsubseteq_{\mathsf{get}_f\ s} (\mathsf{get}_f\ s_2) \Rightarrow (\exists\ s_3\ .\ \mathsf{get}_f\ s_1 = \mathsf{get}_f\ s_3 \wedge s_3 \preceq_s s_2)$$

or $\rho\mathsf{get}_f \cdot \sqsubseteq_{\mathsf{get}_f\ s} \cdot \mathsf{get}_f \subseteq \mathsf{get}_f \cdot \preceq_s$ in the point-free notation. In other words, when a view v_1 is smaller than or equal to another view v_2 (in distance to a view $\mathsf{get}_f\ s$), then at least one source with view v_1 is smaller than or equal to all the sources with view v_2 (in distance to a source s). The

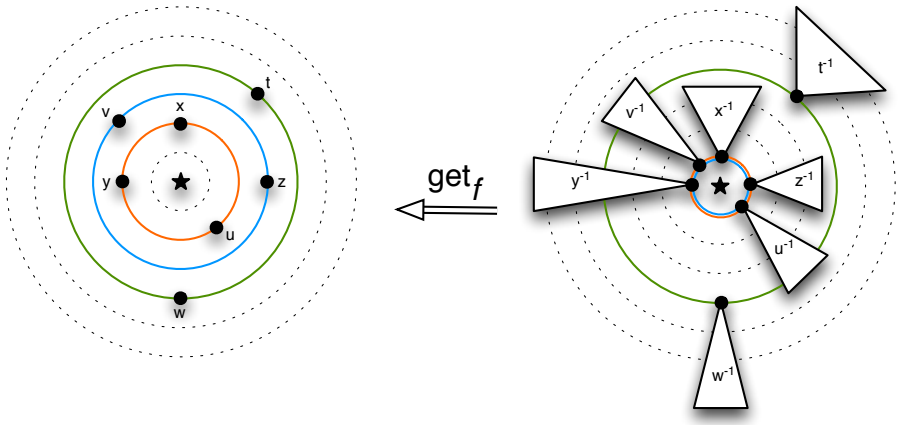


Figure 3: Quasi strictly increasing transformation

difference to StrictInc can easily be seen in Figure 3. Now the pre-image of a target value can contain several sources at different distances from the center. Still, for targets at the same distance to $\mathsf{get}_f\ s$, the minimum values of their pre-images must all be at the same distance from s.

Proposition 9 *If $f : S_{\preceq} \trianglerighteq U$ and $g : U_{\sqsubseteq} \trianglerighteq V$ are well-behaved least-change lenses and get_f is quasi strictly increasing between $\preceq_s$ and $\sqsubseteq_{\mathsf{get}_f\ s}$ for every $s \in S$, then $g \cdot f : S \trianglerighteq_{\preceq} V$ is a well-behaved least-change lens.*

Figure 3 also helps understanding why this proposition holds. Again, put_g is free to choose any value closest to $\mathsf{get}_f\ s$ but, whatever the choice made, put_g will always return one of the value closest to s. An example of a transformation that is not strictly increasing but is quasi strictly increasing is determining the size of a set, with the obvious distance measures (size of symmetric difference in the source and absolute value of the difference in the target).

Unlike with strictly increasing transformations, the composition of two quasi strictly increasing transformations only remains so if get_f is surjective.

Proposition 10 *If two transformations $\mathsf{get}_f : S_{\preceq} \to U$ and $\mathsf{get}_g : U_{\sqsubseteq} \to V$ are quasi strictly increasing and get_f is surjective, then $\mathsf{get}_g \cdot \mathsf{get}_f : S_{\preceq} \to V$ is also quasi strictly increasing.*

In a lens framework requiring surjectivity is not a big drawback, since it is usually already required due to totality constraints [FGM+07, PC10].

An interesting class of lenses is that obtained from transformations with a *perfect complement*. A function cpl is said to be a complement of get if get $\triangle$ cpl is injective, i.e., any source can be losslessly represented as a view-complement pair. For view update translation under a constant complement, there is a unique source update for each view update [BS81]. A complement is said to be *independent* [KU84] when get $\triangle$ cpl is bijective, i.e., any pair corresponds to a source state. If a get has an independent complement (we call it *perfect*), then there exists a lens with a unique optimal put with perfect updatability, since it is possible to translate all view updates onto source updates while keeping the constant complement. Typical examples of transformations that fall

under this category are tuple projections, whose perfect complement is exactly the information projected out.

Having a perfect complement is not sufficient for a least-change lens to be composable. However, if get is strictly increasing among sources with the same complement, then it is also quasi strictly increasing, and thus can safely be composed.

Proposition 11 *If* $\mathsf{get}_f : S_\preceq \to U_\sqsubseteq$ *has perfect complement and* $(\mathsf{get}\ s_1) \sqsubseteq_{\mathsf{get}\ s} (\mathsf{get}\ s_2) \wedge \mathsf{cpl}\ s_1 = \mathsf{cpl}\ s_2 \Rightarrow s_1 \preceq_s s_2$ *for every* $s, s_1, s_2 \in S$*, then* get_f *is quasi strictly increasing.*

Proving that a transformation is quasi strictly increasing is not trivial. This is an example of a useful class of transformations where such proof can be much simplified.

Dually to the deterministic case, the necessary condition for two nondeterministic least-change lenses to be composable is the following:

$$\forall\, s\,.\, \mathsf{get}_f^\circ \upharpoonright \preceq_s \ \subseteq\ \mathsf{Put}_f\ s \wedge \forall\, u\,.\, \mathsf{get}_g^\circ \upharpoonright \preceq_u \ \subseteq\ \mathsf{Put}_g\ u \Rightarrow \forall\, s\,.\, (\mathsf{get}_g \cdot \mathsf{get}_f)^\circ \upharpoonright \preceq_s \ \subseteq\ \mathsf{Put}_{g \cdot f}\ s$$

Once again, an injective get_g guarantees the compositionality.

Proposition 12 *If* $f : S_\preceq \blacktriangleright U$ *and* $g : U_\sqsubseteq \blacktriangleright V$ *are well-behaved nondeterministic least-change lenses and* get_g *is injective, then* $g \cdot f : S_\preceq \blacktriangleright V$ *is a well-behaved nondeterministic least-change lens.*

Interestingly, the remaining laws that ensure nondeterministic compositionality are dual to those for the deterministic case: we can either reverse every law from $R \subseteq S$ to $S \subseteq R$ or, equivalently, replace every $\prec_s$ by $\preceq_s$. For example, instead of requiring get_f to be strictly increasing it suffices to require it to be monotonic:

$$s_1 \preceq_s s_2 \Rightarrow (\mathsf{get}_f\ s_1) \sqsubseteq_{\mathsf{get}_f\ s} (\mathsf{get}_f\ s_2)$$

Remember that monotonicity reads the same way as the strictly increasing property, by simply replacing "smaller than" by "smaller than or equal to" in the text. An equivalent formulation in the opposite direction is:

$$(\mathsf{get}_f\ s_1) \sqsubset_{\mathsf{get}_f\ s} (\mathsf{get}_f\ s_2) \Rightarrow s_1 \prec_s s_2$$

In the point-free notation we have

$$\mathsf{get}_f \cdot \preceq_s \ \subseteq\ \sqsubseteq_{\mathsf{get}_f\ s} \cdot \mathsf{get}_f \qquad \textsc{Monot}$$

Proposition 13 *If* $f : S_\preceq \blacktriangleright U$ *and* $g : U_\sqsubseteq \blacktriangleright V$ *are well-behaved nondeterministic least-change lenses and* get_f *is surjective and monotonic between* $\preceq_s$ *and* $\sqsubseteq_{\mathsf{get}_f\ s}$ *for every* $s \in S$*, then* $g \cdot f : S_\preceq \blacktriangleright V$ *is a well-behaved nondeterministic least-change lens.*

Unlike the strictly increasing property, monotonicity no longer implies that get_f is injective. Unlike in Proposition 7, here we require get_f to be surjective: note that surjectivity is the dual property of being deterministic, which get_f always satisfies. Figure 4 helps understanding the differences between being strictly increasing and monotonic, and why this property ensures a

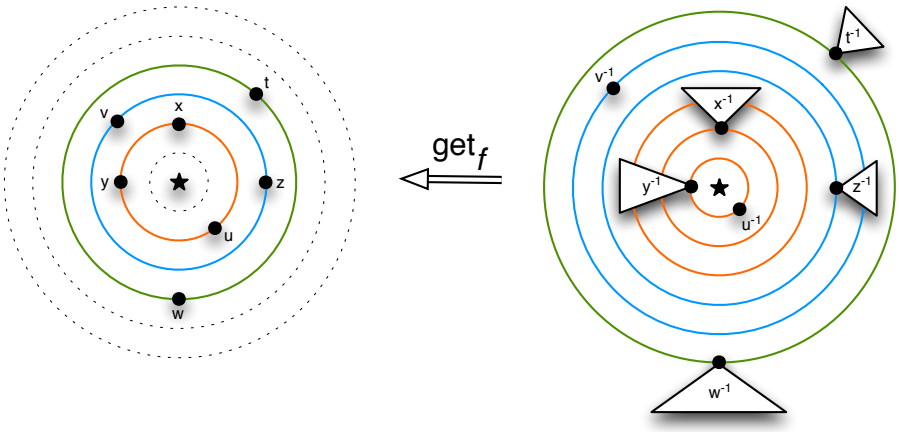


Figure 4: Monotonic transformation

well-behaved nondeterministic composition. Unlike in Figure 2, the pre-images of the values of type U at the same distance from $\mathsf{get}_f\ s$ are allowed to be at different distances from s. However, we have the restriction that, for example, when $y \sqsubset_{\mathsf{get}_f\ s} z$ then $y^{-1} \prec_s z^{-1}$, forcing the pre-images of targets at different distances from $\mathsf{get}_f\ s$ to also be at different distances from s (respecting the relative orders). This ensures that when put_g returns all values at the minimum distance from $\mathsf{get}_f\ s$, all sources at the minimum distance from s are contained in their pre-image.

An interesting fact is that, unlike in total partial-orders, a strictly increasing transformation is not necessarily monotonic. This means that we can can have transformations whose composition is well-behaved in the deterministic scenario, but not well-behaved in the nondeterministic one. Figure 2 helps understanding why: even if put_f returns all values closest to $\mathsf{get}_f\ s$, there is no chance for put_f to return all values closest to s, since some of the pre-images closest to s originate from targets further away from $\mathsf{get}_f\ s$ (namely, v and z).

As with strictly increasing transformations, monotonicity is preserved by composition.

Proposition 14 *If two transformations* $\mathsf{get}_f : S_{\preceq} \to U$ *and* $\mathsf{get}_g : U_{\sqsubseteq} \to V$ *are monotonic, then* $\mathsf{get}_g \bullet \mathsf{get}_f : S_{\preceq} \to V$ *is also monotonic.*

Again, monotonicity can be relaxed to

$$(\forall\ s_3\ .\ \mathsf{get}_f\ s_2 = \mathsf{get}_f\ s_3 \Rightarrow s_1 \preceq_s s_3) \Rightarrow (\mathsf{get}_f\ s_1) \sqsubseteq_{\mathsf{get}_f\ s} (\mathsf{get}_f\ s_2)$$

or with the point-free notation:

$$(\preceq_s / \mathsf{get}_f) \bullet \rho\mathsf{get}_f \subseteq \mathsf{get}_f^{\circ} \bullet \sqsubseteq_{\mathsf{get}_f\ s} \qquad \textsc{QuasiMonot}$$

Quasi monotonicity states that if a source s_1 is smaller than or equal to all the sources with view v (in distance to a source s), then its view $\mathsf{get}_f\ s_1$ shall be at most v (in distance to a view $\mathsf{get}_f\ s$). Alternatively, when a view v_1 is smaller than another view v_2 (in distance to a view $\mathsf{get}_f\ s$), then at least one source with view v_1 is smaller than all sources with view v_2 (in distance to a source s):

$$(\mathsf{get}_f\ s_1) \sqsubset_{\mathsf{get}_f\ s} (\mathsf{get}_f\ s_2) \Rightarrow (\exists\ s_3\ .\ \mathsf{get}_f\ s_1 = \mathsf{get}_f\ s_3 \Rightarrow s_3 \prec_s s_2)$$

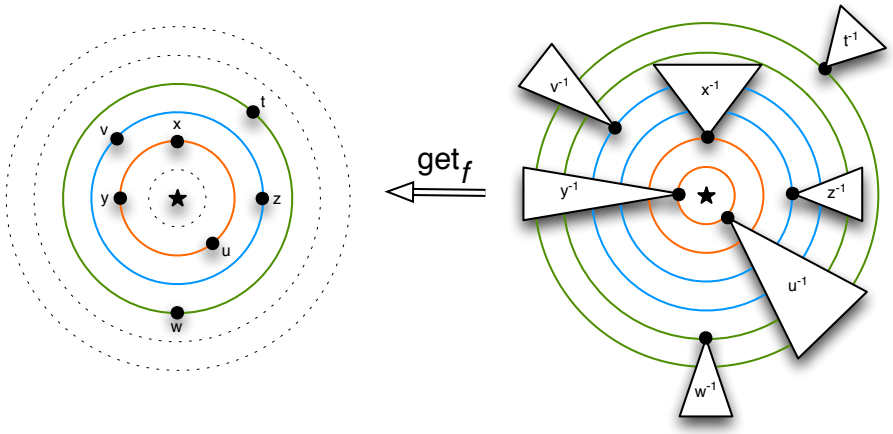


Figure 5: Quasi monotonic transformation

Proposition 15 *If $f : S_{\preceq} \blacktriangleright U$ and $g : U_{\sqsubseteq} \blacktriangleright V$ are well-behaved nondeterministic least-change lenses and get_f is surjective and quasi monotonic between $\preceq_s$ and $\sqsubseteq_{\mathsf{get}_f\, s}$ for every $s \in S$, then $g \cdot f : S_{\preceq} \blacktriangleright V$ is a well-behaved nondeterministic least-change lens.*

Figure 5 illustrates this property. The difference to monotonicity is that only the minimum pre-images of each target are required to respect the relative orders. This suffices for put_f to able to return all closest sources to s given all closest targets to $\mathsf{get}_f\, s$.

Likewise to quasi strictly increasing transformations, preservation of quasi monotonicity by composition requires the surjectivity of the first transformation.

Proposition 16 *If two transformations $\mathsf{get}_f : S_{\preceq} \to U$ and $\mathsf{get}_g : U_{\sqsubseteq} \to V$ are quasi monotonic and get_f is surjective, then $\mathsf{get}_g \cdot \mathsf{get}_f : S_{\preceq} \to V$ is also quasi monotonic.*

Finally, regarding lenses with perfect complement, nondeterministic compositionality is ensured by a monotonic get among sources with the same complement, since this ensures that get is quasi monotonic.

Proposition 17 *If $\mathsf{get}_f : S_{\preceq} \to U_{\sqsubseteq}$ has perfect complement and $s_1 \preceq_s s_2 \wedge \mathsf{cpl}\ s_1 = \mathsf{cpl}\ s_2 \Rightarrow (\mathsf{get}\ s_1) \sqsubseteq_{\mathsf{get}\, s} (\mathsf{get}\ s_2)$ for every $s, s_1, s_2 \in S$, then get_f is quasi monotonic.*

5 Related work

Most modern BX languages (see references [FGM+07, BFP+08, PC10, HHI+10, HEO+11] among others) do not consider the difficult problem of minimizing view updates. In pioneering work, Meertens [Mee98] proposes a framework of constraint maintainers according to a "principle of least change" stating that the action taken by a transformation operation to restore a constraint should be minimal. He then proposes to formalize this notion by introducing a preorder on values and studies the construction of maintainers for particular constraints over sets. However, no

linguistic mechanisms to combine maintainers following the principle of least change, including composition, are proposed.

Much existing work in the context of database view updating is concerned with, given a view function, deriving an update strategy that translates view updates into source updates according to some minimization criteria. [Heg04] introduces a notion of order on sources ($\preceq_S$) and on views ($\preceq_V$) and postulates, among other properties, that view updating shall be order-compatible (get $s \preceq_V v \Rightarrow s \preceq_S$ put $s\ v$). Note that his notion of order is only between two values of the same type (hence $\preceq_S$, for a type S), and the above property formalizes that if an updated view is at most an original view, then the updated source shall be at most the original source[7]. In the context of database tables, his particular orders imply the reflection of view insertions as source insertions, and similarly for deletions. He then establishes that, under particular conditions and for monotonic get functions, there is a unique translation of insertion and deletion view updates under a constant complement approach [BS81]. More recently, [JRW10] shows the connection between the constant complement update translators of [Heg04] and lenses, by demonstrating that they arise from very well-behaved lenses in a category of ordered sets.

[Kel86] acknowledges the ambiguity of view-update translation for non-constant complement approaches (well-behaved lenses), and proposes an interactive algorithm that runs a dialog with the view programmer to choose a particular put function that obeys 5 minimization criteria. [LS91] claims that the view information is not sufficient to disambiguate view updates, and proposes to consider not only a dialog with the view programmer but also a dialog with the view user, obeying similar minimization criteria. Their minimization criteria can be seen as an order on sources and their algorithms as a nondeterministic least-change lens, whereas the dialog is an additional mechanism to allow choosing a minimal source from the various possible. Nevertheless, these approaches consider whole get functions and do not allow reasoning by composition.

[BKT02] studies the complexity of minimizing view updates for a monotonic fragment of relational algebra, for two kinds of deletion and annotation updates, and conclude that this problem is NP-hard for queries including both projection-join or join-union. The authors consider two measures for minimality: first the number of changes in the source, and second the number of side effects in the view caused by the view update. Although our notion of order is more natural with the former measure (since the lens laws disallow view side effects), our framework of least-change lenses is general in the sense that it does not assume any particular order on states nor language of updates. Also, while they study the problem of inverting an individual lens in a minimal way, we consider the orthogonal problem of composing "minimized" lenses.

In previous work [MPC12], we claim that nondeterminism is necessary to support non-surjective BX languages and propose a language of nondeterministic lenses whose put functions are the largest nondeterministic ones satisfying the BX laws. Nevertheless, as we noted in [MPC12], such nondeterminism needs to be controlled to be useful in practice and this paper proposes such an approach by selecting only the minimal values according to a known order.

A solver-based bidirectional model transformation tool is proposed by [CREP10], that introduces the declarative JTL language inspired by a QVT-like syntax. A JTL BX is nondeterministic

[7] These order-based lenses are significantly distinct from least-change lenses: our notion of order $\preceq_s$ is "triangular", as it relates two updated values according to their distance to an original value s, and our least-change properties entail that an updated source must be at most any other possible source with the same updated view.

because: if a modified target model has a trace to a source model, such source model is modified, otherwise all possible source model candidates are returned. In a similar approach, we propose the bidirectionalization of QVT-Relational programs using the Alloy language [MC13]. A BX in our tool is a nondeterministic least-change maintainer that finds all possible source models that are at a minimal distance to the source (and vice-versa), according to user-specified distance metrics. Still, none of these languages nor the QVT standard support composition.

Stevens [Ste10] argues that transformations should always be deterministic in order to be predictable for users, in the sense that users should not observe different results for similar executions. In our perspective, a certain degree of nondeterminism is inescapable in BXs, as predictability also requires allowing users to control the particular semantics of the transformations that the BX language is not able to capture. Notwithstanding, although we reason about composing nondeterministic lenses, we still agree that the final transformations should be deterministic, either by fixing a default choice or by providing a dialog-like mechanism to users.

6 Conclusions and future work

In this paper, we have introduced a framework of least-change lenses that is characterized by two dual scenarios: a deterministic one that only requires that transformations return minimal updates, and a nondeterministic one that requires them to return all minimal updates. We have then presented several sufficient conditions that enable composition of least-change lenses. The properties for well-behavedness and for composition arise naturally from the duality between both scenarios, which has become apparent by formalizing them using the relational calculus notation. Our criteria are still not general enough to encompass least-change lenses we believe can be composed, although more relaxed criteria will most likely not be independent of the composed transformation, denying the advantages of a truly compositional approach. This issue raises the interesting question of completeness: is there a limit on the expressiveness of a least-change lens that can be composed safely with any other such lens? Our (still unproved) conjecture is that such limit is set precisely at quasi strictly increasing transformations for the deterministic scenario, and quasi monotonic transformations for the nondeterministic one.

A problem with some of the proposed criteria is that it is also rather tedious (and unintuitive) to verify if a transformation satisfies them, and we are working on providing better intuitive explanations and proof methods to perform such task. For example, we expected our example transformation T_1 to be quasi monotonic (with distance measures dist_1 and dist_2), but have found that such is not the case. We are currently investigating if other subtle redefinitions of dist_1 and dist_2, namely using distance functions to lexicographic orders, would satisfy any of the criteria.

Our work shows the usefulness of nondeterminism in BXs, namely, by enabling a compositional approach for a wider class of transformations, but still supporting precise and predictable bidirectional laws. We advocate the principle of least change as a tool to trim down nondeterminism to reasonable bounds. In contrast with traditional BX approaches, whose criteria for update translation (besides the regular well-behavedness laws) are usually vague or non-existent, least-change BXs come equipped with an order on sources plus least-change well-behavedness laws that constitute a formal and explicit documentation of the criteria used for update translation. Therefore, reimplementing existing BX approaches in our framework could bring to light their

underlying update translation semantics.

To better suit that goal, we are studying if the composition of least-change lenses should be more flexible, by allowing different orders for different lenses over the same source type: one would write $f : S \trianglerighteq_{\preceq} U$ to state that lens f is well-behaved for order $\preceq$. By doing so, the order could specify more precisely the behavior of a lens, and given two least-change lenses $f : S \trianglerighteq_{\preceq} U$ and $g : U \trianglerighteq_{\sqsubseteq} V$, an interesting research direction is to infer an order $\precsim$ such that $g \cdot f : S \trianglerighteq_{\precsim} V$ is a well-behaved least-change lens. Obviously, one could always default to the discrete metric referred in Section 2, but the goal would be to derive a more precise (i.e. more predictive) order.

In the future we also intend to tackle other combinators to compose BXs besides sequential composition. For example, we are currently researching whether recursive patterns for algebraic data types (like folds or unfolds) can be used to build least-change lenses from simpler ones. A final interesting direction for future work would be to generalize our theory to symmetric BX frameworks like maintainers.

We have modeled our framework in Alloy [Jac12], a lightweight formal specification language with support for automatic model finding via SAT solving. This model has proved very useful in the early stages of this research to rapidly explore and verify/discard different propositions[8]. Of course, such automatic verification is necessarily bounded, and full unbounded calculational proofs for all our propositions were then conducted in Isabelle/HOL [NPW12] proof assistant[9].

Acknowledgements

This work is funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by national funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-020532. The first author is also sponsored by FCT grant SFRH/BD/69585/2010.

Bibliography

[BFP+08] A. Bohannon, J. N. Foster, B. C. Pierce, A. Pilkiewicz, A. Schmitt. Boomerang: resourceful lenses for string data. In *POPL 2008*. Pp. 407–419. ACM, 2008.

[BKT02] P. Buneman, S. Khanna, W.-C. Tan. On propagation of deletions and annotations through views. In *PODS 2002*. Pp. 150–158. ACM, 2002.

[BM97] R. Bird, O. de Moor. *Algebra of Programming*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[BS81] F. Bancilhon, N. Spyratos. Update semantics of relational views. *ACM T. Database Syst.* 6(4):557–575, 1981.

[8] The Alloy model can be downloaded from http://wiki.di.uminho.pt/twiki/pub/Research/FATBIT/Publications/lc-lenses.als.

[9] The Isabelle/HOL theory can be downloaded from http://wiki.di.uminho.pt/twiki/pub/Research/FATBIT/Publications/lc-lenses.thy.

[CREP10] A. Cicchetti, D. D. Ruscio, R. Eramo, A. Pierantonio. JTL: a bidirectional and change propagating transformation language. In *SLE 2010*. LNCS 6563, pp. 183–202. Springer, 2010.

[Dis08] Z. Diskin. Algebraic models for bidirectional model synchronization. In *MoDELS 2008*. LNCS 5301, pp. 21–36. Springer, 2008.

[FGM+07] J. Foster, M. Greenwald, J. Moore, B. Pierce, A. Schmitt. Combinators for bidirectional tree transformations: a linguistic approach to the view-update problem. *ACM T. Progr. Lang. Sys.* 29(3), 2007.

[Fos09] J. Foster. *Bidirectional Programming Languages*. PhD thesis, University of Pennsylvania, December 2009.

[Heg04] S. J. Hegner. An order-based theory of updates for closed database views. *Annals of Mathematics and Artificial Intelligence* 40:63–125, 2004.

[HEO+11] F. Hermann, H. Ehrig, F. Orejas, K. Czarnecki, Z. Diskin, Y. Xiong. Correctness of model synchronization based on triple graph grammars. In *MoDELS 2011*. LNCS 6981, pp. 668–682. Springer, 2011.

[HHI+10] S. Hidaka, Z. Hu, K. Inaba, H. Kato, K. Matsuda, K. Nakano. Bidirectionalizing graph transformations. In *ICFP 2010*. Pp. 205–216. ACM, 2010.

[HPW11] M. Hofmann, B. Pierce, D. Wagner. Symmetric lenses. In *POPL 2011*. Pp. 371–384. ACM, 2011.

[Jac12] D. Jackson. *Software Abstractions: Logic, Language, and Analysis*. MIT Press, London, England, revised edition, 2012.

[JRW10] M. Johnson, R. Rosebrugh, R. Wood. Algebras and update strategies. *Journal of Universal Computer Science* 16(5):729–748, 2010.

[Kel86] A. Keller. Choosing a view update translator by dialog at view definition time. In *VLDB 1986*. Pp. 467–474. Morgan Kaufmann Publishers, 1986.

[KU84] A. Keller, J. D. Ullman. On complementary and independent mappings on databases. In *SIGMOD 1984*. Pp. 143–148. ACM, 1984.

[LS91] J. A. Larson, A. P. Sheth. Updating relational views using knowledge at view definition and view update time. *Information Systems* 16(2):145–168, 1991.

[MC13] N. Macedo, A. Cunha. Implementing QVT-R bidirectional model transformations using Alloy. In *FASE 2013*. Volume 7793, pp. 297–311. 2013.

[Mee98] L. Meertens. Designing constraint maintainers for user interaction. 1998. Manuscript available at http://www.kestrel.edu/home/people/meertens.

[MO11] S.-C. Mu, J. N. Oliveira. Programming from Galois connections. In *RAMiCS 2011*. LNCS 6663, pp. 294–313. Springer, 2011.

[MPC12] N. Macedo, H. Pacheco, A. Cunha. Relations as executable specifications: taming partiality and non-determinism using invariants. In *RAMiCS 2012*. LNCS 7560, pp. 146–161. Springer, 2012.

[NPW12] T. Nipkow, L. C. Paulson, M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer, 2012.

[PC10] H. Pacheco, A. Cunha. Generic Point-free Lenses. In *MPC 2010*. LNCS 6120, pp. 331–352. Springer, 2010.

[Ste10] P. Stevens. Bidirectional model transformations in QVT: semantic issues and open questions. *Software and System Modeling* 9(1):7–20, 2010.