Delta Bx Revisited:
a Rational Reconstruction of Set-based Bx

Complements, Proof-Relevance, Dependent Types, Alignment, Bisimulations, Bicategories and more!

James McKinna

TLCBX@LFCS, Edinburgh

SSBX@LMH, Oxford
2016-07-27

Outline

i) complements witness consistency

    0) proof-relevance, dependent types

ii) deltas as relations, as dependent types

iii) delta propagation as proof-relevant (bi-)simulation

iv) composing bx: fun with tuples (and some bicategories)

v) align, restore, project: reconciling set-based bx

vi) hippocraticness and PutPut

vii) why go to all this trouble?

i) complements witness consistency

what's a complement, anyway?

A lens (get g, put p) induces *equivalence* relation on sources:

$$s \sim t =_{\mathrm{def}} \forall v. p\, s\, v = p\, t\, v$$

Accordingly, let $C =_{\mathrm{def}} S/\sim$. Then we have $S \simeq V \times C$. Consider:

- $\phi : s \mapsto (g\, s, [s]_{\sim})$
- $\psi : (v, c) \mapsto p\, s\, v$, where $c =_{\mathrm{def}} [s]_{\sim}$ for some s

Then, by the defn of $\sim$,

- $\psi(\phi s) = s$, by PutGet
- $\phi(\psi(v, c)) = (v, c)$, by GetPut, *provided* the lens is very-well-behaved

Call C the lens *complement*: it's (the set of)

what's needed to reconstruct s from $v =_{\mathrm{def}} g\, s$

consistency for lenses

Consistency relation:

$$R(s, v) =_{\text{def}} v = g\, s$$

- Idea: let $T(s, v) =_{\text{def}} \{[s]_\sim \mid R(s, v)\}$, a subset of C.
- Then: $c =_{\text{def}} [s]_\sim \in T(s, v)$ if and only if $R(s, v)$ holds.
- And: that c, in the very-well-behaved case, is *unique*.

So:

the existence of the complement element $c \in T(s, v)$
*is a **witness** to the proposition $R(s, v)$*

0) digression:
proof-relevance

an old idea

Idea (older than Gödel's incompleteness, or Turing's computable numbers, or Church's λ-calculus), due to Brouwer, Kolmogorov and others:

> *a proof is a **procedure** which transforms (evidence for) premises into (evidence for) conclusion*

So: instead of a proposition being true, consider the *set* of its possible proofs (evidence for why it holds)

What's *evidence*?
A justification of *why* something is the case.

dependent type theory

So, reorganise set theory along the lines suggested by this intuition:

- sets (data types) correspond to propositions (logical formulas);
- elements to proofs;
- predicates (propositions depending on an argument) become sets (types) depending on a parameter

Distinguished type formers:

- (Π): implication and universal quantification correspond to dependent function space (values are functions)
- (Σ): conjunction and existential quantification correspond to dependent sums (tagged disjoint unions: values are *tuples*)

relations as dependent types

a (binary) relation is now: a family of types with two parameters

inclusion of relations (proof-relevance again):

$$R \subseteq_p S =_{\text{def}} p : \Pi_{a:A} \Pi_{b:B}\, a\, R\, b \to a\, S\, b$$

with:

$$R \subseteq_p S \subseteq_q T =_{\text{def}} R \subseteq_{q \circ p} T$$

relational composition:

$$a\,(R \otimes S)\, c =_{\text{def}} \Sigma_{b:B}\, a\, R\, b \times b\, C\, c$$

elements of $a\,(R \otimes S)\, c$ are thus *triples* (b, r, s):

- an element $b : B$
- a proof $r : a\, R\, b$
- a proof $s : b\, S\, c$

i) complements witness consistency, redux

back to complements

from now on:

- the consistency relation is a dependent type family
- corresponding to a fine-grained notion of generalised lens complement
- elements of the complement witness the corresponding proposition of consistency

omitted: this accounts for the role of lens complement in

- symmetric lenses, edit lenses (consistent triples)
- delta lenses (correspondences)

ii) deltas as dependent types

as you might expect; as you might not expect

what's a delta?

- (Martin) a thing in the wild, that acts partially on states: $\delta : \partial^X$, with $\delta \bullet x = x'$ possibly undefined;
- (Mike) a fully-specified thing, self-describing its action: $\delta : \partial^X\,(x, x')$ means that δ transforms $x \mapsto x'$
- (Cai *et al.*; Hancock *et al.*) a half-way house: for every x, we have a dependent type ∂^X_x of *the allowable deltas available at* x, with a *total* action $\bullet : \Pi_{x:X}\ \partial^X_x \to X$

modulo technicalities, these are equivalent: for our (relational) purposes, we take the (MJ) notion

- Tony's *consistent deltas*: a derivation in the grammar is the witness

iii) delta propagation as proof-relevant (bi-)simulation

fpg and bpg as proof-relevant witnesses

square filling *à la* Tony:

- take a span from the top-left (fpg) or top-right (bpg) corner
- construct a new co-span in the opposite corner

where a span is:

- an element of a relational composition

and a co-span is

- an element of a relational composition

thus we (might) see fpg, bpg, as *dependently-typed functions*

- $(\partial^A)^{\mathrm{op}} \otimes T \subseteq_{fpg} T \otimes (\partial^B)^{\mathrm{op}}$
- $T \otimes (\partial^B)^{\mathrm{op}} \subseteq_{bpg} (\partial^A)^{\mathrm{op}} \otimes T$

which witness that T is a *bisimulation* between ∂^A and ∂^B

iv) composing bx: fun with triples

some notation

Given model spaces $\mathbb{A}$, $\mathbb{B}$, and a relation T between A and B, write

$$\mathbb{T} =_{\text{def}} (T, \rhd_T, \lhd_T) : \mathbb{A} \not\leftrightarrows_T \mathbb{B}$$

Given two such, $\mathbb{S} : \mathbb{A} \not\leftrightarrows_S \mathbb{B}$, $\mathbb{T} : \mathbb{B} \not\leftrightarrows_T \mathbb{C}$, how do we construct

$$\mathbb{S} \otimes \mathbb{T} : \mathbb{A} \not\leftrightarrows_U \mathbb{C}$$

Well…

- Take $U =_{\text{def}} S \otimes T$
- Forward propagation: $\rhd_U =_{\text{def}} \rhd_S \otimes \rhd_T$ where

$$(\rhd_S \otimes \rhd_T)(a', \delta_a, (b, s, t)) = (c', (b', s', t'), \delta_c)$$

$$\textit{where } (b', s', \delta_b) =_{\text{def}} \rhd_S (a', \delta_a, s),\ (c', t', \delta_c) =_{\text{def}} \rhd_T (b', \delta_b, t)$$

pause: bx are proof relevant bisimulations

they form a *bicategory*

with a forgetful mapping back to the bicategory **Rel**

v) align, restore, project: reconciling set-based bx

alignment

Given $a : A$, $b : B$, how do we fit them into an fpg square?

That is: how do we construct an element of $a\,((\partial^A)^{\mathrm{op}} \otimes T)\,b$?

This is the *alignment problem*; a solution will be:

- another dependently-typed function;
- taking something simple (a pair) to something complicated (a triple);
- $\otimes$ is defined in terms of Σ-types: so the result type corresponds to an *existential* proposition;
- this is why alignment usually gets solved by a (heuristic) *search* procedure: matching, ...

alignment examples

Zhenjiang: surjectivity of *put*

- state-based update: deltas are trivial (one-point sets)
- get-based consistency: consistency relation has trivial inhabitants
- surjectivity: witness the existential quantifier

Martin: alignment from the initial state

- given (x, y) need a c so that (x, c) is fit for processing by *putl* (or (y, c) by *putr*);
- use default *init* elements: (x_0, c_0), (y_0, c_0) are guaranteed to be aligned; then build up inductively by edit sequences

Tony: alignment via *constructing derivations*

- delta arises by constructing a derivation in the model-space grammar
- consistency witness by constructing derivation in the triple space TGG: *matching* required

align, restore, project

A (rationally-reconstructed) set-based consistency restorer:

- takes an (a', b) pair as input (arbitrary!);
- (align) computes an alignment $a' \; ((\partial^A)^{\mathrm{op}} \otimes T) \; b$, that is a triple (a, δ_a, t);
- (restore) applies forward restorer $\triangleright_T$, yielding another triple (b', t', δ_b);
- (project) returns b'

Remarks:

- vanilla set-based bx don't obviously compose, but now these bx do (we filled in the missing data)
- correctness is automatic! enforced by the type
- alignment is important! (restore) step expects 'good' input
- traceability is important! don't (project): it throws information away

vi) hippocraticness and PutPut

hippocraticness: how to do nothing

Enrich model space $\mathbb{A} =_{\mathrm{def}} (A, \partial^A)$ with *no-op* deltas

$$\iota : \Pi_{a:A}\ \partial^A(a, a)$$

Then $\mathbb{A}$ becomes a *reflexive graph*

What is hippocraticness for $\mathbb{T} =_{\mathrm{def}} (T, \triangleright_T, \triangleleft_T) : \mathbb{A} \rightleftarrows_T \mathbb{B}$?

Demand: for all $a : A, b : B, t : a\, T\, b$,

$$\triangleright_T\ (a, \iota^A_a, t) = (b, t, \iota^B_b)$$

Remark: this is a very strong, *intensional*, definition:

- the resulting value is b itself (classical hippocraticness)
- the resulting delta ι^B_b is itself a no-op
- the resulting consistency witness t is preserved on-the-nose: t is for 'trace'

Lemma: Hippocratic bx are closed under composition

hippocraticness and alignment

If

- a, b are *already* consistent
- should be able to compute a witness t

then

- alignment should return the identity delta ι_a
- restoration preserves that identity, and the witness t

Role for *checkonly* mode:

- compute $checkonly : \Pi_{a:A}\Pi_{b:B}\ Maybe(T(a,b))$
- a better boolean: if consistent, return a witness; otherwise return a dummy token

PutPut

Suppose that deltas compose: that is, the model spaces $\mathbb{A}$, $\mathbb{B}$ are *categories*

Natural to demand that 'restoration respect composition of deltas':

- if

$$\triangleright_T\ (a', \delta_a, t) = (b', t', \delta_b)\ \textit{and}\ \ \triangleright_T\ (a'', \delta_{a'}, t') = (b'', t'', \delta_{b'})$$

- then

$$\triangleright_T\ (a'', (\delta_a \otimes \delta_{a'}), t) = (b'', t'', (\delta_b \otimes \delta_{b'}))$$

Remark: this is not history ignorance

Lemma: PutPut bx are closed under composition

monadicity: is PutPut innocent?

Categories are monadic over graphs…

Question:

can every bx between general model spaces (graphs) be understood as a PutPut bx between the associated freely-generated categories?

vii) why go to all this trouble?

dependent types

Pros:

- no missing information! logical aspects reified as *data*
- type-checking is proof-checking: correctness enforced by typing
- compositionality via… composition of dependently-typed functions
- functions which operate on triples, and… shuffle their components
- type-theory is implementable as a programming language: Agda, Idris, …

Cons: apparently none!

bx as proof-relevant bisimulation

Bisimulation:

- well-studied in theory: process algebra, games, concurrency
- well-implemented in practice: model-checking, games for model-checking
- *proof-relevant* versions have **not** been well-studied
- but: revisit Edinburgh Concurrency Workbench

Bx:

- zoo of competing formalisms
- try to reconcile, unify, explain in well-known terms
- bx as 'bisimulations with traceability' *book-keeping*
- fruitful interplay? relate existing bisim tools for concurrency with bx tools

type theory and (enriched) category theory

Type theory is:

- 'pre-categorical': no commitment to model spaces as categories, but no restriction either
- interpretable in a wide variety of (bi-)categories with suitable, well-behaved structure; we take care to express constructions so as to support this
- analysis in terms of 'types-as-sets': not a necesary restriction

Enriched category theory:

- hom objects (deltas!) *need not* be sets
- suffices to have a $\otimes$ structure on homs… plus a bit more (!): (symmetric) monoidal category $\mathbb{V}$
- develop category theory *relative to* structure in $\mathbb{V}$
- consider model spaces as $\mathbb{V}$-categories: metric spaces (Lawvere), non-determinism (folklore), probabilistic non-determinism (?)
- consistency is now '$\mathbb{V}$-valued'

Spans

Rel is (morally) the bicategory of *spans* in **Set**

Question: does **Bisim**, **Bx** arise as a bicategory of spans?

Consistency as a surface

The alignment structure

$$a' \, (\partial^{A^{\mathrm{op}}} \otimes T) \, b =_{\mathrm{def}} \Sigma_{a:A} \, \partial^{A}(a, a') \times T(a, b)$$

is something like

> *the weighted sum (integral) over all possible local changes in a of the 'values' of the consistency relation*

Consider: T valued not logically, but numerically

A differential geometry of software development…

Questions?