Journal of Object Technology
Published by AITO — Association Internationale pour les Technologies Objets
http://www.jot.fm/

On principles of Least Change and Least Surprise for bidirectional transformations

James Cheney[a] Jeremy Gibbons[b] James McKinna[a]
Perdita Stevens[a]

a. Laboratory for Foundations of Computer Science, School of Informatics, University of Edinburgh

b. Department of Computer Science, University of Oxford

Abstract In software engineering and elsewhere, different people may work intensively with different, but related, artefacts, e.g. models, documents, or code. They may use bidirectional transformations (bx) to maintain consistency between them. Naturally, they do not want their deliberate decisions disrupted, or their comprehension of their artefact interfered with, by a bx that makes changes to their artefact beyond the strictly necessary. This gives rise to a desire for a principle of Least Change, which has been often alluded to in the field, but seldom addressed head on. In this paper we present examples, briefly survey what has been said about least change in the context of bx, and identify relevant notions from elsewhere that may be applicable. We conclude that we cannot expect a Principle of Least Change to determine the optimal behaviour of a bx based on the consistency relation it embodies alone. Any such principle would bind the hands of the bx developer too tightly: the specification of *how* consistency is restored is as important a part of the development of a bx as the specification of *what* consistency means. Rather, what is required is a notion of *reasonable* behaviour of a bx that captures the idea that the bx's consistency restoration does not gratuitously surprise its user. We suggest considering continuity variants, particularly Hölder continuity. Such properties are too strong to expect them to hold universally, so we introduce the idea of a property holding piecewise on an *atlas* of subspace pairs.

Keywords bidirectional transformation; least change; least surprise; semantics; metrics; continuity; atlas

James Cheney, Jeremy Gibbons, James McKinna, Perdita Stevens. *On principles of Least Change and Least Surprise for bidirectional transformations.* In Journal of Object Technology, vol. 16, no. 1, 2011, pages 3:1–31.
doi:10.5381/jot.2017.16.1.a3

1 Introduction

When people engage with complex tasks that involve very large amounts of information, such as software development, they can easily become overwhelmed. One way to manage this information overload is to enable each person to work with only the information they actually need: that is, to provide an artefact that records, not all the information that exists, but rather, only the information that a particular person must read and modify. Whilst this can simplify each person's task to the point that it is doable, it introduces a new problem: now, when one artefact is changed, other artefacts may need to be changed in response, or else the set of information, taken as a whole, would be inconsistent, stymieing the completion of the whole task.

What are bx? Bidirectional transformations (hereinafter "bx") restore consistency between artefacts; this is the primary definition of what it is to be a bx. Defining a bx includes specifying what consistency should mean in a particular case, though this may have a fixed, understood definition in the domain; it also involves specifying how consistency should be restored, where there is a choice. In interesting cases, the bx does not just regenerate one artefact from another. This is essential in situations where different people or teams are working on both artefacts, otherwise their work would be thrown away every time consistency was restored. Formalisms and tools differ on what information is taken into account: is it just the current states of the two artefacts (the *state-based* or *relational* approach: we will use the latter term in this paper) or does it also include intensional information about how they have recently changed, about how parts of them are related, etc.? A motivation for including such extra information, even at extra cost, is often that it enables bx to be written that have more intuitively reasonable behaviour. We still lack, however, any bx language in mainstream industrial use. The reasons for this are complex, but we believe a factor is that it is not yet known how best to provide reasonable behaviour at reasonable cost, or even what "reasonable behaviour" should mean.

For already in the relational setting, the best definition of "reasonable behaviour" is not obvious. Meertens' seminal but unpublished paper [Mee98] discussed the need for a Principle of Least Change for constraint maintainers, essentially what we now call relational bx. His formulation is:

> The action taken by the maintainer of a constraint after a violation should change no more than is needed to restore the constraint.

It turns out, however, that there are devils in the details.

How should a bx restore consistency? Before diving into how it may be automated, let us consider the motivating model-driven development (MDD) setting further. We will refer to our artefacts as "models", using the term inclusively, and we will use the term "model space" for the collection of possible models, with any structure (e.g. metric) it may have.

MDD is useful to humans because it separates concerns: each person's model includes just what they need, in order to fulfil their specific role. The reason this is helpful is precisely that it avoids distracting – surprising – people by forcing them to be aware of information, and especially *changes*, that do not affect them. The problem is that the separation between relevant information (should be in the model) and irrelevant information (should not be in the model) often cannot be made perfectly: changes to the model are necessitated by changes outside it. This is problematic,

because such changes are inherently surprising: they are caused by changes to information that the user of the model does not have.

Without automated bx, these changes have to be decided on by humans. Imagine I am a member of a large software development project; I normally work on my model, but there are other developers who work with different models, and periodically we must negotiate how our models shall be brought back into a consistent state so that the eventual software can be correctly modelled by all the models. Suppose I am at a developers' meeting focused (solely) on deciding how to restore consistency by changing my model. The meeting may consider many courses of action. Some, however, would be considered unreasonable. For example, if I have added a section to my model which is (agreed to be) irrelevant to the question of whether the models are consistent, then to delete my new section as part of the consistency restoration process would plainly be not only suboptimal, but even unreasonable.

Occasionally the meeting's job will be easy: for example, if there is only one way to restore consistency, the meeting must pick that way, while if the models turn out to be consistent already, then nothing should be done. Sometimes there will be disagreement about what choice is *optimal*, and even about what choices are *(un)reasonable*. There may be discussion, for example, about trade-offs between keeping the changes small, and keeping them easy to describe, or natural in some sense, or maintaining good properties of the models. The meeting attempts to ensure that changes made in order to restore consistency are not more disruptive to the development process than they need to be. For a given informal idea of the "size" of a change to a model, it does not necessarily follow that two changes of that size will be equally disruptive. For example, a change imposed on a part of a model that its developers had thought they understood well is likely to be more disruptive than a change imposed on a part that they did not know much about, e.g. a placeholder; and a change to something the developers felt they owned exclusively is likely to be felt as more disruptive than a change to something that they felt they shared. We see the need for something more like "least disruption" or "least surprise" than "least change".

Humans agree what is reasonable by a social process. Meetings like the ones we have imagined are time-consuming and error-prone, but they allow progress to be made without the need for formalism. If they are to be replaced by the use of a bx tool, however, then the tool must provide usable, trustworthy guarantees of good behaviour.

Why formalise properties of bx? We think that making explicit the least change properties that one might hope to achieve in a bx is a step towards guiding the design and use of future languages. In the long term, this should support bx being incorporated into software development in such a way that they do not violate the "least surprise" rule familiar from HCI: their developers and their users should be able to rely on bx behaving "reasonably". If a community could agree on a precise version of a least change principle that should always be obeyed, we might hope to develop a bx language in which *any* legal bx would satisfy that principle (analogous to "well typed programs don't go wrong"). If that proves too ambitious – perhaps there is a precise version of the principle that is agreed to be desirable, but which must on occasion be disobeyed by a reasonable bx – then a fall-back position would be to develop analysis tools that would be capable of flagging when a particular bx was at risk of disobeying the principle. It could then be scrutinised particularly closely e.g. in testing; or perhaps a bx engine would provide special run-time behaviour in

situations where it might otherwise surprise its users, e.g. asking the user to make or confirm the selection of changes, while a "safer" change propagation could be made without interaction.

A key insight from imagining the developers' meeting is this: the best way to restore consistency depends on the circumstances. We cannot generally expect to synthesise the consistency restoration functions from the consistency relation. Rather, we need to capture a property of the bx that will rule out consistency restoration functions that are *unreasonable*, leaving the writer of the bx to choose between multiple reasonable options for how to program consistency restoration.

1.1 Structure of the paper

Section 2 introduces some terminology and notation that we shall need. Next, in Section 3 we consider perhaps the purest form of "least change", based on metrics on the model spaces. We prove an NP-hardness result for this approach, and discuss other reasons why we might not wish to adopt it.

In Sections 4 and 5 we address two major dimensions on which possible principles may vary. The first dimension concerns what structure is relevant to the relative cost we consider a change to have. This dimension has had considerable attention, but we think it is worth taking a unified look. The second, concerning whether we offer guarantees when concurrent editing has taken place, seems to be new, and based on our investigations we think it may be important.

In Section 6 we consider approaches based on ordering changes themselves, especially where a change can be identified with sets of parts to be added and removed, ordered by inclusion.

In Section 7 we consider in some detail an approach based on guaranteeing "reasonably small" changes. We find a promising property, but consider it too strong to be expected to hold universally. Therefore in Section 8 we introduce the idea of an *atlas* of subspace pairs, allowing us to describe bx that have some desirable property piecewise. We speculate about how this could be embodied in future languages and tools.

Section 9 briefly considers what might be learned from the field of automated software repair. Section 10 considers categorical approaches, and Section 11 concludes and discusses some potentially relevant work that has not otherwise been mentioned. A major goal of the paper is to inspire future work, though, and we point out areas that might repay it throughout. We do not have a Related Work section, because most of the paper is about related work. Our own technical contributions are in Section 7 and Section 8, along with the NP-hardness result in Section 3. That result together with the entirely new Sections 8 and 9 are the main discrete additions in this paper compared with the parent workshop paper [CGMS15]; the rest of the paper has also been reworked and expanded following helpful discussion at Bx 2015.

Throughout, we give examples to illustrate points we make. Names are those used in the Bx Examples Repository[1] [CMSG14].

2 Background

Bx basics When talking about state-based relational bx we use the now-standard (see e.g. [Ste13]) notation $R : M \leftrightarrow N$ for a bx comprising consistency relation R and

[1] http://bx-community.wikidot.com/examples:home

restorers $\overrightarrow{R} : M \times N \to N$, $\overleftarrow{R} : M \times N \to M$. Throughout this paper such bx will be assumed to have the two most important properties, correctness and hippocraticness:

Definition A bx $R : M \leftrightarrow N$ is *correct* if the consistency restorers do restore consistency: that is, for all $m \in M$, $n \in N$, we have $R(m, \overrightarrow{R}(m, n))$, and dually.

Definition A bx $R : M \leftrightarrow N$ is *hippocratic* if the consistency restorers make no change to an already-consistent pair of models: that is, for all $m \in M$, $n \in N$, we have $R(m, n) \Rightarrow \overrightarrow{R}(m, n) = n$, and dually.

Hippocraticness is the simplest least change property: if nothing needs to be changed, change nothing.

For a correct and hippocratic bx, the consistency relation can be recovered from either consistency restorer, because $R(m, n)$ holds iff $\overrightarrow{R}(m, n) = n$, iff $\overleftarrow{R}(m, n) = m$.

A pair of properties of bx that we shall refer to later is undoability and history ignorance.

Definition A bx $R : M \leftrightarrow N$ is *undoable* if for all $m, m' \in M$, $n \in N$, we have $R(m, n) \Rightarrow \overrightarrow{R}(m, \overrightarrow{R}(m', n)) = n$, and dually.

Definition A bx $R : M \leftrightarrow N$ is *history ignorant* or *strongly undoable* if for all $m, m' \in M$, $n \in N$, we have $\overrightarrow{R}(m, \overrightarrow{R}(m', n)) = \overrightarrow{R}(m, n)$, and dually.

This last property was named history ignorance in [Dis08], but here the alternative name "strong undoability" (see [Ste12] for terminological discussion) usefully illustrates a pattern that will recur later. The difference between (weak) and strong undoability is simply the domain of quantification: for strong undoability, (m, n) may be *any* pair of models, while for (weak) undoability, it must be a pair of *consistent* models.

Bx formalisms We will mention, and give an example using, the OMG bidirectional transformation language QVT-R [OMG15]. For purposes of this paper what the reader needs to know is that a QVT-R *transformation* encodes, in one artefact, a consistency relation and forward and backward restorers. The model sets on which such a transformation operates are defined using metamodels in MOF. A transformation is structured as a number of *relations*, which may be related using when and where clauses. A relation may be thought of as capturing consistency in a certain local sense, while combining relations with when and where clauses allows the expression of global constraints.

Triple Graph Grammars, or TGGs, are another way to express bx, this time on typed attributed graphs. A TGG is a collection of *rules*; each rule captures consistency in a certain local sense, while the way in which the rules combine allows the expression of global consistency properties. A triple is a pair of (typed attributed) graphs, together with a *correspondence graph* which links elements of the two graphs. An *integrated triple* is such a triple that can be derived using the rules of the TGG. A pair of graphs is consistent according to a TGG if there exists an integrated triple comprising that pair of graphs, together with some correspondence graph. To restore consistency, say by replacing model n in an inconsistent pair (m, n) by n^+ that should be consistent with m, the tool essentially constructs a derivation tree that, starting from an initial state or axiom, first derives an integrated triple $m^- \leftarrow t \rightarrow n$ for some hypothetical model m^- (typically a submodel of m) and correspondence

graph t, then goes on, by applying further TGG rules, to reach an integrated triple $m \leftarrow t^+ \rightarrow n^+$ (where t^+, n^+ typically have t, n as submodels). In practice, unique parse constraints are usually imposed on the TGG so that this process is feasible, and the correspondence graphs are used to guide it.

Each of these formalisms is too complex for a complete explanation to be given here. We refer readers wishing to know more to [OMG15, SK08].

Analysis We will use a few basic notions of mathematical analysis (see e.g. [Sut75]), such as

Definition A *metric* on a set X is a function $d : X \times X \rightarrow \mathbb{R}$ such that for all $x, y, z \in X$:

1. $d(x, y) \geq 0$
2. $d(x, y) = 0 \Leftrightarrow x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, y) + d(y, z) \geq d(x, z)$

Further definitions will be given at the point of use for ease of reference.

3 A first attempt: measuring changes

In this section we will formally define one approach which is natural, particularly in the pure state-based setting, and explain why we think it is does not, alone, solve the problem.

3.1 Definition of metric least change

Assume we are given a metric on each model space. Suppose a consistency restorer is replacing an old model n by a new model, $\overrightarrow{R}(m, n)$ (taking m into account, as usual). Let us require that the distance between n and the chosen $\overrightarrow{R}(m, n)$ is minimal among all distances between n and any possible n' that restores consistency. That is, ensure that the effect on the model is as small as it can possibly be, given that consistency must be restored: if this is the case, one argues, it is pointless to insist on more. However, the approach has some limitations, such as failure to compose (essentially because not all triangles are degenerate).

Formally, for relational bx we may define:

Definition A bx $R : M \leftrightarrow N$ is *metric-least*, with respect to given metrics d_M, d_N on M and N, if for all $m \in M$ and for all $n, n' \in N$, we have

$$R(m, n') \;\Rightarrow\; d_N(n, n') \geq d_N(n, \overrightarrow{R}(m, n))$$

and dually.

An instance of this approach has been explored and implemented by Macedo and Cunha [MC16]. They take as given a pair of models and a QVT-R transformation; the QVT-R transformation is used only to specify consistency, the metric-based consistency restoration explored here being used as a drop-in replacement for the standard QVT-R consistency restoration. The models and consistency relation are translated

into Alloy, and the tool searches for the nearest consistent model. Their metric is "graph edit distance"; they also briefly considered allowing the user to define their own notion of edits ("operation-based distance"), which amounts to defining their own metric on the space of models.

3.2 Problem: lack of canonical metric

This approach is, of course, very sensitive to the specific metric chosen[2], and, because it operates after a model has been translated, the graph edit distance used in [MC16] is not a particularly good match for any user's intuitive idea of distance. There may not be a canonical choice, because different tools for editing models in the same modelling language provide different capabilities. We illustrate this with a simple example.

Example 3.1 (notQuiteTrivial) Let $M = \mathbb{B}$ and $N = \mathbb{B} \times \mathbb{B}$ be related by saying that $m \in M$ is consistent with $n = (s, t) \in N$ iff $m = s$. Otherwise, to restore consistency by changing n, we must flip its first component. We have a choice about whether or not also to flip its second component.

Expressed like that, the example suggests that flipping the second component as well as the first will violate any reasonable Least Change Principle. But this is because the wording has suggested that flipping both components is a larger change than flipping just one. It is possible to imagine situations in which this might not be the case. If n represents the presence or absence of a pebble in each of two pots, and it is possible to create or destroy pebbles, then moving a pebble from one pot to the other might be considered a small change, while creating/destroying a pebble might be considered a large change. In that case, with $m = \top$ and $n = (\bot, \top)$, modifying n to $(\top, \bot)$ might indeed be considered better than modifying it to $(\top, \top)$. This could be captured in a variety of ways, e.g. by a suitable choice of metric on N.

If my tool provides a simple menu item to make a change that, in your tool, requires many manual steps (e.g. mine has a menu of complex refactorings), is that change small or large? Relative to a specific tool, one can imagine defining a metric by something like "minimal number of clicks and keystrokes to achieve the change", but this is not satisfying when models are likely to be UML models or programs, with hundreds of different available editors. If we must settle on a metric that should be intuitive, perhaps symmetric difference on the sets of model elements will prove better.

3.3 Problem: metric least change is too constrained

Metric-leastness is a property which a given bx may or may not satisfy: it does not generally give enough information to *define* deterministic consistency restoration behaviour, because of the possibility that there may be many models at equal distance. Nevertheless, it is a very constraining property. Once the metrics are fixed, the bx developer has no power to define preferred behaviour of the consistency restorers, other than this choice of how to resolve non-deterministic choices between equidistant consistent models.

Here is an MDD-inspired example illustrating that we may not always want the consistent model which is intuitively closest.

[2] intuitively, more so than the continuity-based approach we shall see shortly, because there is less scope for varying the metric itself and the behaviour of the bx separately

Example 3.2 (ModelTests) Suppose bx R relates UML model m with test suite n, saying that they are consistent provided that every class in m stereotyped $\langle\langle$persistent$\rangle\rangle$ has a test class of the same name in n, containing an appropriate (in some specified sense) set of tests for each public operation, but n may also contain other tests. You modify the test class for a $\langle\langle$persistent$\rangle\rangle$ class C, to reflect changes made in the code to the signatures of C's methods, e.g., say `int` has changed to `long` throughout. R now propagates necessary changes to the model m. You probably expect R to perform appropriate changes to the detail of persistent class C in the model, changing `int` to `long` in the signatures of its operations. However, a different way to restore consistency would be to remove the stereotype from C, so that there would no longer be any consistency requirements relating to C; this probably involves a shorter edit distance, but is not what is wanted.

We think that for a metric-based bx to give comprehensible results, it must use metrics that are close to the developer's intuitions concerning how close together models are; although one can attempt to get round problems like this by defining special metrics on the model spaces that give the desired results, this does not seem promising.

As remarked, one still needs a way to resolve non-determinism; it is unlikely that there will be a unique closest consistent model. In [MC16], the tool offered to the user all consistent models found at the same minimum distance from the current model. In any practical version of such an approach, it would have to be possible for the bx programmer to define the choice of models if they wished, for usability reasons. The implementation is very resource intensive and not practical for non-toy cases (even, we understand, when there is only one choice of closest consistent model), because of the need to consider all models at a given distance from the current one.

3.4 Problem: complexity of metric-least least change

We might reasonably wonder whether the impracticality of this approach might be illusory; the implementation described in [MC16] was after all a research prototype, not the result of careful performance investigations. However, we can show NP-hardness of a reasonable formalisation of a subproblem of the one we would have to be addressing to use this approach in general, which is cause for pessimism (although of course, one must remember that the practical performance of tools depends on more than worst-case complexity).

To do this we apply a result from Buneman, Khanna and Tan's seminal paper [BKT02] addressing the closely-related *minimal view update problem* in databases. They showed a number of NP-hardness results even in very restricted settings. For example, where S is a database and $Q(S)$ a view of it defined by a query (even a project-join query of constant size involving only two relations), they showed that it is NP-hard to decide, given a tuple $t \in Q(S)$, whether there exists a set $T \subseteq S$ of tuples such that $Q(S \setminus T) = Q(S) \setminus \{t\}$. Their result that is useful to us here is a slight variant: they also showed that it is NP-hard to determine the smallest T such that $Q(S \setminus T) \subseteq Q(S) \setminus \{t\}$ (strictly speaking, of course, the decision form of the problem is whether there exists a T below a certain size). We reduce the problem of finding metric-least consistency restoration functions to this problem. First let's be clear what we're proving the non-existence of.

A good algorithm for embedding in a tool based on the metric-least approach to least change would be an algorithm which we could instantiate at a pair of model

sets M and N with their metrics d_M and d_N, and at a fixed consistency relation $R \subseteq M \times N$. The instantiated algorithm would take models $m \in M$ and $n \in N$, and a boolean to say which should be modified, and would work in time polynomial in the sizes of m and n to find an m' such that $R(m', n)$ (or dually, an n' such that $R(m, n')$, depending on the boolean). It would thus act as the pair of consistency restoration functions completing a correct and hippocratic bx between M and N. It would offer the least-change guarantee, i.e. that no m'' exists, satisfying $R(m'', n)$, that is closer to m than the m' it computes (and dually). Of course the permissible kinds of models, metrics and consistency relations that the tool could handle are variable, and the more flexibility offered here the more valuable the tool would be.

Suppose we had such an algorithm, whose scope included at least model sets such that models are sets of tuples, with the standard metric given by the size of the symmetric difference of these sets, $d(m, m') = \|m \triangle m'\|$, and consistency relations that can be defined using project-join SQL queries, e.g. $R_Q(m, n)$ should hold iff $Q(m) \subseteq n$. A key property of this class of SQL queries is that they are monotonic: $S' \subseteq S \Rightarrow Q(S') \subseteq Q(S)$.

Given an instance of the minimal source deletion problem defined by Q, S and $t \in Q(S)$, define $R_Q(m, n)$ to hold iff $Q(m) \subseteq n$, and apply our hypothetical least-change algorithm to evaluate $\overleftarrow{R_Q}(S, Q(S) \setminus \{t\})$, finding (by definition of R_Q) an S' such that $Q(S') \subseteq Q(S) \setminus \{t\}$. Now, this S' is by hypothesis closer (in the symmetric difference metric) to S than any other S'' with that property. Take $T = S \setminus S'$, so $\|T\| \leq d(S, S')$ and $S \setminus T \subseteq S'$ (even though equality might not hold, because our least-change algorithm might in fact have "cunningly" added things to S to get S', not just deleted things as we rather expect). So since the query Q is monotonic, $Q(S \setminus T) \subseteq Q(S')$ and by correctness of our algorithm $Q(S') \subseteq Q(S) \setminus \{t\}$. Thus T is a candidate solution to the minimal update problem. Is it a minimal solution? Yes: for if T' were smaller, then $S'' = S \setminus T'$ would be strictly closer to S than S' is $(d(S, S'') = \|T'\| < \|T\| \leq d(S, S'))$, and would also be consistent with $Q(S) \setminus \{t\}$, so our hypothetical polynomial least-change algorithm would not have erroneously given us S' as the least-change restoration. This reduction combines with the NP hardness result Theorem 2.5 of [BKT02] to show that this hypothetical algorithm cannot exist. We note also that [BKT02] also rules out efficiently approximating solutions to the problem, though it is unclear that approximate solutions would be of interest in the bx setting anyway.

Referring to the above discussion for the expressivity assumptions in force, we may summarise

Theorem 3.3 *Computing metric-least consistency restoration is NP-hard.*

4 Beyond changes to models: structure to which disruption might matter

How should one change to a model be judged larger than another? Typically it is not hard to come up with a naive notion based on the presentation of the model, but this can mislead. We can sometimes explain the phenomenon that "similarly sized" changes to a model are not equally disruptive, by identifying (making explicit in our formalism) auxiliary structure which changes more in one case than the other: the reason one change feels more disruptive than another is because it disrupts the

auxiliary structure more, even if the disruption to the actual model is no larger. In the database setting, Hegner's information ordering [Heg11] makes explicit *what is true about* a value. A change to a model which changes the truth value of more propositions is considered larger. Hegner's auxiliary structure does not actually add information: it can be calculated from the model. However, his setting had an established notion of change to a model (sets of tuples added and dropped, with supersets being larger changes). Thus, although we might wonder about redefining the "size" of changes so that changes that changed more truth values would be considered larger, that would not have been an attractive option in his setting. In MDD the idea of preferring changes that make less difference to what the developer thinks they know is attractive, but things are (as always!) more blurred: what is known, let alone knowable, about part of a model may not be deducible from the model, and there is no commonly agreed notion of change. Perhaps future bx language developers should explore, for example, weighting classes by how often their names appear in documentation, and/or allowing the bx user to assign weights, and using the weights in deciding on changes.

Witness structures Another major class of auxiliary structure to which disruption may matter, which definitely does involve information outside the model itself, is witness structures: structures whose intention is to capture something about the relationship between the models being kept consistent. A witness structure witnessing the consistency of two models m and n is an auxiliary structure that may help to demonstrate that consistency. In the simplest setting of relational bx, the unique witness structure relating m and n is just a point, and the witness structure in fact carries no extra information beyond the consistency relation itself ("they're consistent because I say so"). In TGGs, the derivation tree, or the correspondence graph, may be seen as witness structures ("they're consistent because this derivation shows how they are built up together using the TGG rules"). In an MDD setting, a witness structure may be a set of traceability links that helps demonstrate the consistency by identifying parts of one model that go with parts of another ("they're consistent because this part of this links to that part of that"). In [Ste13] some subtle issues were discussed concerning whether the links that demonstrate consistency embodied in QVT transformations should be followable in only one direction or both, and the paper also presented a game whose winning strategies can be seen as richer witness structures ("they're consistent because here's how Verifier wins a consistency game on them"). A witness structure could even be a proof ("they're consistent because Coq gave me this proof that they are").

The uses to which witness structures are put varies between different settings. They may or may not be used to guide consistency restoration, for example, by specifying alignment. Sometimes they are seen as explanatory devices, as in traditional traceability; sometimes they are seen just as a means to optimise the performance of repeated transformation application by avoiding recomputation. In QVT-R, for example, trace links are assumed to exist but their presence has no influence on the semantics of the language. See [WvP10] for a discussion of the ways in which traceability links are used in model-driven development, and [DEPC16] for a recent example of their use to guide consistency restoration in bx. We think this topic will be of increasing importance in bx.

Thus many types of witness structures can exist, and even within a type, different witness structures might witness the consistency of the same pair of models. (A dependently typed treatment is outside the scope of this paper, but is work in progress [McK16].) The extent to which the witness structure is explicit in the minds of the

developers who work with the models probably varies greatly – indeed, future bx language designers should consider the implications of this (compare: considering the user's mental model, in UI design). Care will be needed to ensure practical usability of change size comparisons based on witness structures, but at least there is some hope that we can do better with than without them.

Fortunately, for our purposes, it suffices to observe that much of what we say in a relational setting can be "lifted" to a setting with auxiliary structures; the developer's modification is still to a model, but the bx may in response modify not just the other model but also the auxiliary structure, and when we compare or measure changes we may be taking into account the changes to the auxiliary structure. We leave the rest of this fascinating topic for future work.

5 Beyond least change: weak and strong least surprise

In this section we consider more carefully the way in which varying the domain of quantification tunes properties of bx from strong to weak versions, touched on in Section 2. This turns out to be important for least surprise principles – particularly, but not only, those we shall consider in Section 7.

Properties of bx, such as the strong and weak undoability properties that we met earlier, typically involve predicates on model pairs (m, n) that are defined in terms of the components of the bx R. Formally, such properties involve *families* of predicates that are defined *uniformly* for all bx between given model spaces, and moreover in dual pairs, one version for $\overrightarrow{R}$, one for $\overleftarrow{R}$. We stay relatively informal here.

We may ask whether such predicates hold of all the pairs (m, n) in some interesting subset of $M \times N$, such as the whole set or the set of consistent pairs.

Definition Let P be a family $\{P_R \mid R : M \leftrightarrow N\}$ of predicates on pairs (m, n) of models, defined (uniformly) in the components of bx R, including $\overrightarrow{R}$. Then for a particular bx R, we say $\overrightarrow{R}$ is *strongly* P if for all $m \in M$, $n \in N$ we have $P_R(m, n)$. It is *weakly* P if for all $m \in M$, $n \in N$ we have $R(m, n) \Rightarrow P_R(m, n)$.

Where the dual property (involving $\overleftarrow{R}$) also holds on $M \times N$ (rsp. on $\{(m, n) : R(m, n)\}$), we will say the bx R is strongly (rsp. weakly) P.

The reader may like to try this with properties they know already. For example, "weak correctness" turns out to be vacuously true, while "weak hippocraticness" is the same as (strong) hippocraticness. We have already discussed weak and strong undoability. The property of being metric-least, defined in Section 3, also fits this scheme. We most naturally cast it as a *strong* property (note that although there is a consistency check to the left of an implication, it is a check that $R(m, n')$ holds, rather than that $R(m, n)$ does). Then the weak variant of metric leastness holds vacuously, because if $R(m, n)$ then the second application of the distance function yields 0 by hippocraticness.

We illustrate the practical scenarios in which this distinction is important by using a class of least surprise principle that we shall be considering more formally in Section 7. Here $P_R(m, n)$ means $\overrightarrow{R}(\_, n)$ is continuous at m; but we will explain it informally.

Suppose there are two users, Anne working with models from set A and Bob working with model set B. Suppose Anne has made a change she regards as small. We would like to be able to guarantee, by restricting to bx with a certain good

property, that the *difference* this change makes to Bob is small. The intuition is that Anne is likely to consider a large change much more carefully than a small change. We do not wish a not-very-considered choice by Anne (e.g. adding a comment) to have a large unintended consequence for Bob. (If Bob changes his model and Anne's must be modified to restore consistency, everything is dual.)

Weak least surprise guarantees that the change from b to $b' = \overrightarrow{R}(a', b)$ is small, *provided that a and b are consistent* and that the change from a to a' is small. That is, it supports the following scenario:

- models a and b are currently consistent
- now Anne modifies her model, a, by a small change, giving a'
- then we restore consistency, taking Anne's model as authoritative; that is, we calculate $b' = \overrightarrow{R}(a', b)$ and hand it to Bob to replace his current model, b
- Bob's model has suffered a small change, from b to b'.

That is, the guarantee we offer operates under the assumption that Bob's model hasn't changed, since it was last consistent with Anne's, until we change it. If Bob continues to work on it in the meantime, all bets are off. We guarantee nothing if the two models have been edited concurrently.

Strong least surprise imposes a more stringent condition on the bx and offers a correspondingly stronger guarantee to the bx users. It allows that Bob may be working concurrently with Anne; his current model, b, might, therefore, not be consistent with Anne's current model a. Here the guarantee we want is that, provided the change Anne makes from a to a' is small, the choice of whether to restore consistency before or after making this small change will make only a small difference to Bob, regardless of what state his model is in at the time. It supports the scenario:

- we don't know what state Bob's model is in with respect to Anne's – both may be changing simultaneously
- Anne thinks about changing hers a little, and isn't sure whether or not to do so (e.g., she dithers between a and a' which are separated by a small change)
- the difference between
  - the effect on Bob's model if Anne does decide to make her small change and
  - the effect on Bob's model if she doesn't,

  is small, regardless of what state Bob's model is in when we do restore consistency. In the state-based relational setting, this amounts to saying that *for any* b the change from $\overrightarrow{R}(a, b)$ to $\overrightarrow{R}(a', b)$ can be guaranteed to be small, provided the change from a to a' is small. Note that both $\overrightarrow{R}(a, b)$ and $\overrightarrow{R}(a', b)$ might be a "large change" from b; to demand otherwise would be unreasonable, as a and b, being arbitrary, might be very seriously inconsistent. We can ask only that these results are a small change from one another.

Weak least surprise is a weakening of strong least surprise because by hippocraticness, if $R(a, b)$ then $\overrightarrow{R}(a, b) = b$ so we get the weak definition by instantiating the strong definition only at models b that are consistent with a.

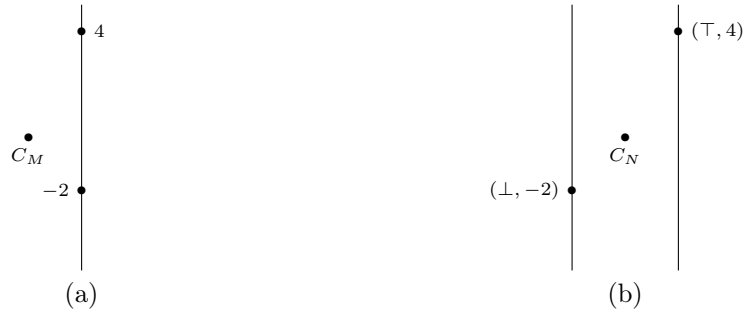

Figure 1 – (a) Model space M, (b) model space N for Example 5.1

Strong and weak least change vary in the condition that is placed on the initial state of the world, before we will know that a small change on one side will cause only a small change on the other: does the condition hold for all $(a, b) \in A \times B$, or only for $(a, b) \in C \subseteq A \times B$? In the weak variant C is the consistent pairs. Alternatively, we could take $C = \{(a, b) : b \in N(a)\}$ where for any a, $N(a)$ is the models that are, maybe not consistent with, but at least reasonably sensible with respect to, a. Another promising option is $C = M \times N$ for a subspace pair [Ste14] (M, N) in (A, B). A subspace pair is a place where the developers working with both models can, jointly, agree to stay, in the sense that if they do not move their model outside the subspace pair, the bx will not do so when it restores consistency. Imposing least surprise would guarantee, additionally, that small changes on one side lead to small changes on the other. These variants should repay further study, particularly in that identifying "good" parts of the pair of model spaces might be possible even if inevitably the bx must have bad behaviour elsewhere. We finish this section with an example which illustrates this. We shall return to the topic of "good" subspace pairs in Section 8.

Example 5.1 (weakVsStrongContinuity) Let $M = \mathbb{R} \cup \{C_M\}$ and $N = (\mathbb{B} \times \mathbb{R}) \cup \{C_N\}$. We use the consistency relation: $R(C_M, C_N)$ and $R(m, (\_, m))$ for all $m \neq C_M$. Figure 1 illustrates. We give M the metric generated by $d_M(C_M, m) = 1 + |m|$ for all $m \in \mathbb{R}$, together with the usual metric $d_M(m, m') = |m - m'|$ on $\mathbb{R}$. Give N the metric generated by the usual metric on both copies of $\mathbb{R}$, together with: $d_N(C_N, (\_, n)) = 1 + |n|$ for all $n \in \mathbb{R}$, and $d_N((\top, n_1), (\bot, n_2)) = 2 + |n_1| + |n_2|$.

Now, because there is a unique element of M consistent with any given element $n \in N$, there is no choice for $\overleftarrow{R}(m, n)$ given that this must be correct: it must ignore m and return that unique consistent choice. Let us consider the choices we have for the behaviour of $\overrightarrow{R}(m, n)$, and the extent to which each is reasonable from a least change point of view.

- If m is C_M, there is no choice: we must return C_N to be correct.
- If $m = x_m$ and $n = (b, x_n)$ are both drawn from the copies of $\mathbb{R}$, we must return (b', x_m) for some $b' \in \mathbb{B}$. If in fact $x_m = x_n$, hippocraticness tells us we must return exactly n. Otherwise, we could legally return a result which is on the other branch from n, that is, flip the boolean as well as changing the real number. It is easy to argue, though, that to do so violates any reasonable least change principle, since the boolean-flipping choice gives us a change in the model which is larger by our chosen metric, with no obvious prospect for any compensating advantage. Let us suppose we agree not to do so.

- The interesting case is $\overrightarrow{R}(x_m, C_N)$ for real x_m. We must return either $(\top, x_m)$ or $(\bot, x_m)$; neither correctness nor hippocraticness place any further restrictions, and when we look at one restoration scenario in isolation, our metric does not help us make the choice, either. We could, for example:

  1. pick a boolean value and always use that, e.g. $\overrightarrow{R}(x_m, C_N) = (\top, x_m)$ for all $x_m \in \mathbb{R}$;
  2. return $(\top, x_m)$ if $x_m \geq 0$, $(\bot, x_m)$ otherwise; or we could even go for bizarre behaviour such as
  3. return $(\top, x_m)$ if x_m is rational, $(\bot, x_m)$ otherwise.

All of these options for $\overrightarrow{R}(x_m, C_N)$ will turn R into a correct and hippocratic bx. It seems intuitive that these are in decreasing order of merit from a strong least surprise point of view. Imagine that the developers on the M side were not quite sure whether they wanted to put one real number, or another very close to it. The danger that their choice on this matter has determined which branch the N model ends up on, "merely because" the state of the N model at the moment they chose to synchronise "happened" to be C_N, increases as we go down the list of options.

From the point of view of weak least surprise, however, there is no difference between the options, at least for a sufficiently small notion of "small change". For, if $R(m, n)$ holds, and then m is changed to m' by a change that has size greater than 0 but less than 1, it follows that neither m nor n can be the special points C_M and C_N: there are no other models close to C_M, so m has to be one of the real number points, say $x_m \in \mathbb{R}$, m' has to be a nearby real number $x_{m'}$, and from consistency it follows that n must be (b, x_m) for some b. We have already agreed that the result of $\overrightarrow{R}(m', n)$ should be $(b, x_{m'})$.

The key point here is that only in the first option is $\overrightarrow{R}(\_, C_N) : M \to N$ a continuous function in the usual sense of mathematics; in the middle option this function has a discontinuity, while in the final option it is discontinuous everywhere except C_M. This motivates our considerations of continuity in Section 7. Note that not only is $(\{C_M\}, \{C_N\})$ a subspace pair on which any of these variants is continuous (trivially, any pair of consistent states always forms a subspace pair), so is $(M \setminus \{C_M\}, N \setminus \{C_N\})$. This illustrates the potential for a future bx tool to use subspace pairs to warn developers of discontinuous behaviour.

6 Ordering changes

If we wish to identify changes that are defensibly "least", the most basic thing we can do is to identify the possible changes that could restore consistency, place a partial order on these changes, and insist that the chosen change be minimal. Of course this does not solve anything, and any solution to our problem can be cast in this setting.

Meertens [Mee98] requires structure stronger than this, but weaker than a metric on the model sets. For any model $m \in M$ he assumes given a reflexive and transitive relation on M, notated $x \sqsubseteq_m y$ and read "x is at least as close to m as y is", satisfying the property that $m \sqsubseteq_m x$ for any m, x. If M is a metric space of course we derive this relation from the metric; most of Meertens' examples do actually use a metric, typically size of symmetric difference of sets. He takes it as axiomatic that consistency restoration should give a closest consistent model (according to the preorder), and that it should do so deterministically; much of his paper is devoted to showing how

to calculate systematically a *biased selector* that does this job. Here is an example which illustrates this, adapted from Example 5.2d of [Mee98].

Example 6.1 (meertensIntersect) Let $M = \mathcal{P}(\mathbb{N}) \times \mathcal{P}(\mathbb{N})$, let $N = \mathcal{P}(\mathbb{N})$, and let $m = (s, t)$ be consistent with n iff $s \cap t = n$. Of course $\overrightarrow{R}((s, t), n)$ has no choice; it must ignore n and return $s \cap t$. $\overleftarrow{R}((s, t), n)$ must:

1. add to both s and t any element of n that is not already present;
2. preserve in both s and t any element of n that is already present;
3. delete from *at least one of* s and t any element of their intersection that is not in n.

As far as correctness goes, it is also at liberty to add any element that is not in n to just one of s, t, provided it is not already present in the other. But intuitively this would be unnecessarily disruptive (in the extreme case where the arguments are already consistent it will violate hippocraticness).

If we take as distance between (s, t) and (s', t') the sum of the sizes of the symmetric differences of the components, we have the language in which to express that that behaviour would give a larger change than necessary. Similarly, we justify that in case 3 above, the offending element should be removed from just one set, not both. The choice is arbitrary; Meertens uses the concept of bias to explain how it is resolved by the bx language or programmer. Given one choice of resolution, his calculations produce the result that $\overleftarrow{R}((s, t), n) = (n \cup (s \setminus t), n \cup t)$.

Using a similar structure, Macedo et al. [MPCO13] address the tricky question of when it is possible to compose bx that satisfy such a least change condition. As might be expected, they have to abandon determinacy (so that the composition can choose "the right path" through the middle model), and impose stringent additional conditions; fundamentally, there is no reason why we would expect bx that satisfy this kind of least change principle to compose.

6.1 Changes as sets of small changes

A preorder on changes is useful where changes are identified with sets of discrete elements to be added to/deleted from a structure; this is usually taken to be the case for databases. We generate a preorder on changes from the inclusion ordering on sets. This lets us prefer changes that do not add or delete elements unnecessarily. A drawback is that if an element is modified, even very slightly, we must model this as a deletion of one thing and an addition of something very similar. Then this change appears bigger than it "really" is.

Here are two examples. Example 6.2 is adapted from [Heg11], where it is presented in a database context. It demonstrates that it is not obvious whether to consider it more disruptive to reuse an existing element of a model, or to introduce a new "freely added" element. Example 6.3 is a cautionary tale on how modelling modifications as deletions followed by additions can produce poor results, where models are structured collections of elements, not just sets.

Example 6.2 (hegnerInformationOrdering) Let $M = \mathcal{P}(A \times B \times C)$ for some sets A, B, C, and let $N = \mathcal{P}(A \times B)$, with the consistency relation that $m \in M$ is only consistent with its projection. For example, $m = \{(a_0, b_0, c_0), (a_1, b_1, c_1)\}$ is consistent

with $n = \{(a_0, b_0), (a_1, b_1)\}$, but not with $n' = \{(a_0, b_0), (a_1, b_1), (a_2, b_2)\}$. If m is to be altered so as to restore consistency with n', (at least) some triple (a_2, b_2, c) must be added. But what should the value of c be? One may argue that it is better to reuse an already-present element of C, adding (a_2, b_2, c_0) or (a_2, b_2, c_1); or one may argue as Hegner does in [Heg11] that it is better to use a previously unseen element c_2.

It is reasonably intuitive that just one triple should be added (for example, we should not take advantage of the licence to change m in order to add the legal but unhelpful triple (a_0, b_0, c)); but in certain circumstances, considerations such as those in Example 3.1 may bring even this into doubt.

The same issue arises in our next example, which is adapted from [BS12]; it also illustrates the current behaviour of QVT-R in the OMG standard semantics. Interestingly in this case the strategy of creating new elements rather than reusing old ones is far less convincing.

Example 6.3 (qvtrPreferringCreationToReuse) Let M and N be identical model sets, comprising models that contain two kinds of elements `Parent` and `Child`. Both kinds have a string attribute `name`; `Parent` elements can also be linked to children which are of type `Child`. We represent such models in the obvious way as forests, and notate them using the `name`s of elements, e.g. $\{p \to \{c_1, c_2, c_2\}\}$ represents a model containing one element of type `Parent` with `name` p, having three children of type `Child` whose `name`s are c_1,c_2,c_2. Notice that we do not forbid two model elements having the same `name`.

The consistency relation between $m \in M$ and $n \in N$ we consider is given by a pair of unidirectional checks. m and n are consistent "in the direction of n" iff for any `Parent` element in m, say with `name` t, linked to a child having `name` c, there exists a(t least one) `Parent` element with `name` t in n, also linked to a child with `name` c. The check in the direction of m is dual.

This is expressed in QVT-R as follows (the "enforce" specifications will allow us later to use the same transformation to enforce consistency, but any QVT-R transformation can be run in "check only mode" to check whether given models are consistent):

```
transformation T (m : M ; n : N) {
top relation R {
  s : String;
  firstchild : M::Child;
  secondchild : N::Child;
  enforce domain m me1:Parent {name = s, child = firstchild};
  enforce domain n me2:Parent {name = s, child = secondchild};
  where { S(firstchild,secondchild); }}

relation S {
  s : String;
  enforce domain m me1:Child {name = s};
  enforce domain n me2:Child {name = s};}}
```

Suppose that (for some strings p,c_1,c_2,c_3) we take $m = \{p \to \{c_1, c_2, c_3\}\}$, and for n we take the empty model. When we restore consistency by modifying n, what are reasonable results, from the point of view of least change and considering only the consistency specification?

We might certainly argue that n should be replaced by a copy of m; intuitively, m is a good candidate for the least complex thing that is consistent with m in this case. What the QVT-R specification, and its most faithful implementation, ModelMorf, actually produces is $\{p \to \{c_1\}, p \to \{c_2\}, p \to \{c_3\}\}$.

We refer the reader to [BS12] for details of the semantics that gives these results, but in brief: QVT-R only *changes* model elements when it is forced to do so by "key" constraints. That is, when a certain kind of element is required, and one exists but with the wrong properties, and more than one is not allowed, then QVT-R changes the element's properties. Otherwise, QVT-R modifies a model in two phases. First it adds model elements that are required for consistency to hold. Because it does not change properties of model elements, which include their links (`Parent` to `Child` in our case), it takes an "all or nothing" view of whether an entire valid binding, that is, configuration of model elements that is needed according to a relation, exists. If not, it creates elements for the whole configuration.

So here, the transformation might first discover that it needs a `Parent` named p linked to a `Child` named c_1; since there isn't such a configuration, it creates both elements. Next, it discovers that it needs a `Parent` named p linked to a `Child` named c_2; since such a configuration does not exist, it creates both a new `Parent` and a new `Child`. (We could have written the QVT-R transformation differently, e.g. used a "key" constraint on `Parent`, but this would have other effects that might not be desired.) Similarly for c_3.

Next, with the same m and a further string x, consider $n = \{p \to \{c_1, c_2, x\}\}$. Intuitively, the name of the third `Child` is wrong: it is x and should be changed to c_3. In fact, QVT-R and ModelMorf, using the same transformation as before, actually produce $\{p \to \{c_1, c_2\}, p \to \{c_3\}\}$. As in the previous example, rather than modify an existing `Child`, a whole new binding $\{p \to \{c_3\}\}$ has been created to be the match to the otherwise unmatchable binding involving c_3 in m. In the second phase, the `Child` with `name` x has been deleted, as otherwise the check in the direction of m could not have succeeded.

A final example in [BS12], omitted here, demonstrates that the question of precisely when elements need to be deleted is problematic. Altogether, modelling modifications as additions and deletions is delicate. Even if the end result of a modification is the same as the end result of deleting one thing and adding another, it is simply not the case that modifying an element is typically more surprising than deleting it, as suggested by that representation of what happens.

A similar approach is used by TGGs when used in an incremental change scenario; [LAS+14] explains and compares several TGG tools from the point of view of various properties including "least change". In the context of a set of triple graph grammar rules, we suppose given: a derivation of an *integrated triple* $M_S \leftarrow M_C \rightarrow M_T$; that is, a pair of consistent models M_S and M_T together with a correspondence graph M_C; a change Δ_S to M_S. The task is to produce a corresponding change Δ_T to M_T, updating the auxiliary structures appropriately. What the deltas can be is not formally defined in [LAS+14] but their property (p7)

> Least change (F4): An incremental update must choose a Δ_T to restore consistency such that there is no subset of Δ_T that would also restore consistency, i.e., the computed Δ_T does not contain redundant modifications.

makes the assumption clear. Notice that this property is formulated in a *weak* version: the model to be changed is assumed consistent with a previous version of the model

that has been changed. The issues of handling changes other than as additions plus deletions (mentioned above) and of avoiding creating new elements where old ones could instead be reused (cf Example 6.3 and Example 6.2), are mentioned. Two tools (MoTE and TGG Interpreter) are said to "provide a sufficient means to attain [least change] in practical scenarios" but we are aware of no formal guarantee.

7 Continuity and other forms of structure preservation

In the metric-least least change approach, the only "lever" to pull (apart from how to resolve the choice between equally close consistent models) was the choice of metric; committing to that approach tends to require, as discussed, that a tool allow its users to define the metrics that should be in use, perhaps even on a per-bx basis. Although this does give some extra expressivity, it is not very satisfactory, as we discussed. The notions we discuss in this section still rely on a choice of metric, but because, even for fixed metrics, bx developers will have more freedom of choice in how to define the consistency restoration without losing the properties we discuss, we will not expect them to need to define their own metrics in order to get the bx behaviour they want. We would rather expect that a sensible metric for a given kind of model (e.g., the size of the symmetric difference between sets of model elements) would be fixed.

The most basic idea, and the most natural way to move on from the metrics-based approach, is continuity, in the following metric-based formulation. (The other setting in which continuity appears in undergraduate mathematics, namely topology, we leave as future work.) Informally, a map is continuous (at a source point) if, however close you want to get to your target, you can ensure you get that close by starting within a certain distance of your source. Formally

Definition $f : S \to T$ is continuous at s iff

$$\forall \epsilon > 0 \,.\, \exists \delta > 0 \,.\, \forall s' \,.\, d_S(s, s') < \delta \Rightarrow d_T(f(s), f(s')) < \epsilon$$

We say just "f is continuous" if it is continuous at all s.

Standard results [Sut75] apply: the identity function, and constant functions, are continuous (everywhere), the composition of continuous functions is continuous (at the appropriate points) etc.

To see how to adapt these notions to bx it will help to be more precise. In particular, metric-based continuity of a map is defined at a point: the idea that a map is continuous overall is a derived notion, defined by saying that it is continuous if it is continuous at every point. For us, the points are clearly going to be pairs of models. Supposing that we have metrics d_M, d_N on the model spaces M, N related by a relational bx R:

Definition $\overrightarrow{R}$ is continuous at (m, n) iff

$$\forall \epsilon > 0 \,.\, \exists \delta > 0 \,.\, \forall m' \,.\, d_M(m, m') < \delta \Rightarrow d_N(\overrightarrow{R}(m, n), \overrightarrow{R}(m', n)) < \epsilon$$

This is nothing other than the standard metrics-based continuity of $\overrightarrow{R}(\_, n) : M \to N$ at m. Dually, $\overleftarrow{R}$ is continuous at (m, n) iff $\overleftarrow{R}(m, \_) : N \to M$ is continuous at n.

Definition $\overrightarrow{R}$ is strongly continuous if it is continuous at all (m, n); that is, for every n, $\overrightarrow{R}(\_, n)$ is a continuous function. The definition for $\overleftarrow{R}$ is dual. We say a bx R is strongly continuous if its restorers $\overrightarrow{R}$, $\overleftarrow{R}$ are so.

The terminology "strongly continuous" is justified with respect to our earlier discussion of strong versus weak least surprise, because of the insistence that $\overrightarrow{R}(\_, n)$ is continuous at all m, regardless of whether m and n are consistent.

Definition $\overrightarrow{R}$ is weakly continuous if it is continuous at all *consistent* (m, n); that is, for every n, $\overrightarrow{R}(\_, n)$ is continuous at all points m such that $R(m, n)$ holds. The definition for $\overleftarrow{R}$ is dual. We say a bx R is weakly continuous if its restorers $\overrightarrow{R}$, $\overleftarrow{R}$ are so.

Just as discussed in Section 5, one could consider further variants in which $\overrightarrow{R}(\_, n)$ is required to be continuous at points m where (m, n) is in some other subset of $M \times N$.

Example 5.1 shows that weakly continuous really is weaker than strongly continuous. While all three of the options we considered for $\overrightarrow{R}(m, C_N)$ yield a weakly continuous $\overrightarrow{R}$, only the first gives strong continuity. However, history ignorance – that is, the property that $\overrightarrow{R}(m, \overrightarrow{R}(m', n)) = \overrightarrow{R}(m, n)$, and dually, for all values of m, m', n, generalising PutPut for lenses – makes weak and strong continuity coincide.

Lemma 7.1 *If $R : M \leftrightarrow N$ is history ignorant as well as correct and hippocratic, then $\overrightarrow{R}$ is strongly continuous if and only if it is weakly continuous. Dually this holds for $\overleftarrow{R}$ and hence for R.*

Proof Suppose R is weakly continuous and consider (m, n) not necessarily consistent. We are given ϵ and must find $\delta > 0$ such that

$$\forall m' \,.\, d_M(m, m') < \delta \Rightarrow d_N(\overrightarrow{R}(m, n), \overrightarrow{R}(m', n)) < \epsilon$$

Using the same ϵ, we apply weak continuity at $(m, \overrightarrow{R}(m, n))$ to find δ' such that

$$\forall m' \,.\, d_M(m, m') < \delta' \Rightarrow d_N(\overrightarrow{R}(m, \overrightarrow{R}(m, n)), \overrightarrow{R}(m', \overrightarrow{R}(m, n))) < \epsilon$$

Applying history ignorance, this implies the condition we had to satisfy, so we take $\delta = \delta'$. ∎

In contrast to the properties we have considered so far, which fail to compose, it is straightforward to prove:

Theorem 7.2 *Let $R : M \leftrightarrow N$ and $S : N \leftrightarrow P$ be strongly (respectively weakly) continuous bx which, as usual, are correct and hippocratic. Suppose further that R is lens-like, i.e., $\overrightarrow{R}$ ignores its second argument; we write $\overrightarrow{R}(m)$. It follows that $\overrightarrow{R}(m)$ is the unique $n \in N$ such that $R(m, n)$. Define the composition $R; S : M \leftrightarrow P$ as usual for lenses: $(R; S)(m, p)$ holds iff there exists $n \in N$ such that $R(m, n)$ and $S(n, p)$; $\overrightarrow{R; S}(m, p) = \overrightarrow{S}(\overrightarrow{R}(m), p)$; $\overleftarrow{R; S}(m, p) = \overleftarrow{R}(m, \overleftarrow{S}(\overrightarrow{R}(m), p))$. Then $R; S$ is also correct, hippocratic and strongly (respectively weakly) continuous.*

Less positively, continuity is not useful in discrete model spaces, such as those that arise in (non-idealised) model-driven development, because:

Lemma 7.3 *Suppose $m \in M$ is an* isolated point*, in the sense that for some real number $\Delta > 0$ there is no $m' \neq m \in M$ such that $d_M(m, m') < \Delta$. Then with respect to d_M, any $\overrightarrow{R}$ is continuous at (m, n), for every $n \in N$.*

This holds just because for any ϵ one may pick $\delta < \Delta$ and then the continuity condition holds vacuously.

As we would expect, metric-leastness is incomparable with continuity: they offer different kinds of guarantee. All three options in Example 5.1 are metric-least, so metric-leastness does not imply strong continuity. In fact it does not even imply weak continuity; the following example transformation is metric-least, but not weakly continuous at $(0, +)$.

Example 7.4 (plusMinus) Let M be the interval $[-1, +1] \subseteq \mathbb{R}$, and let $N = \{+, -\}$. We say $m \in M$ is consistent with $+$ if $m \geq 0$, and consistent with $-$ if $m \leq 0$. That is, the bx relates a real number with a record of whether it is (weakly) positive or negative.

Suppose $m < 0$. We have no choice to make about the behaviour of $\overrightarrow{R}(m, +)$: to be correct it must return $-$. On the other hand, $\overleftarrow{R}(m, +)$ could correctly return any non-negative m'. Based on the usual measurement of distance in M, and an intuitive idea of least change, it seems natural to suggest 0 as the choice of m', because it is closer to m than any other correct choice. Dual statements apply for $m > 0$ and $-$.

Two variants are of interest:

1. We change the consistency condition so that $m \in M$ is consistent with $+$ if $m \geq 0$, and consistent with $-$ if $m < 0$ (strictly). This gives a problem in deciding what the result of $\overleftarrow{R}(m, -)$ should be when $m > 0$, because 0 is no longer a correct result, and for any value returned, a "better" one could be found.

Observing that this is the result of the model space being non-discrete, we may also consider a second variant:

2. we replace M by a discrete set, say $\{x/100 : x \in \mathbb{Z}, -100 \leq x \leq 100\}$.

Conversely, Lemma 7.3 shows that even strong continuity does not imply metric-leastness; apply discrete metrics to M and N in Example 3.1, so that by Lemma 7.3 any variant of the bx discussed there is strongly continuous, and pick a variant that is not metric-least by the chosen metric.

7.1 Stronger variants of metric-based continuity

Given that continuity is unsatisfactory because in many of the cases we wish to cover it holds vacuously, a reasonable next step is to consider the standard panoply of strengthened variants of continuity. (Standardly, these definitions *do* imply continuity.) Will any of them be better for our purposes? Let S and T be metric spaces as before.

Definition $f : S \to T$ is uniformly continuous iff

$$\forall \epsilon > 0 \,.\, \exists \delta > 0 \,.\, \forall s, s' \,.\, d_S(s, s') < \delta \Rightarrow d_T(f(s), f(s')) < \epsilon$$

That is, in contrast to the standard continuity definition, δ depends only on ϵ, not on s.

This, adapted to bx, will obviously also be vacuous on discrete model spaces, so let us not pursue it.

Next we consider strengthening continuity by systematically bounding the amount of change in the result of a function, in terms of the amount of change in the argument

to the function. That is the motivation for Hölder continuity, which is stronger than the variants we have considered so far, but not as strong as differentiability.

Definition Given non-negative real constants C, α, we say $f : S \to T$ is Hölder continuous (with respect to C, α) at s iff

$$\forall s' \,.\, d_T(f(s), f(s')) \leq C d_S(s, s')^\alpha$$

We say that f is Hölder continuous if it is so at all s. The special case where $\alpha = 1$ is known as Lipschitz continuity.

Note that, to be congruent with our other definitions and for ease of adaptation, we have defined Hölder continuity first at a point, and then of a function. Since the definition is symmetric in s and s', it is not often presented that way. Adapting to bx as before by considering the Hölder continuity of $\overrightarrow{R}(\_, n) : M \to N$ at m, we get

Definition $\overrightarrow{R}$ is Hölder continuous (with respect to C, α) at (m, n) iff

$$\forall m' \,.\, d_N(\overrightarrow{R}(m, n), \overrightarrow{R}(m', n)) \leq C d_M(m, m')^\alpha$$

Then as before, we may say that R is strongly (C, α)-Hölder continuous if it is so at all (m, n), weakly (C, α)-Hölder continuous if it is so at consistent (m, n), and we may consider intermediate notions if we wish.

The fact that the adaptation to bx is symmetric in m and m' raises the question of whether strong Hölder continuity is actually stronger than weak Hölder continuity, however. In fact Example 5.1 was designed to demonstrate this: for example, variant 2 is easily seen to be weakly (1,1)-Hölder continuous, but is not strongly (1,1)-Hölder continuous because we can pick $n = C_N$ and m, m' to be real numbers which are arbitrarily close but on opposite sides of 0.

We get again the analogue of Lemma 7.1, by the same argument.

Lemma 7.5 *If $R : M \leftrightarrow N$ is history ignorant as well as correct and hippocratic, then R is strongly (C, α)-Hölder continuous if and only if it is weakly (C, α)-Hölder continuous.*

The next interesting question is whether Hölder continuity might avoid the problem we noted for continuity, viz. that it is trivially satisfied at isolated points. The answer is that it does. Here is an example to illustrate.

Example 7.6 (continuousNotHolderContinuous) Our model spaces are subsets of real space with the standard metric. Let $M \subseteq \mathbb{R}^2$ comprise the origin, which we label C_M, together with the unit circle centred on the origin, parameterised by θ running over the half-open interval $(0, 1]$. Let $N \subseteq \mathbb{R}$ be $\{0\} \cup [1, +\infty)$. We say that the origin C_M is consistent only with 0, while the point on the unit circle at parameter θ is consistent only with $1/\theta$.

Note that this example is boring from the point of view of consistency restoration, as consistency is a bijective relation here so there is no choice about how to restore consistency. The interest is the technical point: although $\overrightarrow{R}$ is continuous at every (m, n), it is not (C, α)-Hölder continuous at (C_M, n) for *any* $n \in N$ because, while the distance between C_M and any other m' is 1, the distance between $\overrightarrow{R}(C_M, n) = 0$ and $\overrightarrow{R}(m', n)$ can be made arbitrarily large by judicious choice of m'.

7.2 Choice of constants

Hölder continuity, unlike basic continuity and some other variants, is defined relative to constants (C, α). We have not yet discussed the implications of this; for example, should the same constants apply to $\overrightarrow{R}$ and to $\overleftarrow{R}$? Not necessarily: for example, if the metrics on M and N are intuitively "not on the same scale" we might well want different constants in the two directions.

It may already be interesting to say that *there exists some* (C, α) such that R is Hölder continuous with respect to those constants, but in some cases that will not be informative: if the model spaces are discrete and bounded in the sense that the distances between distinct models are bounded both below (by something > 0) and above, then there will always be *some* choice of (C, α) that will do the job. Even then, though, it might be interesting to know whether Hölder continuity held for some particular pair of constants, e.g. to judge how much work on one model should be allowed to happen before the task of a human validating consistency restoration becomes infeasible.

Thus it is possible that Hölder continuity might turn out to be a useful least change principle. We remark, though, that continuity is already a strong condition, Hölder continuity much stronger, so we should not usually expect it to hold globally. In the next section we explore how to combine the advantages of such a strong condition with the flexibility needed in practice. We will use Hölder continuity in our example, but future work might consider e.g. locally Lipschitz functions, or in a sufficiently special space, continuously differentiable functions, etc.

8 Atlases of subspace pairs: piecewise good behaviour

In this section we formalise the idea that a bx might have good, unsurprising behaviour on straightforward parts of the model spaces, while unavoidably causing surprise under well-defined circumstances. We give an example and discuss how usable tools might incorporate such ideas.

First let us recall the precise definition of a subspace pair.

Definition Let $R : M \leftrightarrow N$ be a correct and hippocratic bx, and suppose $M_1 \subseteq M$ and $N_1 \subseteq N$. Then (M_1, N_1) is a subspace pair in (M, N), with respect to R, if $\overrightarrow{R}(M_1, N_1) \subseteq N_1$ (that is, for any $m_1 \in M_1$ and $n_1 \in N_1$ we have $\overrightarrow{R}(m_1, n_1) \in N_1$) and dually $\overleftarrow{R}(M_1, N_1) \subseteq M_1$.

That is, the bx's consistency restoration functions will never be responsible for moving a pair of models outside the subspace pair. If the current pair of models (m, n) is in a certain subspace pair before consistency restoration (in either direction), it will still be in that subspace pair afterwards. This is the sense in which the developers working with the models can jointly agree to stay within the subspace pair. This also means that the restriction of the bx to the subspace pair is itself a well-defined, correct and hippocratic bx.

Now, as mentioned in [Ste14], by hippocraticness any consistent pair (m, n) forms a one-point subspace pair $\{(m, n)\}$; moreover, (M, N) itself is always a subspace pair. Thus any bx gives rise to various "atlases" of subspace pairs, in the following sense:

Definition Let R be as before. A collection $\{(M_i, N_i) : i \in I\}$ of subspace pairs in (M, N) is an atlas if $\cup_{i \in I} M_i = M$ and $\cup_{i \in I} N_i = N$.

We may partially order atlases by saying $\mathcal{A}_1 \leq \mathcal{A}_2$ (read "$\mathcal{A}_2$ dominates $\mathcal{A}_1$") iff for any subspace pair $(M_1, N_1) \in \mathcal{A}_1$ there exists a subspace pair $(M_2, N_2) \in \mathcal{A}_2$ with $M_1 \subseteq M_2$ and $N_1 \subseteq N_2$.

If the bx is well behaved[3] on individual subspace pairs, we may think of it as being piecewise well behaved; we have a chance of characterising under what circumstances it may be ill behaved. This works for almost any property of bx we might want to consider.

Definition Let $\mathcal{P}$ be a property of bx. A correct and hippocratic bx $R : M \leftrightarrow N$ has $\mathcal{P}$ with respect to an atlas $\mathcal{A}$ if $\mathcal{P}$ holds on the restriction of R to each subspace pair in $\mathcal{A}$. If the atlas is understood, we may simply say the bx is *piecewise* $\mathcal{P}$.

Generally speaking, the larger the subspace pairs in the chosen atlas, the more stringent this condition is; typically it is trivial for the trivial atlas comprising all one-point subspace pairs of consistent pairs of models. We might like to have a notion of coarsest atlas on which a given property held, but unfortunately the pointwise union of subspace pairs is not a subspace pair so this is not achievable. We can use the partial order on atlases given above to identify, possibly multiple, dominant atlases for a given property.

8.1 Language and tool implications

It is not our intention in this paper to propose a new bx language to take advantage of these ideas! But intuitively it is easy to see that some constructs in a language will, when exercised, give rise to well-behaved, e.g. Hölder continuous, bx while others will not. Projecting away certain information while leaving independent information untouched, for example, will formalise to being continuous provided that the chosen metrics on the two model spaces are appropriately related. On the other hand, `if-then-else` statements and similar notoriously have properties that will be formalisable as a failure of continuity; most recently this was explored, although not in those terms, in [HBTM15]. Effectful bx that use interaction with a user to resolve the choice of consistent model [ASCG+15] will provide another important class of non-continuous bx. One can certainly envisage future bx languages having their definitions annotated to indicate to the bx writer which constructs are "safe" from the point of view of a desirable property such as Hölder continuity, perhaps under assumptions concerning the metrics to be used.

Suppose that, by judicious instrumenting of a bx and the models on which it works (perhaps augmented by witness structure information, depending on the formalism), we can make the bx engine aware of an atlas with respect to which the bx has a good property. The "tool that goes beep" then operates as follows: at any moment, it knows which subspace pair(s) the last-synchronised pair of models was in. When it is asked to restore consistency between the current pair, it checks whether this pair is still in at least one of the same subspace pairs. If so, it restores consistency without beeping, because it knows that only an acceptable amount of surprise will be caused by what it does. If not, it beeps, because the good property can no longer be guaranteed and the user's attention may be required. There is a spectrum of possible behaviour, from automated restoration that simply needs to be validated by human attention, through effectful restoration that might require human confirmation before changes are made, through no automation at all: perhaps in this case the tool declines to

[3] NB "well behaved" is here used informally, not in the technical sense of the lens literature

restore consistency and just beeps to tell the users they must do so. Regardless, we think this is a promising avenue that may enable users to trust the bx's automated consistency restoration when it is trustworthy, reserving their attention for when it is not, and reducing the unpleasant surprises that unexpected behaviour of a bx might otherwise lead to.

8.2 Atlas examples

Looking at our technical examples, the most interesting case is Example 5.1. As remarked there, both $(\{C_M\}, \{C_N\})$ and $(M \setminus \{C_M\}, N \setminus \{C_N\})$ are subspace pairs; it is easy to see that they comprise an atlas on which any of the variants of the bx we considered there are piecewise strongly (1, 1)-Hölder continuous. A tool, like the one we have been discussing, that is aware of this property with respect to this atlas, would beep only in the difficult cases such when computing $\overrightarrow{R}(x_m, C_N)$, because (x_m, C_N) is not in any subspace pair of the atlas.

For a more naturalistic, informally expressed example, we return to the setting of Example 3.2, which we previously used to show that metric-least bx might not always behave as expected. Intuitively the problem in this example is that changing the set of classes stereotyped ⟪persistent⟫ is a change of a different kind from making changes to the signatures of individual methods or the code of tests. Keeping the same notion of consistency, let S be a bx with "better" consistency restorers than the R we considered previously. Specifically, $\overleftarrow{S}$ never touches a ⟪persistent⟫ stereotype if it can restore consistency without doing so: if necessary, it will modify the operations in a ⟪persistent⟫ class to bring them into consistency with the test suite. If the relevant classes are deleted from a test suite, so that there is no way for $\overleftarrow{S}$ to restore consistency without touching a ⟪persistent⟫ stereotype, then it will remove the stereotype. $\overrightarrow{S}$ has an easier job because tests are allowed that do not correspond to anything in the model; it simply has to change tests to match changes in the operation signatures within ⟪persistent⟫ classes if necessary, and generate default tests for any classes that are newly stereotyped ⟪persistent⟫.

For this S, the interesting subspace pairs may be identified with the sets of classes that are stereotyped ⟪persistent⟫. For a set P of classes, let M_P comprise all models in which precisely the classes of P are stereotyped ⟪persistent⟫, and let N_P comprise all test suites in which *at least* the classes in P have appropriately named test classes. Then each of the pairs (M_P, N_P) is a subspace pair, and the (infinite) collection of them is an atlas. On each subspace pair, the behaviour of S is very simple: subject to making it precise we expect it to be Hölder continuous (the constants depending on details such as the languages used, which we have not specified). Indeed, we also expect it to be metric-least. For once the bx is restricted to a given subspace pair, the "cheating" way to get a closer consistent model by deleting a stereotype is not available.

What about the behaviour of S on the whole of (M, N)? This will not, as previously discussed, be metric-least. Without further restrictions it will not be Hölder continuous either, since if a class can have arbitrarily many methods then the constant-sized change of adding a stereotype could lead to an arbitrarily-sized change to a test suite. In practice it is likely to be possible to impose some limit, so that the whole of S will be Hölder continuous, albeit with respect to laxer constants than the restriction to a subspace pair.

9 Automated software repair

Automated software repair is a diverse field with some clear connections to bx that have, as yet, been little explored. It concerns transforming code to make it better with respect to some criterion. The connection to bx is most obvious in test-suite based repair, as most famously implemented in the Genprog tool [LNFW12]. Such tools take a program and a test suite, such that the program fails at least one test, and attempt to create a patch that, when applied to the program, makes it pass all the tests. In our terminology, the program is consistent with the test suite if it passes all the tests, and automated software repair attempts to restore consistency (by modifying the program).

Typically, the challenge addressed by this work is restoring consistency at all: success is often measured by the increase in the number of passing tests, as causing all tests to pass may be too hard. In bx, this idea has been explored in [Ste14]. We can allow the consistency of a pair of models to be described using an element from an arbitrary partial order, rather than just a boolean value, and then we can consider bx that might not succeed in restoring consistency completely, but which do a worthwhile partial job. In the case of test-suite based repair, consistency could be described using the number (or alternatively the set) of passing test cases. Then a bx would be *improving* if it never decreased the number of passing test cases (or alternatively, if it never caused an individual test to fail that previously passed). In the partial setting, these basic properties have a Least Surprise flavour. However, this is off the main track of our concerns in this paper.

Setting the partiality concern aside, there may be a choice of ways to achieve the same degree of improvement. There are a few papers in the automatic software repair literature that consider the choice of repair, preferring "simpler" repairs, and these give us a direct connection with Least Surprise. In [TR15] Tan and Roychoudhury present their tool *relifix* and examine it against criteria that include "C1. Introduces small changes" and "C5. Only change if no regression will be introduced". In the same year, an overlapping set of authors present DirectFix [MYR15], an approach for generating "simple" patches. Interestingly, the principal simplicity criterion they use is preservation of the structure of a program. However, regressions may be introduced.

For further discussion of automated software repair we refer the interested reader to [Mon15].

10 Category theory

As previously discussed in Section 6, various authors have investigated least change in a partially (or even pre-) ordered setting. A natural generalisation of such work is to move from posets to categories. In particular, a number of people (notably Diskin *et al.* [DXC11a], Johnson, Rosebrugh *et al.* [JRW12, JR13, among others]) have considered generalisations of very well-behaved (a)symmetric lenses from the category of Sets to more general settings, notably Cat itself, the category of small categories.

The basic idea underlying these approaches is to go beyond the basic set-theoretic (state-based, whole-update) approach of lenses in order to incorporate additional information about the updates themselves, modelled as arrows. Rather than consider models, database states, as *elements* of an unstructured *set* (corresponding to a *discrete* category), they are taken as objects of a category $\mathbb{S}$. Arrows $\gamma : S \longrightarrow S'$

correspond to *updates*, from old state S to new state S'. Arrows from a given S carry a natural preorder structure induced by post-composition, generalising the order induced by multiplication in a monoid:

$$\gamma : S \longrightarrow S' \leq \gamma' : S \longrightarrow S'' \quad \text{iff} \quad \exists \delta : S' \longrightarrow S''.\gamma' = \gamma; \delta$$

Johnson, Rosebrugh and Wood introduced the idea of a *c-lens* [JRW12] as the appropriate categorical generalisation of very-well-behaved asymmetric lens: given by a functor $G : \mathbb{S} \longrightarrow \mathbb{V}$ specifying a *view* of $\mathbb{S}$, together with data defining the analogues of Put, satisfying appropriate analogues of the GetPut, PutGet and PutPut laws (we omit the details, which are spelt out very clearly, if compactly, in their subsequent paper [JR13]). They make explicit the connection with, and generalisation of, Hegner's earlier work characterising least-change update strategies on databases [Heg04]; in the categorical setting, database instances are models of a sketch defining an underlying database schema or entity-relationship model.

The crucial detail is that the 'Put' functor then operates not on pairs of states and views alone (that is, objects of $\mathbb{S} \times \mathbb{V}$), but on updates from the image of some state S under G to a new view V, that is, on pairs consisting of an $\mathbb{S}$-state S and a $\mathbb{V}$-arrow $\alpha : GS \longrightarrow V$, returning a new $\mathbb{S}$-arrow $\gamma : S \longrightarrow S'$ such that $G\gamma = \alpha$. In other words, Put not only returns a new updated state S' on the basis of a updated view V, but also an update from S to S' that is correlated with the view update α. Notice that, because we have an update to a source S correlated with an update to a view GS which is *consistent* with the source, we are in the *weak* least surprise setting – but since very-well-behavedness in their framework is history-ignorance, the notions of weak and strong least surprise coincide.

They then show that the laws for Put establish that such an update γ is in fact *least* (indeed, *unique*) up to the $\leq$ ordering, namely as an *op-cartesian* lifting of α. Thus very-well-behaved asymmetric c-lenses do enjoy a Principle of Least Change. Extending this analysis, which applies to *insert* updates, with *deletes* being considered dually via a *cartesian* lifting condition on arrows $\alpha : V \longrightarrow GS$, to the symmetric case is the object of ongoing study in terms of spans of c-lenses.

Johnson and Roseburgh further showed that Diskin *et al.*'s *delta lenses* [DXC11a] generalise the c-lens definition; in particular, every c-lens gives rise to an associated d-lens. However, such d-lenses do *not* enjoy the unique arrow-lifting property, so there is some outstanding issue about the generality of the d-lens definition. In subsequent work [DXC$^+$11b], Diskin *et al.* have given a definition of symmetric d-lenses. It remains to be seen what least-change properties such structures might enjoy, on their own, or by reference to spans of c-lenses.

11 Conclusions and future work

The vision of bx in MDD is that developers should be able to work on essentially arbitrary models, which capture just what is relevant to them, supported by languages and tools which make it straightforward to define consistency and restore consistency between their model and others being used elsewhere. Clearly, if anything like this is to be achieved, there is vastly more work to be done on all fronts. Although there are some islands of impressive theoretical results (e.g. in category theory) and some pragmatically useful tools (e.g. based on TGGs), the theory currently works only in very idealised settings and the tools offer inadequate guarantees while still lacking flexibility. For software development, this situation limits productivity.

We understand enough already to be sure that we will never achieve behaviour guarantees as strong as we would ideally like within the flexible tools we need: history ignorance is a long-understood example of a desirable, but usually unobtainable, property. In this paper we have begun to speculate about what might be achieved by using strong properties such as Hölder continuity, by identifying "good" parts of the model spaces where such guarantees can be offered, and by developing tools that warn their users when these guarantees do not apply, so that they can spend their attention where it is most needed. We have discussed how a bx might offer a strong behaviour guarantee but only piecewise, with respect to an atlas of subspace pairs. All this needs to be pursued and validated in practical tools and languages. Many questions remain open.

If we decide, despite its disadvantages, to use the metric-least approach (perhaps only piecewise) and need information from users to define domain-specific (or even bx-specific) metrics, how can we elicit this information without unacceptably burdening users? Besides the metric-least approach, are there other ways of inferring *optimal* consistency restoration behaviour from the consistency relation alone? Or are we correct in thinking that we should rather focus on *reasonable* behaviour? Can we glean any further inspiration from the Principle of Least Surprise (or Astonishment) in HCI (itself really about reasonable behaviour, despite the name: interface users might not know precisely what to expect, but when they see what happens, they should not be surprised)? How can a bx language best be designed to allow the bx developer to express what they mean, and to support the bx user in their work?

There are other possible approaches to the Least Surprise problem that we have not yet covered. We have hardly mentioned topology, although we have touched on both metric spaces (a specialisation) and category theory (a generalisation). Perhaps the language of topology, or even algebraic topology, might help us to make progress. Even more speculatively, as type theory, especially dependent type theory, is a language we work in elsewhere, it is natural to wonder whether at some point spatial aspects of types such as for example homotopy type theory will have a role.

We have not attempted to analyse which properties any of the existing formalisms provide, or could provide with feasible instrumentation or modification. Do any of the many existing bx formalisms that we have not mentioned, each thoughtfully designed in an attempt to "do the right thing", satisfy any of the properties discussed here – and if not, why not? Under what circumstances do only global optima exist, so that guaranteeing reasonable behaviour would automatically guarantee optimal behaviour? Would identifying such circumstances help to resolve the tension between wanting optimality and wanting composition? Is there any mileage in applying the kind of guarantees of reasonable behaviour considered here to partial bx in the sense of [Ste14], which do not necessarily restore consistency but, in an appropriate sense, at least improve it (perhaps making use of ideas from automatic software repair)? Could one make use of insights from Lagrangian and Hamiltonian mechanics? Or from simulated annealing?

Nailing our colours to the mast, we think: change to witness structures is important; pursuing reasonable behaviour will be more fruitful than pursuing optimal behaviour; identifying "good" subspace pairs will help tools in practice; and weak least surprise is not enough. But there is plenty of room for other opinions.

References

[ASCG+15] Faris Abou-Saleh, James Cheney, Jeremy Gibbons, James McKinna, and Perdita Stevens. Notions of bidirectional computation and entangled state monads. In *Proceedings of MPC*, number 9129 in LNCS, 2015. doi:10.1007/978-3-319-19797-5_9.

[BKT02] Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. On propagation of deletions and annotations through views. In *Proceedings of PODS*, pages 150–158. ACM, 2002. doi:10.1145/543613.543633.

[BS12] Julian C. Bradfield and Perdita Stevens. Recursive checkonly QVT-R transformations with general when and where clauses via the modal mu calculus. In *Proceedings of FASE*, volume 7212 of *LNCS*, pages 194–208. Springer, 2012. doi:10.1007/978-3-642-28872-2_14.

[CGMS15] James Cheney, Jeremy Gibbons, James McKinna, and Perdita Stevens. Towards a principle of least surprise for bidirectional transformations. In *Proceedings of Bx*, volume 1396 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[CMSG14] James Cheney, James McKinna, Perdita Stevens, and Jeremy Gibbons. Towards a repository of bx examples. In K. Selçuk Candan, Sihem Amer-Yahia, Nicole Schweikardt, Vassilis Christophides, and Vincent Leroy, editors, *Proceedings of the Workshops of EDBT/ICDT*, volume 1133 of *CEUR Workshop Proceedings*, pages 87–91. CEUR-WS.org, 2014.

[DEPC16] Zinovy Diskin, Romina Eramo, Alfonso Pierantonio, and Krzysztof Czarnecki. Incorporating uncertainty into bidirectional model transformations and their delta-lens formalization. In Anthony Anjorin and Jeremy Gibbons, editors, *Proceedings of Bx*, volume 1571 of *CEUR Workshop Proceedings*, pages 15–31. CEUR-WS.org, 2016.

[Dis08] Zinovy Diskin. Algebraic models for bidirectional model synchronization. In Krzysztof Czarnecki, Ileana Ober, Jean-Michel Bruel, Axel Uhl, and Markus Völter, editors, *Proceedings of MODELS*, volume 5301 of *Lecture Notes in Computer Science*, pages 21–36. Springer, 2008. doi:10.1007/978-3-540-87875-9_2.

[DXC11a] Zinovy Diskin, Yingfei Xiong, and Krzysztof Czarnecki. From state- to delta-based bidirectional model transformations: the asymmetric case. *JOT*, 10:6: 1–25, 2011. doi:10.5381/jot.2011.10.1.a6.

[DXC+11b] Zinovy Diskin, Yingfei Xiong, Krzysztof Czarnecki, Hartmut Ehrig, Frank Hermann, and Fernando Orejas. From state- to delta-based bidirectional model transformations: The symmetric case. In *Proceedings of MODELS*, pages 304–318, 2011. doi:10.1007/978-3-642-24485-8_22.

[HBTM15] Soichiro Hidaka, Martin Billes, Quang Minh Tran, and Kazutaka Matsuda. Trace-based approach to editability and correspondence analysis for bidirectional graph transformations. In Alcino Cunha and Ekkart Kindler, editors, *Proceedings of Bx*, volume 1396 of *CEUR Workshop Proceedings*, pages 51–65. CEUR-WS.org, 2015. URL: http://ceur-ws.org/Vol-1396/p51-hidaka.pdf.

[Heg04] Stephen J. Hegner. An order-based theory of updates for closed database views. *Ann. Math. Artif. Intell.*, 40(1-2):63–125, 2004. doi:10.1023/A:1026158013113.

[Heg11] Stephen J. Hegner. Information-based distance measures and the canonical reflection of view updates. *Ann. Math. Artif. Intell.*, 63(3-4):317–355, 2011. doi:10.1007/s10472-012-9278-x.

[JR13] Michael Johnson and Robert D. Rosebrugh. Delta lenses and opfibrations. *ECEASST*, 57, 2013. doi:10.14279/tuj.eceasst.57.875.866.

[JRW12] Michael Johnson, Robert D. Rosebrugh, and Richard J. Wood. Lenses, fibrations and universal translations. *MSCS*, 22:25–42, 2 2012. doi:10.1017/S0960129511000442.

[LAS+14] Erhan Leblebici, Anthony Anjorin, Andy Schürr, Stephan Hildebrandt, Jan Rieke, and Joel Greenyer. A comparison of incremental triple graph grammar tools. *ECEASST*, 67, 2014. doi:10.14279/tuj.eceasst.67.939.928.

[LNFW12] Claire Le Goues, ThanhVu Nguyen, Stephanie Forrest, and Westley Weimer. Genprog: A generic method for automatic software repair. *IEEE Trans. Software Eng.*, 38(1):54–72, 2012. doi:10.1109/TSE.2011.104.

[MC16] Nuno Macedo and Alcino Cunha. Least-change bidirectional model transformation with QVT-R and ATL. *Software and System Modeling*, 15(3):783–810, 2016. doi:10.1007/s10270-014-0437-x.

[McK16] James McKinna. Bidirectional transformations with deltas: A dependently typed approach (talk proposal). In Anthony Anjorin and Jeremy Gibbons, editors, *Proceedings of Bx*, volume 1571 of *CEUR Workshop Proceedings*, page 14. CEUR-WS.org, 2016. URL: http://ceur-ws.org/Vol-1571/paper_11.pdf.

[Mee98] Lambert Meertens. Designing constraint maintainers for user interaction. Unpublished manuscript, June 1998. URL: http://www.kestrel.edu/home/people/meertens/.

[Mon15] Martin Monperrus. Automatic software repair: a bibliography. Technical Report hal-01206501, University of Lille, 2015. Latest version: http://www.monperrus.net/martin/survey-automatic-repair.pdf. This version retrieved 15/6/16.

[MPCO13] Nuno Macedo, Hugo Pacheco, Alcino Cunha, and José Nuno Oliveira. Composing least-change lenses. *ECEASST*, 57, 2013. doi:10.14279/tuj.eceasst.57.868.862.

[MYR15] Sergey Mechtaev, Jooyong Yi, and Abhik Roychoudhury. Directfix: Looking for simple program repairs. In Antonia Bertolino, Gerardo Canfora, and Sebastian G. Elbaum, editors, *Proceedings of ICSE*, pages 448–458. IEEE Computer Society, 2015. doi:10.1109/ICSE.2015.63.

[OMG15] OMG. MOF2.0 query/view/transformation (QVT) version 1.2. OMG document formal/2015-02-01, 2015. URL: http://www.omg.org/spec/QVT/.

[SK08] Andy Schürr and Felix Klar. 15 years of triple graph grammars. In *Proceedings of ICGT*, volume 5214 of *LNCS*, pages 411–425. Springer, 2008. doi:10.1007/978-3-540-87405-8_28.

[Ste12] Perdita Stevens. Observations relating to the equivalences induced on model sets by bidirectional transformations. *ECEASST*, 049, 2012. doi:10.14279/tuj.eceasst.49.714.720.

[Ste13] Perdita Stevens. A simple game-theoretic approach to checkonly QVT Relations. *Journal of Software and Systems Modeling (SoSyM)*, 12(1):175–199, 2013. doi:10.1007/s10270-011-0198-8.

[Ste14] Perdita Stevens. Bidirectionally tolerating inconsistency: Partial transformations. In Stefania Gnesi and Arend Rensink, editors, *Proceedings of FASE*, volume 8411 of *LNCS*, pages 32–46. Springer, 2014. doi:10.1007/978-3-642-54804-8_3.

[Sut75] W A Sutherland. *Introduction to metric and topological spaces*. Oxford University Press, 1975.

[TR15] Shin Hwei Tan and Abhik Roychoudhury. relifix: Automated repair of software regressions. In Antonia Bertolino, Gerardo Canfora, and Sebastian G. Elbaum, editors, *Proceedings of ICSE*, pages 471–482. IEEE Computer Society, 2015. doi:10.1109/ICSE.2015.65.

[WvP10] Stefan Winkler and Jens von Pilgrim. A survey of traceability in requirements engineering and model-driven development. *Software and System Modeling*, 9(4):529–565, 2010. doi:10.1007/s10270-009-0145-0.

About the authors

James Cheney is Royal Society University Research Fellow and Reader in the Laboratory for Foundations of Computer Science, School of Informatics, University of Edinburgh. Contact him at jcheney@inf.ed.ac.uk, or visit http://homepages.inf.ed.ac.uk/jcheney/.

Jeremy Gibbons is Professor of Computing at the University of Oxford, where he is Director of the part-time professional Software Engineering Programme. Contact him at jeremy.gibbons@cs.ox.ac.uk, or visit http://www.cs.ox.ac.uk/jeremy.gibbons/.

James McKinna is Senior Research Fellow in the Laboratory for Foundations of Computer Science, School of Informatics, University of Edinburgh. Contact him at `James.McKinna@ed.ac.uk`, or visit `http://homepages.inf.ed.ac.uk/jmckinna/`.

Perdita Stevens is Professor of Mathematics of Software Engineering in the Laboratory for Foundations of Computer Science, School of Informatics, University of Edinburgh. Contact her at `Perdita.Stevens@ed.ac.uk`, or visit `http://homepages.inf.ed.ac.uk/perdita/`.

Acknowledgments We thank the anonymous reviewers, Faris Abou-Saleh, and the bx community for helpful comments and discussions. The work was funded by EPSRC (EP/K020218/1, EP/K020919/1).