Composing Least-change Lenses

Nuno Macedo¹, Hugo Pacheco², Alcino Cunha³ and José N. Oliveira⁴

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, Compositional-ity, Relational calculus

1 Introduction

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

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

\[
\begin{align*}
\text{PUTGET} & \quad \text{GetPut} \\
\text{GetPut} & \quad \text{PutGet}
\end{align*}
\]

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

Person

follows

(a) Model $M_1$.

Person

popularity : Nat

(b) Model $M_2$.

Person

(c) Model $M_3$.

Figure 1: Model ($M_1$) and respective views ($M_2, M_3$) of a simplified Twitter.

Here $s' \in \text{put}_i s \forall$ means that $\text{put}_i s \forall$ is defined and that $s' = \text{put}_i s \forall$. 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 main goal of a BX framework is, given a bidirectional program specified in a (restricted subset of a) general purpose language or a domain-specific BX language, to automatically derive suitable forward and backward transformations. There are several approaches to perform such “bidirectionalization”\footnote{We refer to “bidirectionalization” as the process of providing a bidirectional semantics for a language, either by construction or other means, as classified in [Pac12, Chapter 3].}, ranging from syntactic or semantic inversion, to compositional techniques, where “correct-by-construction” complex BXs are built from primitive well-behaved transformations. The classic lens framework is compositional: writing a bidirectional program amounts to writing the get transformation using a set of primitive transformations and combinators, and a well-behaved put transformation is provided for “free”.

In general, the above BX laws allow multiple valid backward transformations. For instance, consider the class diagram $M_1$ 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 $M_2$ 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 $M_3$ 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 the 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 lead to a non-minimal update of the composed global transformation ($T_2$ after $T_1$). Some sort of compositionality can still be achieved if we allow put to be nondeterministic and require it to be able to generate all sources at the minimal distance. In some situations, this will allow the put of the first transformation to consider the different alternatives, and eventually 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
Composing least-change lenses

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 total preorder comparing sources relative to their “distance” to some other source. For any source value \( s \in S \), such an order on \( S \) will be denoted by \( \preceq_s \) and will required 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.,

\[
s_1 \preceq_s s_2 \iff s_1 = s
\]

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 \( \text{dist} : S \to S \to \mathbb{N} \) on sources \( S \), such that [Mee98]

\[
s_1 \preceq_s s_2 \iff (\text{dist}_s s_1) \leq (\text{dist}_s s_2)
\]

where \( \leq : \mathbb{N} \to \mathbb{N} \) is the standard order on natural numbers. The order is stable if the distance is stable, i.e., the closest value to a source \( s \) is itself\(^4\):

\[
\text{dist}_s s' = 0 \iff s = s'
\]

An instance of \( \mathcal{M}_1 \) 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 \( \mathcal{M}_1 \) is the size of their symmetric differences:

\[
\text{dist}_1 (W, F) (W', F') = \#(W \triangle W') + \#(F \triangle F')
\]

where \( X \triangle Y = (X - Y) \cup (Y - X) \)

An instance of \( \mathcal{M}_2 \) 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):

\[
\text{dist}_2 (W, P) (W', P') = \#(W \triangle W') + \#(P \triangle P')
\]

\(^3\) By a total order we mean an order \( \leq \) such that, for any points \( x, y \), either \( x \leq y \) or \( y \leq x \) holds. In case both hold this does not mean they are the same (anti-symmetry). For instance, any two points at the same distance will be comparable in either way and they need not be the same. However, this would be the case for injective distance functions.

\(^4\) In a metric space, this clause is known as **identity of indiscernibles**. In fact, it could be reasonable to assume that distances satisfy other properties of metric spaces, such as **symmetry** or **triangle inequality**, although these are orthogonal to the properties of least-change lenses.
We could however ignore the popularity of people not contained in both instances:

\[
\text{dist}'_s (W, P) (W', P') = \#(W \Delta W') + \#((P' \setminus (W \cap W')) \Delta (P \setminus (W' \cap W')))
\]

where \( (X \setminus Y) x = \text{if } x \in Y \text{ then } X x \text{ else } 0 \)

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 \(l : S \triangleright\preceq V\) consists of a forward total transformation \(\text{get}_l : S \rightarrow V\), and a partial backward transformation \(\text{put}_l : S \rightarrow V \rightarrow S\). A least-change lens is well-behaved if it satisfies \(\text{GETPUT}\) and

\[
\begin{align*}
\text{LEASTPUTGET} & \quad s' \in \text{put}_l s \triangleright v \Rightarrow v = \text{get}_l s' \land (\forall s_1 . v = \text{get}_l s_1 \Rightarrow s' \preceq_s s_1)
\end{align*}
\]

\(\text{LEASTPUTGET}\) is equivalent to the original least change principle proposed in [Mee98]. It is a refinement of \(\text{PUTGET}\), 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 \(v\). \(\text{LEASTPUTGET}\) thus lowers the upper bound for \(\text{put}\) and reduces the range of valid updates.

Although the backward transformation is more restricted, there may still be more than one valid \(\text{put}\) for the same \(\text{get}\) 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 \(\text{dist}_1\).

If we add a person \(p\) to the view with popularity \(n\), \(\text{put}\) 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 \(\text{dist}_2\), then when a new person is added to the view \(\text{put}\) must assign him popularity 0. However, if we use the preorder induced by \(\text{dist}'_2\) instead, \(\text{put}\) is free to assign the new person any valid popularity. In both cases, \(\text{put}\) 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 \(^5\):

\[
\text{dist}_s s' = \text{if } s = s' \text{ then } 0 \text{ else } 1
\]

In this case, \(\text{LEASTPUTGET}\) degenerates into the original \(\text{PUTGET}\), 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 \(\text{put}\). This will occur when \(\text{dist}_s\) 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 \(\text{put}\) 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 \(\text{put}\) is nondeterministic, and enumerates all possible minimal updates. To distinguish it from the deterministic case it will be denoted as \(\text{Put}\).

\(^5\) In metric space terminology, this is known as the discrete metric on a set.
Composing least-change lenses

Definition 4 (Nondeterministic least-change lens) Given a family of stable total preorders $\leq_s$ for every $s \in S$, a nondeterministic least-change lens $l : S \rightharpoonup V$ consists of a forward total transformation $\text{get}_l : S \rightarrow V$, and a nondeterministic backward transformation $\text{Put}_l : S \rightarrow V \rightarrow \mathcal{P}(S)$.

The lens is said to be well-behaved if it satisfies $\text{PUTGET}$ and $\text{LEASTGETPUT}$

$$v = \text{get}_l s' \land (\forall s_1 . v = \text{get}_l s_1 \Rightarrow s' \leq_s s_1) \Rightarrow s' \in \text{Put}_l s . v$$

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

Returning to our transformation $T_2$ with distance measure $\text{dist}_2$, when a new person is added to the view, a well-behaved $\text{Put}$ 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 $f : A \rightarrow \mathcal{P}(B)$ 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 $2 \leq 3$. Infix notation suggests writing $R : B \leftarrow A \text{ (or } R : A \rightarrow B \text{) instead of } R \subseteq B \times A \text{ as the type of relation } R$, 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 $R : A \rightarrow B$ and $S : B \rightarrow C$, their composition, denoted by $S \cdot R : A \rightarrow C$, such that $c \in (S \cdot R) a$ 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 $R : A \rightarrow B$, $R^c : B \rightarrow A$, which is such that $a R^c b$ means the same as $b R a$. A function $f : A \rightarrow B$ is a special case of relation, in that it is deterministic and totally defined. Thus both $b f a$ and $a f^o b$ mean $b = f a$. For instance, expression $f^o \cdot R \cdot g$ 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 $R : A \rightarrow B$ and $S : A \rightarrow C$, denoted by $R \triangle S : A \rightarrow B \times C$, $(b, c) \in (R \triangle S) a$ holding whenever $b R a$ and $c S a$ hold. The notation $\delta a : B \rightarrow A$ 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 $R : C \rightarrow A$ and $S : C \rightarrow B$ denoted $S / R : A \rightarrow B$ (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 $R : A \rightarrow B$ will be denoted by $\delta R : A \rightarrow A$ (a $\delta R a \Leftrightarrow \exists b . b R a$) and $\rho R : B \rightarrow B$ (a $\rho R b \Leftrightarrow \exists a . b R a$) 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 $s \subseteq \text{get}^o$ means the same as the $\text{PUTGET}$ law: $s' \in \text{put} s v \Rightarrow v = \text{get} s'$. Similar re-statement of $\text{PUTGET}$ and $\text{GETPUT}$ in relational algebra makes it clear that they establish lower and upper bounds for put,

$$\text{put} s \subseteq \text{get}^o \quad \text{PUTGET}$$
$$\delta s \cdot \text{get}^o \subseteq \text{put} s \quad \text{GETPUT}$$
where \( \rho_s = \{(s, s)\} \) filters out values other than \( s \), restricting the output of \( \text{get}^o \) to \( s \). As \( \rho_s \) 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 \text{LeastPutGet} and \text{LeastGetPut} laws, can be encoded in terms of the so-called \textit{shrink} operator \( R \upharpoonright S : A \rightarrow B \) [MO11] that minimizes the output of a relation \( R : A \rightarrow B \) in regard to a relation \( S : B \rightarrow B \), and defined as follows:

\[
R \upharpoonright S = R \setminus S / R
\]

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

\[
\begin{align*}
\text{Put } s \subseteq \text{get}^o \upharpoonright \preceq_s & \quad \text{LeastPutGet} \\
\text{get}^o \upharpoonright \preceq_s \subseteq \text{Put } s & \quad \text{LeastGetPut}
\end{align*}
\]

Clearly, these laws are dual of each other. Moreover, it can be shown that they refine the original laws in the sense that \text{LeastPutGet} lowers the upper-bound imposed by \text{PutGet} and \text{LeastGetPut} raises the lower-bound imposed by \text{GetPut}.

**Proposition 1** Given a family of stable total preorders \( \preceq_s : S \rightarrow S \), we have

\[
\rho_s \cdot \text{get}^o \subseteq \text{get}^o \upharpoonright \preceq_s \subseteq \text{get}^o
\]

**Proof.** The upper bound is trivial since \( \text{get}^o \cap \preceq_s / \text{get} \subseteq \text{get}^o \); for the lower bound we have

\[
\begin{align*}
\rho_s \cdot \text{get}^o & \subseteq \text{get}^o \upharpoonright \preceq_s \\
\equiv & \{ \text{-shrink definition -} \} \\
\rho_s \cdot \text{get}^o & \subseteq \text{get}^o \cap \preceq_s \cap \text{get} \\
\equiv & \{ \text{-universal-\cap; division -} \} \\
\rho_s \cdot \text{get}^o & \subseteq \text{get}^o \cap \rho_s \cdot \text{get} \cap \preceq_s \\
\equiv & \{ \rho_s = s \cdot s^o; \text{-shunting -} \} \\
\rho_s \cdot \text{get}^o \cdot \text{get} & \subseteq s^o \cdot \preceq_s \\
\equiv & \{ \text{-OrdTable thus } s^o \cdot \preceq_s = T \} \\
s^o \cdot \text{get}^o \cdot \text{get} & \subseteq T
\end{align*}
\]

Note how \text{OrdTable} is required for this proof. This was expected, since the lower bound is only restricting the result of \text{put} for the original source \( s \), and \text{OrdTable} states precisely that it is a minimum of the preorder. This proposition also shows that \text{LeastGetPut} and \text{PutGet} are consistent bounds.

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

\(^6\) Since antisymmetric shrinking criteria ensure determinism [MO11].
Composing least-change lenses

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

*Proof.* Trivial, since \(\preceq_s\) is reflexive and shrinking deterministic relations by reflexive orderings has no effect \([\text{MO11}]\). \(\square\)

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

**Proposition 3** Every well-behaved deterministic least-change lens \(l : S \succcurlyeq V\) is a well-behaved regular lens \(l : S \triangleright V\).

*Proof.* \text{GETPUT} is a requirement for least-change lenses and \text{PUTGET} follows from Proposition 1. \(\square\)

**Proposition 4** Every well-behaved regular lens \(l : S \triangleright V\) is a well-behaved deterministic least-change lens \(l : S \succcurlyeq V\).

*Proof.* Take the total preorder induced by dist \(s s' = \text{if} s = s' \text{ then } 0 \text{ else } 1\). Then \text{LEASTPUTGET} degenerates into \text{PUTGET}. \(\square\)

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

**Proposition 5** Every well-behaved regular lens \(l : S \triangleright V\) is a well-behaved nondeterministic least-change lens \(l : S \succcurlyeq V\).

*Proof.* For the total preorder

\[
s'' \preceq_s s' \iff (s' \in \text{Put}_l s (\text{get}_l s') \Rightarrow s'' \in \text{Put}_l s (\text{get}_l s''))
\]

get\(^\circ\) \(\mid \preceq_s\) is deterministic and equal to the only possible put. \text{LEASTGETPUT} follows easily. \(\square\)

## 4 Criteria for composing least-change lenses

Given two well-behaved least-change lenses \(f : S \succcurlyeq U\) and \(g : U \triangleright V\), we will now discuss when can they be composed in order to obtain a well-behaved least-change lens \(g \cdot f : S \succcurlyeq V\) using the lens composition combinator \((\cdot)\) \([\text{FGM}^+07]\):

\[
\begin{align*}
\text{get}_{g \cdot f} s &= (\text{get}_g \cdot \text{get}_f) s \\
\text{put}_{g \cdot f} s &= (\text{put}_g s) \cdot (\text{put}_f (\text{get}_g s))
\end{align*}
\]

In general this is not true, because \(\text{put}_{g \cdot f}\) is not guaranteed to return minimal values. For instance, consider the following example depicting the forward transformations get\(_f\) and get\(_g\), and where
\( \text{dist } x_i x_j = |j - i| \) for both lenses:

\[
\begin{array}{c}
S_1 \xrightarrow{\text{get}_f} U_1 \xrightarrow{\text{get}_g} V_1 \\
S_2 \xrightarrow{\text{get}_g} V_2
\end{array}
\]

Consider the backward transformation of the view \( V_2 \) over the original source \( S_1 \) of the \( g \cdot f \) lens: first, \( \text{put}_g v_2 (\text{get}_f s_1) = u_2 \), since \( u_2 \) is closer to \( u_1 \) (the original intermediate \( U \)-view) than \( u_5 \); then, \( \text{put}_f s_1 u_2 = s_9 \), since it is the only choice for \( u_2 \). However, this is not the minimal result for the least-change lens \( g \cdot f \), 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 \( \text{put}_g \) selected from the range of valid ones may not result in a minimum \( S \)-view when passed through \( \text{put}_f \).

We will see that in order to preserve \text{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 \text{LEASTPUTGET} is preserved by composition, i.e.:

\[
\forall s . \text{put}_f s \subseteq \text{get}_f^\circ \downarrow \leq_s \land \forall u . \text{put}_g u \subseteq \text{get}_g^\circ \downarrow u \Rightarrow \forall s . \text{put}_f s \subseteq (\text{get}_g \cdot \text{get}_f)^\circ \downarrow \leq_s
\]

The simplest condition for this to hold is for \( \text{get}_g \) to be injective: since \( \text{get}_g^\circ \) is deterministic, there is a unique valid \( \text{put}_g \), and thus there is no ambiguity in the choice.

**Proposition 6** If \( f : S \trianglerighteq U \) and \( g : U \trianglerighteq V \) are well-behaved least-change lenses and \( \text{get}_g \) is injective, then \( g \cdot f : S \trianglerighteq V \) is a well-behaved least-change lens.

Having an injective \( \text{get}_f \), however, is not enough to preserve \text{LEASTPUTGET}. Still, a sufficient condition on \( \text{get}_f \) is for it to be strictly increasing between the preorders \( \leq_s \) and \( \sqsubseteq_{\text{get}_f} s \):

\[
s_1 \prec_s s_2 \Rightarrow (\text{get}_f s_1) \sqsubseteq_{\text{get}_f} (\text{get}_f s_2)
\]

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

\[
\text{get}_f \cdot \prec_s \sqsubseteq \sqsubseteq_{\text{get}_f} \cdot \text{get}_f
\]

**Proposition 7** If \( f : S \trianglerighteq U \) and \( g : U \trianglerighteq V \) are well-behaved least-change lenses and \( \text{get}_f \) is strictly increasing between \( \leq_s \) and \( \sqsubseteq_{\text{get}_f} s \), then \( g \cdot f : S \trianglerighteq V \) is a well-behaved least-change lens.

Unfortunately this requirement is too restrictive, as it forces \( \text{get}_f \) to be injective, meaning that it cannot abstract information from the source. To help understand why composition succeeds
Composing least-change lenses

Figure 2: Strictly increasing transformation

when \( \text{get}_f \) 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 \( \text{get}_f s \) (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 \( \text{get}_f \). In this case, we have a one-to-one correspondence since \( \text{get}_f \), being strictly increasing, must be injective. Another consequence of this property is that all values of \( U \) that are at the same distance from \( \text{get}_f s \) 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 \( y \preceq_{\text{get}_f} z \) 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, \( \text{put}_g \) will return one of the values that is closest to \( \text{get}_f s \), in this case either \( x, y \) or \( u \); \( \text{put}_f \) 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 \( \text{put}_g \).

A more interesting requirement is the following more relaxed version of \text{StrictINC}, that does not imply injectivity:

\[
(\forall s_3 . \text{get}_f s_2 = \text{get}_f s_3 \Rightarrow s_1 \prec s_3) \Rightarrow (\text{get}_f s_1) \sqsubseteq_{\text{get}_f} (\text{get}_f s_2)
\]

In point-free notation this is written

\[
(\prec / \text{get}_f) \cdot \rho \text{get}_f \subseteq \text{get}_f \cdot \rho \text{get}_f
\]

\text{QuasiStrictINC}

Now a view \( \text{get}_f s_1 \) is required to be smaller than another view \( \text{get}_f s_2 \) 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:

\[
(\text{get}_f s_1) \sqsubseteq_{\text{get}_f} (\text{get}_f s_2) \Rightarrow (\exists s_3 . \text{get}_f s_1 = \text{get}_f s_3 \land s_3 \preceq s_2)
\]

written \( \rho \text{get}_f \cdot \sqsubseteq_{\text{get}_f} \cdot \text{get}_f \subseteq \text{get}_f \cdot \preceq \) in point-free. In other words, when a view \( v_1 \) is smaller than or equal to another view \( v_2 \) (in distance to a view \( \text{get}_f s \)), 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 difference to \text{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 \( s \), the minimum values of their pre-images must all be at the same distance from \( s \).

**Proposition 8**  
If \( f : S \geq U \) and \( g : U \preceq V \) are well-behaved least-change lenses and \( \text{get}_f \) is quasi strictly increasing between \( \preceq_s \) and \( \preceq_{\text{get}_f s} \), then \( g \cdot f : S \geq V \) is a well-behaved least-change lens.

Figure 3 also helps understanding why this proposition holds. Again, \( \text{put}_x \) is free to choose a value closest to \( \text{get}_f s \) but, whatever the choice made, \( \text{put}_x \) 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).

An interesting class is that of lenses with a perfect complement. A function \( \text{cpl} \) is said to be a complement of \( \text{get} \) if \( \text{get} \cdot \text{cpl} \) 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 \( \text{get} \cdot \text{cpl} \) 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 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, as \( \text{get} \cdot \text{cpl} \) is also required to be strictly increasing in the following sense (essentially, assuming an order on pairs that is product consistent [Mee98]):

\[
s_1 \prec_s s_2 \Rightarrow (\text{get } s_1) \sqsubseteq_{\text{get } s} (\text{get } s_2) \lor (\text{cpl } s_1) \sqsubseteq_{\text{cpl } s} (\text{cpl } s_2)
\]

**Proposition 9**  
If \( f : S \geq U \) and \( g : U \preceq V \) are well-behaved least-change lenses, and \( \text{get}_f \) has
Composing least-change lenses

**perfect complement** and \( \text{get}_f \triangle \text{cpl}_f \) is strictly increasing in the above sense, then \( g \cdot f : S \trianglerighteq V \) is a well-behaved least-change lens. The same holds for \( \text{get}_g \).

Having a perfect complement requires \( \text{get}_f \) to be surjective. In a lens framework this is not a drawback, since it is usually required due to totality constraints [FGM+07, PC10].

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

\[
\forall s . \text{get}_f^o \mid \preceq_s \subseteq \text{Put}_f s \land \forall u . \text{get}_g^o \mid \preceq_u \subseteq \text{Put}_g u \Rightarrow \forall s . (\text{get}_g \cdot \text{get}_f)^o \mid \preceq_s \subseteq \text{Put}_{g \cdot f} s
\]

Once again, an injective \( \text{get}_g \) guarantees the compositionality.

**Proposition 10** If \( f : S \trianglerighteq U \) and \( g : U \trianglerighteq V \) are well-behaved nondeterministic least-change lenses and \( \text{get}_g \) is injective, then \( g \cdot f : S \trianglerighteq V \) 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 \( \text{get}_f \) to be strictly increasing it suffices to require it to be monotonic:

\[
s_1 \preceq_s s_2 \Rightarrow (\text{get}_f s_1) \sqsubseteq_{\text{get}_f} (\text{get}_f s_2)
\]

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:

\[
(\text{get}_f s_1) \sqsubseteq_{\text{get}_f} (\text{get}_f s_2) \Rightarrow s_1 \prec_s s_2
\]

In the point-free notation we have

\[
\text{get}_f \cdot \preceq_s \subseteq \sqsubseteq_{\text{get}_f \cdot \text{get}_f} \quad \text{MONOT}
\]

**Proposition 11** If \( f : S \trianglerighteq U \) and \( g : U \trianglerighteq V \) are well-behaved nondeterministic least-change lenses and \( \text{get}_f \) is surjective and monotonic between \( \preceq_s \) and \( \sqsubseteq_{\text{get}_f s} \), then \( g \cdot f : S \trianglerighteq V \) is a well-behaved nondeterministic least-change lens.

Unlike the strictly increasing property, monotonicity no longer implies that \( \text{get}_f \) is injective. Unlike in Proposition 7, here we require \( \text{get}_f \) to be surjective: note that surjectivity is the dual property of being deterministic, which \( \text{get}_f \) always satisfies. Figure 4 helps understanding the differences between being strictly increasing and monotonic, and why this property ensures a well-behaved nondeterministic composition. Unlike in Figure 2, the pre-images of the values of type \( U \) at the same distance from \( \text{get}_f s \) are allowed to be at different distances from \( s \). However, we have the restriction that, for example, when \( y \sqsubseteq_{\text{get}_f s} z \) then \( y^{-1} \prec_s z^{-1} \), forcing the pre-images of targets at different distances from \( \text{get}_f s \) to also be at different distances from \( s \) (respecting the relative orders). This ensures that when \( \text{pay}_{g\cdot s} \) returns all values at the minimum distance from \( \text{get}_f s \), 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 have transformations whose composition is well-behaved in the deterministic scenario, but not well-behaved in the non-deterministic one. Figure 2 helps understanding why: even if \( \text{put}_f \) returns all values closest to \( \text{get}_f s \), there is no chance for \( \text{put}_f \) to return all values closest to \( s \), since some of the pre-images closest to \( s \) originate from targets further away from \( \text{get}_f s \) (namely, \( v \) and \( z \)).

Again, monotonicity can be relaxed to

\[
(\forall s_3 . \text{get}_f s_2 = \text{get}_f s_3 \Rightarrow s_1 \preceq_s s_3) \Rightarrow (\text{get}_f s_1) \sqsubseteq_{\text{get}_f} (\text{get}_f s_2)
\]

or with the point-free notation:

\[
(\preceq_s / \text{get}_f) \cdot \rho \text{get}_f \subseteq \text{get}_f \cdot \rho \text{get}_f \quad \text{QUASIMONOT}
\]

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 \( \text{get}_f s_1 \) shall be at most \( v \) (in distance to a view \( \text{get}_f s \)). Alternatively, when a view \( v_1 \) is smaller than another view \( v_2 \) (in distance to a view \( \text{get}_f s \)), then at least one source with view \( v_1 \) is smaller than all sources with view \( v_2 \) (in distance to a source \( s \)):

\[
(\text{get}_f s_1) \sqsubseteq_{\text{get}_f} (\text{get}_f s_2) \Rightarrow (\exists s_3 . \text{get}_f s_1 = \text{get}_f s_3 \Rightarrow s_3 \prec_s s_2)
\]

**Proposition 12** If \( f : S \rhd \succeq U \) and \( g : U \rhd \subseteq V \) are well-behaved nondeterministic least-change lenses and \( \text{get}_f \) is surjective and quasi monotonic between \( \preceq_s \) and \( \sqsubseteq_{\text{get}_f} \), then \( g \cdot f : S \rhd \succeq V \) 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 \( \text{put}_f \) to able to return all closest sources to \( s \) given all closest targets to \( \text{get}_f s \).

Finally, regarding lenses with perfect complement, nondeterministic compositionality is ensured by a monotonic \( \text{get} \Delta \text{cpl} \) in the following sense:
Composing least-change lenses

\[
\begin{align*}
&\text{Figure 5: Quasi monotonic transformation}
\end{align*}
\]

\[s_1 \preceq s_2 \Rightarrow (\text{get } s_1) \sqsubseteq_{\text{get } s} (\text{get } s_2) \wedge (\text{cpl } s_1) \sqsubseteq_{\text{cpl } s} (\text{cpl } s_2)\]

Remarking that for a lens with perfect complement a source \(s\) can be uniquely decomposed into a \((\text{get } s, \text{cpl } s)\) pair, this monotonicity property acknowledges that the order on sources shall be \textit{product consistent} [Mee98], i.e., built according to the orders on views and on complements.

**Proposition 13** If \(f : S \triangleright S U\) and \(g : U \triangleright V\) are well-behaved nondeterministic least-change lenses, and \(\text{get}_f\) has perfect complement and \(\text{get}_f \triangle \text{cpl}_f\) is monotonic in the above sense, then \(g \cdot f : S \triangleright V\) is a well-behaved nondeterministic least-change lens. The same holds for \(\text{get}_{g\cdot f}\).

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 (\(\text{get } s \preceq_V v \Rightarrow s \preceq_S \text{ put } s v\)). 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

\(^7\) These order-based lenses are significantly distinct from least-change lenses: our notion of order \(\preceq_S\) is “triangular”, as

Proc. BX 2013 14 / 18
context of database tables, his particular orders imply the reflection 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 introduce the declarative JTL language inspired by a QVT-like syntax. A JTL BX is nondeterministic 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 only finds all possible source models that are at a minimal distance to the source (and vice-versa), according to possibly 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.

---

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.
that the BX language is not able to capture. Notwithstanding, although we reason about composing nondeterministic lenses, we still agree that the final transformations shall 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 involve both lenses being composed. An interesting question concerns 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 $\text{dist}_1$ and $\text{dist}_2$), but recently found a counter-example for this claim. We are also currently investigating if other subtle redefinitions of $\text{dist}_1$ and $\text{dist}_2$ 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.

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\footnote{The Alloy model can be downloaded from \url{http://wiki.di.uminho.pt/twiki/pub/Research/FATBIT/Publications/lc-lenses.als}.}. Of course, such automatic verification is necessarily bounded, and we are in the process of concluding full unbounded calculational proofs for all our propositions using the Isabelle/HOL [NPW12] proof assistant\footnote{The current status of this effort can be downloaded from \url{http://wiki.di.uminho.pt/twiki/pub/Research/FATBIT/Publications/lc-lenses.thy}.}.

\footnote{The Alloy model can be downloaded from \url{http://wiki.di.uminho.pt/twiki/pub/Research/FATBIT/Publications/lc-lenses.als}.}

\footnote{The current status of this effort can be downloaded from \url{http://wiki.di.uminho.pt/twiki/pub/Research/FATBIT/Publications/lc-lenses.thy}.}
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. Also, the fine-grained bounds introduced by the least-change lens laws suggest that a forward transformation plus an order on sources can provide a better (partial) specification for constructing conforming backward transformations. Building concrete least-change lens languages that can automatically derive such specifications shall involve the judicious study of suitable orders and primitive lenses for particular data domains. A final interesting direction for future work would be to generalize our theory to symmetric BX frameworks like maintainers.

Bibliography


Composing least-change lenses


