Towards a Principle of Least Surprise for bidirectional transformations

James Cheney[1], Jeremy Gibbons[2], James McKinna[1], and Perdita Stevens[1]

1. School of Informatics
University of Edinburgh, UK
`Firstname.Lastname@ed.ac.uk`

2. Department of Computer Science
University of Oxford, UK
`Firstname.Lastname@cs.ox.ac.uk`

Abstract

In software engineering and elsewhere, it is common for different people to 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 identify that what is actually needed is a Principle of Least Surprise, to limit a bx to reasonable behaviour. We present candidate formalisations of this, but none is obviously right for all circumstances. We point out areas where further work might be fruitful, and invite discussion.

1 Introduction

Bx restore consistency between artefacts; this is the primary definition of what it is to be a bx. In broad terms, 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 models (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" should mean.

Even in the relational setting, the best definition of "reasonable" is not obvious. Meertens' seminal but unpublished paper [14] discussed the need for a Principle of Least Change for constraint maintainers, essentially what we now call relational bx. His formulation is:

In: A. Cunha, E. Kindler (eds.): Proceedings of the Fourth International Workshop on Bidirectional Transformations (Bx 2015), L'Aquila, Italy, July 24, 2015, published at http://ceur-ws.org

> 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.

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 "principle of least surprise" 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.

We find it helpful to bear in mind the domain of model-driven development (MDD) and how bx fit into it. Hence in this paper 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. The fundamental idea of MDD is that development can be made more efficient by allowing people to work with models that express just what they need to know to fulfill 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 *changes* that do not affect them. The need for bx arises because 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 inherently dangerous, because the changes are, to a degree, inherently surprising: they are caused by changes to information the user of the model does not know about.

Without automated bx, these changes have to be decided on by humans. Imagine I am at a developers' meeting focused (solely) on deciding how to restore consistency by changing my model. There might well be disagreement about the optimal way to achieve this end. Some courses of action, 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. As well as disagreement about what course of action was *optimal*, we might also expect disagreement also about what courses were *(un)reasonable*; there might 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.

Humans agree what is reasonable by a social process; to ensure that a bx's behaviour will be seen by humans as reasonable, we must formalise the desired properties. We will bear in mind the meeting of reasonable developers as a guide: in particular, it suggests that we should not expect to find one right answer.

One property is (usually, but not always!) easily agreed: if the models are already consistent, the meeting, or bx, has no job to do. It will make no change. This is hippocraticness.

Going beyond this, 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".

It would be easy to throw up one's hands at the complexity of this task. Fortunately, the study of bx is far from the only situation in which such considerations arise, and we are able to draw on a rich literature, mathematical and otherwise.

1.1 Notation and scope

When talking about state-based relational bx we use the now-standard (see e.g. [15]) notation $R : M \leftrightarrow N$ for a bx comprising consistency relation R and restorers $\overrightarrow{R} : M \times N \to N$, $\overleftarrow{R} : M \times N \to M$. Such bx will be assumed to be correct (restorers do restore consistency) and hippocratic (given consistent arguments, they make no change). Where possible we will broaden the discussion to permit, for example, intensional information about changes to models, and auxiliary structures besides the models themselves. However, even then, we will assume that consistency restoration involves taking one model as authoritative, and changing the other model (and, perhaps, some auxiliary structures). More complex scenarios are postponed to future work.

Since consistency is primary, we will assume that the bx developer defines what it is for models to be consistent. It may be simply a relation over a pair of sets of models, or it may be something more complex involving witnessing structures such as sets of trace links. Regardless, we assume that we do not have the right to change the developers' notion of what is a good, consistent state of the world. Our task is to constrain how a bx restores the world to such a state. Of course, in general there are many ways to do so: this is the essence of the problem. In a setting where, given one model, there is a unique choice of consistent other model, there is no role for a Least Change Principle, because there is no choice between consistent resolutions to be made. Even here, though, we might wish to know whether a specific principle was obeyed, in order to have an understanding of whether developers using it were likely to get nasty surprises. This begins the motivation for replacing a search for a Least *Change* Principle with a broader consideration of possible Least *Surprise* Principles.

Let us go on to make explicit a key non-ambition of this work: we do not seek to identify a principle that is so strong that, given the consistency relation, there is a unique way to restore consistency that satisfies it. We envisage that the bx programmer has to specify consistency and may also have to provide information about how consistency is restored where there is a choice; we also wish to leave open the door to applying the chosen Principle to effectful bx [1], e.g. bx that involve user interaction, where the user might also be involved in this choice (in a more sophisticated way than having to choose between all consistent models). By contrast, some of the work we shall draw on ([8] for example) does regard it as a deficiency in the principle if there might be more than one way to satisfy it.

We will use a few basic notions of mathematical analysis (see e.g. [17]), such as

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

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

1.2 Structure of the paper

In Sections 2 and 3 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, but based on our investigations we think it may be important. In Section 4 we collect some examples that we shall use in the rest of the paper; the reader may not wish to read them all in detail until they are referred to, but, as many are referred to from different places, we have preferred to place them together. In Section 5 we consider approaches based on ordering changes themselves; we go on in Section 6 to consider perhaps the purest form of "least change", based on metrics on the model spaces. Because of the disadvantages inherent in that approach, we decide to go on, in Section 7, to consider in some detail an approach based on guaranteeing "reasonably small" changes instead. Section 8 considers categorical approaches, and Section 9 concludes and discusses some future 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 the entire paper is about related work. Our own modest technical contribution is in Section 7.

2 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 [8] 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.

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 is 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 [15] 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.")

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.) 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 (cf 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 using the larger structure. We leave the rest of this fascinating topic for future work.

3 Beyond least change: weak and strong least surprise

Strictly following a formalisation of a "least change" definition, such as Meertens suggested, allows the bx writer very limited freedom to choose how consistency restoration is done: the definition prescribes *optimal* behaviour by a rigid definition of optimal, and the only real flexibility the user has concerns the way the size of a change is measured. In the spirit of the "least surprise" principle from UI design, it may be more fruitful to formalise what behaviour is *reasonable*; this gives more flexibility to the bx writer, while still promising absence of surprise. Here, reasonableness will be about the sizes of changes: we will set the scene for continuity-based ideas, guaranteeing that the size of the change to a model caused by consistency restoration can be limited, in terms of the size of the change being propagated.

When should such guarantees apply? Does the bx guarantee to behave reasonably even when both models have been modified concurrently? We illustrate this with informal scenarios in the relational setting.

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 at models b that are consistent with a, only.

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.[1] 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 sane with respect to, a. Another promising option is $C = M \times N$ for a subspace pair [16] (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 might 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. Example 4.7 illustrates.

4 Examples

In this section we present a number of examples intended to be thought-provoking, in that they demonstrate various ways in which it can fail to be obvious what consistency restoration best obeys a Least Surprise Principle. Some are drawn from the literature. We have collected them in one section for ease of reference; the reader should feel free to skim and return. Names are those used in the Bx Examples Repository[2] [4].

[1]Cf the distinction between undoability and history ignorance.

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

Let us begin with an MDD-inspired example, which illustrates that we may not always want the consistent model which is intuitively closest.

Example 4.1 (ModelTests) *Suppose bx R relates UML model m with test suite n, saying that they are consistent provided that every class in m stereotyped ⟪persistent⟫ 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 ⟪persistent⟫ 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 not what is wanted.*

There are two possible reactions to this. One is to argue that the desired change to the detail of C should satisfy a good Principle of Least Change, and that if it does not, we have the wrong Principle. We may take this as a justification for considering the size of changes made to auxiliary structures as well as the size of changes made to the model itself. We might imagine a witness structure comprising trace links between public operations of ⟪persistent⟫ classes and the corresponding tests. In this case, we might argue that although removing the stereotype from C is a small change to the UML model, it involves removing a trace link for every public operation of C from the witness structure; this might be formalised to justify an intuitive feeling that actually removing the stereotype is a large change, so that a Principle of Least Change might indeed prefer the user's expected change. The other possible reaction is to say that indeed, the desired modification does not satisfy a Least Change Principle in this case, and take this as evidence that there may be cases where following such a principle is undesirable.

Next we look at the most trivial possible examples. For there to be a question about what the best restoration is, there must be inconsistent states, and there must be a choice of how to restore consistency. One moral of this example is that the notion of "size of change" or "amount of disruption" can, even in the simplest case, be context-dependent. Which of two changes appears bigger is interdependent with the way in which changes are represented, which in turn is interdependent with the representation of the models.

Example 4.2 (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.

Many bx involve abstracting away information, and intuitively, bx which violate "least change" can arise when a small change in a concrete space results in a large change in an abstract space. Let us illustrate.

Example 4.3 (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 $-$.

We will later want to consider two variants on this example.

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 $\{x/100 : x \in \mathbb{Z}, -100 \leq x \leq 100\}$.

Discreteness of model spaces turns out to be important.

Our next example is adapted from [8], where it is presented in a database context. One moral of this example is 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 4.4 (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 [8] 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 4.2 may bring even this into doubt.

The same issue arises in our next example, which is adapted from [2]; 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 4.5 (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 [2] 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*

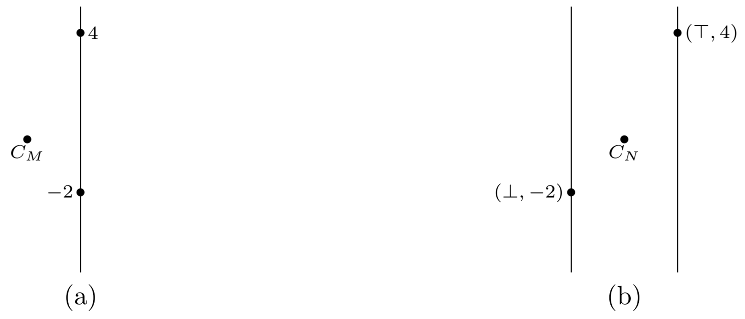

Figure 1: (a) Model space M, (b) model space N for Example 4.7

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 [2], 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.

This example is adapted from Example 5.2d of [14].

Example 4.6 (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 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 $\overleftarrow{R}((s,t),n) = (n \cup (s \setminus t), n \cup t)$.

Our next example will enable us to study notions of least change coming from mathematical analysis, and also to illustrate the difference between the "strong" and "weak" least surprise scenarios.

Example 4.7 (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 $d_N((\top, n_1), (\perp, n_2)) = 2 + |n_1| + |n_2|$.*

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.

The next 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, but it will allow us to demonstrate a technical point later.

Example 4.8 (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 $1/\theta$.*

5 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 [14] 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. Example 4.6 illustrates. Using a similar structure, Macedo et al. [13] 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.

5.1 Changes as sets of small changes

A preorder on changes is a useful starting point in cases where changes are uncontroversially 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 can then generate a preorder on changes from the inclusion ordering on sets, and this gives a way to prefer changes that do not add or delete elements unnecessarily. Even there, a drawback is that if an element is modified, perhaps very slightly, and we must model this as a deletion of one thing and an addition of something very similar, this change appears bigger than it is. Indeed Example 4.5 is a cautionary tale on how such an approach can produce poor results, where models are structured collections of elements, not just sets.

A similar approach is used by TGGs when used in an incremental change scenario; [11] 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 [11] 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. The issues of handling changes other than as additions plus deletions (mentioned above) and of avoiding creating new elements where old ones could instead of reused (cf Example 4.5 and Example 4.4), 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.

6 Measuring changes

One very natural approach, particularly in the pure state-based setting, is to assume we are given a metric on each model space, and require that the distance between the old, and the chosen new, target models is minimal among all distances between the old target model and any new target model that restores consistency. That is, the effect on their 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).

An instance of this approach has been explored and implemented by Macedo and Cunha in [12], although they do not explicitly use the term "metric". 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, which amounts to defining their own metric on the space of models.

This approach is very sensitive to the specific metric chosen, and, because it operates after a model has been translated, the graph edit distance used in [12] 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 saw an example of this already in Example 4.2. If my tool provides a simple menu item to make a change that, in your tool, requires many manual steps, 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. Still, it is reasonable to expect this approach to give more intuitive results than those of the standard QVT-R specification illustrated in Example 4.5.

One still needs a way to resolve non-determinism; it is unlikely that there will be a unique closest consistent model. In [12], the tool offered to the user all consistent models found at the same minimum distance from the current model. The implementation is very resource intensive and not practical for non-toy cases, and this is probably essential because of the need to consider all models at a given distance from the current one.

We are pessimistic about whether usably efficient algorithms that guarantee to find optimal solutions to the least change problem will ever be available, because Buneman, Khanna and Tan's seminal paper [3] addressing the closely-related *minimal update problem* in databases 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 $Q(S \setminus T) = Q(S) \setminus \{t\}$.

Example 4.1 shows, we think, that the model that will be found by this approach will not always be what the user desires in any case. It is possible, though, that by applying exactly the same approach to the witness structure, rather than the model, we might get better behaviour. This would be interesting to investigate.

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.

Note that this 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.

The bx in Example 4.2 will be metric-least or not depending on the chosen metric on N. That in Example 4.3 will be metric-least, with respect to the usual metric on M and with any positive distance between $+$ and $-$ in N. Variant 1 of that example cannot be, however, for the reason given there. Variant 2 is metric-least, regardless of whether 0 is considered consistent with both of $+$ and $-$ or just one. All three variants of Example 4.7 are metric-least.

7 Continuity and other forms of structure preservation

We turn now to codifying reasonable, rather than optimal, behaviour; in a sense we now make the equation "least change = most preservation". The senses in which transformations preserve things have of course been long studied in mathematics (and abstracted in category theory).

The most basic idea, and the most natural way to move on from the metrics-based approach considered in the previous section, is continuity, in the following metric-based formulation. (The other setting in which continuity appears in undergraduate mathematics, 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 [17] 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 3, 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 4.7 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 $\delta = \delta'$. ▮

This approach has advantages over metric-leastness that we do not have space to explore, but it is worth giving one example. It is straightforward to prove:

Theorem 7.2 *Let $R : M \leftrightarrow N$ and $S : N \leftrightarrow P$ be strongly [rsp. 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 [rsp. 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 4.7 are metric-least, so metric-leastness does not imply strong continuity. In fact it does not even imply weak continuity; Example 4.3 is metric-least, but not weakly continuous at $(0,+)$. Conversely, Lemma 7.3 shows that even strong continuity does not imply metric-leastness; apply discrete metrics to M and N in Example 4.2, 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.

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 4.7 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 7.4 *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, as Example 4.8 (which, recall, actually had no choice about how to restore consistency) shows: 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'.

Thus it is possible that Hölder continuity might turn out to be a useful least change principle, and we leave this as an open research question; we remark, though, that continuity is already a strong condition, Hölder continuity much stronger, and it seems more likely that this will be a "nice if you can get it" property rather than one it is reasonable to insist on. Still, it might be interesting to consider, for example, subspace pairs on which it holds, and the possibility that a tool might indicate when a user leaves such a subspace pair.

Future work might consider e.g. locally Lipschitz functions, or in a sufficiently special space, continuously differentiable functions, etc.

8 Category theory

As previously discussed in Section 5, 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.* [5], Johnson, Rosebrugh *et al.* [10, 9, 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** [10] 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 [9]). They make explicit the connection with, and generalisation of, Hegner's earlier work characterising least-change update strategies on databases [7]; 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* [5] 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 [6], 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.

9 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. For researchers it is an adventure playground.

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. But the limits of what can be achieved are unknown. How far can we go, for example, by identifying "good" parts of the model spaces, where guarantees can be offered, and developing tools that warn their users when danger is near, so that they can spend their attention where it is most needed? If we need information from users to define domain-specific metrics, how can we elicit this information without unacceptably burdening users? Can we do better by focusing on *reasonable* behaviour than on *optimal* behaviour? It is noteworthy that, despite the name, HCI experts following the Principle of Least Surprise (or Astonishment) there are not really making an optimality claim so much as a reasonableness one: interface users might not know precisely what to expect, but when they see what happens, they should not be surprised. We think this is a good guide.

We have been pointing out, through the paper, areas we think need work (or play). Let us now mention some others that we have not touched on here. We have not 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. 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 [16], which do not necessarily restore consistency but, in an appropriate sense, at least improve it? Could someone 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.

Acknowledgements

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).

References

[1] 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. To appear.

[2] 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.

[3] 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.

[4] 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.

[5] Zinovy Diskin, Yingfei Xiong, and Krzysztof Czarnecki. From state- to delta-based bidirectional model transformations: the asymmetric case. *Journal of Object Technology*, 10:6: 1–25, 2011.

[6] 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.

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

[8] Stephen J. Hegner. Information-based distance measures and the canonical reflection of view updates. *Ann. Math. Artif. Intell.*, 63(3-4):317–355, 2011.

[9] Michael Johnson and Robert D. Rosebrugh. Delta lenses and opfibrations. *ECEASST*, 57, 2013.

[10] Michael Johnson, Robert D. Rosebrugh, and Richard J. Wood. Lenses, fibrations and universal translations. *Mathematical Structures in Computer Science*, 22:25–42, 2 2012.

[11] Erhan Leblebici, Anthony Anjorin, Andy Schürr, Stephan Hildebrandt, Jan Rieke, and Joel Greenyer. A comparison of incremental triple graph grammar tools. *ECEASST*, 67, 2014.

[12] Nuno Macedo and Alcino Cunha. Implementing QVT-R bidirectional model transformations using alloy. In Vittorio Cortellessa and Dániel Varró, editors, *Proceedings of FASE*, volume 7793 of *LNCS*, pages 297–311. Springer, 2013.

[13] Nuno Macedo, Hugo Pacheco, Alcino Cunha, and José Nuno Oliveira. Composing least-change lenses. *ECEASST*, 57, 2013.

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

[15] Perdita Stevens. A simple game-theoretic approach to checkonly QVT Relations. *Journal of Software and Systems Modeling (SoSyM)*, 12(1):175–199, 2013. Published online, 16 March 2011.

[16] 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.

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