Distributing Commas, and the Monad of Anchored Spans

Michael Johnson
Departments of Mathematics and Computing, Macquarie University
Robert Rosebrugh
Department of Mathematics and Computer Science
Mount Allison University

Abstract

Spans are pairs of arrows with a common domain. Despite their symmetry, spans are frequently viewed as oriented transitions from one of the codomains to the other codomain. The transition along an oriented span might be thought of as transitioning backwards along the first arrow (sometimes called 'leftwards') and then, having reached the common domain, forwards along the second arrow (sometimes called 'rightwards'). Rightwards transitions and their compositions are well-understood. Similarly, leftwards transitions and their compositions can be studied in detail. And then, with a little hand-waving, a span is 'just' the composite of two well-understood transitions — the first leftwards, and the second rightwards.

In this paper we note that careful treatment of the sources, targets and compositions of leftwards transitions can be usefully captured as a monad L built using a comma category construction. Similarly the sources, targets and compositions of rightwards transitions form a monad R, also built using a comma category construction. Our main result is the development of a distributive law, in the sense of Beck [3] but only up to isomorphism, distributing L over R. Such a distributive law makes RL a monad, the monad of anchored spans, thus removing the hand-waving referred to above, and establishing a precise calculus for span-based transitions.

As an illustration of the applicability of this analysis we use the new monad RL to recast and strengthen a result in the study of databases and the use of lenses for view updates.

1 Introduction

1.1 Set-based bidirectional transformations

There is an important distinction among extant bidirectional transformations between those that are "set-based" and those that are "category-based" (the latter are sometimes also called "delta-based"). This paper analyses the origins of that distinction and lays the mathematical foundations for a calculus integrating both points of view along with an often implicit point of view in which deltas sometimes have a "preferred" direction.

So, what are these set-based and category-based transformations, and how did the distinction arise?

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

To quote from the Call for Papers: "Bidirectional transformations (Bx) are a mechanism for maintaining the consistency of at least two related sources of information. Such sources can be relational databases, software models and code, or any other document following standard or ad-hoc formats."

A fundamental question in bidirectional transformations is: What are the *state spaces* of the sources among which consistency needs to be maintained? After all, consistency only needs to be worked on when one source changes state, so a careful analysis of permissible state changes is important.

This analysis will be important whatever the nature of the sources, but for ease of explication we will focus at first on an example. Let us consider relational databases. Recall that state spaces of systems are most often represented by graphs whose nodes are states and whose arrows represent transitions among the states.

Consider the state space of a relational database. Nodes in the state space, states, are just snapshots of the database at a moment in time — all of the data, stored in the database, in its structured form, at that moment.

Typically, with a database, one can transition (update the database) from any snapshot to any other. So, the state space could be all possible snapshots with an arrow between every pair of snapshots. (This kind of state space is sometimes called a "chaotic" or "co-discrete" category. A *co-discrete category* is a category with exactly one arrow between each ordered pair of objects.)

This kind of state-space, a co-discrete state-space, is a perfectly reasonable analysis of database updating. It comes naturally from focusing on the states, and from noticing that there is always an update that will lead from any state S to any other state S'. After all, when one can transition between any two states, why keep track of arrows that tell you that you can do that? All that one needs to know is what the new state S' is. And, no matter what the current state S might be, the database can transition to that new state S'.

So, it is natural to do away with the arrows, consider simply the set of states, and plan one's Bx assuming that any state S might be changed to any other state S'.

This is the foundation of set-based bidirectional transformations.

Important, pioneering work on set-based bidirectional transformations was carried out in this community by Hoffmann, Pierce et al, and by Stevens et al, among others.

1.2 Category-based bidirectional transformations

Other workers chose to analyse the state spaces differently.

Pioneering work by Diskin [4] et al noted that while the states are vitally important (and on one analysis, that of the database user, they are all that really matters) one might expect very different side-effects when one updates from S to S' in very different ways, and so it might be important to distinguish different updates leading from S to S'.

For example, S' might be obtainable from S by inserting a single row in a single table. Let's call that transition α. But there are many other ways to transition from S to S'. To take an extreme example, let's call it β, one might in a single update delete all of the contents of the database state S, and then insert into the resultant empty database all of the contents of the database state S'. Both α and β are transitions from S to S', so if one wishes to distinguish them, then a set-based, or indeed a co-discrete category based, description of the state space will not suffice. While the states themselves remain pre-eminently important, the transitions need to be tracked in detail, along with their compositions (one transition followed by another) and the state space becomes a category (we assume that the reader is familiar not just with set theory, but also with category theory).

Incidentally, much of the former controversy surrounding put-put laws arises from the different analyses of set-based and category-based bidirectional transformations. If a state space does not distinguish α from β then the result of maintaining consistency with α (a single row insert) has to be the same as the result of maintaining consistency with β and the latter could, on breaking down the update result in synchronising with the empty database state, and then synchronising with the update from that state to S'. On that analysis the put-put law (which says that an update can be synchronized only at the end, or at any intermediate step and then resynchronized at the end, with the same outcome in either case) is a very strong, probably unreasonably strong, requirement. Alternatively, if α and β are different updates then the put-put law says nothing *a priori* about their synchronizations and is a much less stringent requirement.

1.3 Information-order-based bidirectional transformations

There is yet another way in which one might analyse the state space of a relational database.

A basic transition might be inserting one or more new rows in some table(s). So we might consider a state-space with the same snapshots as before, but with an arrow $S \longrightarrow S'$ just when S' is obtained from S by inserting

rows. This is in fact the "information order" — the arrow $S \longrightarrow S'$ can be thought of as the inclusion of the snapshot S in the bigger (more information stored) snapshot S'.

Notice that this state space has arrows corresponding to inserts, and we will below call the inserts "rightwards" transitions. But those very same arrows correspond to deletes if we transition them "leftwards", backwards, along the arrow (one can move from S' to S by deleting the relevant rows).

The information order provides yet another potential state space for a relational database, and is well-understood by database theorists. It has the added complication that arrows can be transitioned in both directions. But it has the advantage of separating out two distinct kinds of transition, the rightwards, insert, transitions, and leftwards, delete, transitions, and these can be analysed separately.

It is of great convenience that transitions of the same kind — all leftwards or all rightwards — have very good properties: For example, an insert followed by another insert is definitely just an insert, and the corresponding "monotonic" (rightwards) put-put law for a Bx using multiple inserts has never been controversial. Similarly for multiple leftwards transitions.

Of course an arbitrary update can involve some mixture of leftwards and rightwards transitions, and the main technical content of this paper is the development of a detailed calculus for mixed transitions.

1.4 Bx state spaces

We have seen that there are (at least) three fundamental approaches to state spaces for relational databases. These approaches apply more generally to various sources for bidirectional transformations.

1. In many systems we can concentrate on the states, assume that we can transition from any state S to any other state S', and view the state space either as a set over which we might build a set-based Bx (for example a well-behaved lens), or equivalently as a co-discrete category (which explicitly says that there is a single arrow $S \longrightarrow S'$ for any two states S and S').

2. Alternatively, we can attempt to distinguish among different ways of updating $S \longrightarrow S'$, different "deltas", and view the state space as a category over which we might build a category-based Bx (for example, a delta-lens).

3. And very often among the categories of state spaces there is a "preferred direction" for arrows that can be identified in which case the state space as a category can be simplified by showing only the preferred direction, with arbitrary transitions recovered as composites of rightwards (normal direction along an arrow) and leftwards (backwards along an arrow) transitions.

Naturally, if a particular application lends itself well to set-based analysis then a set-based Bx will suffice.

Frequently however, knowledge of the deltas, when available, is sufficiently advantageous for us to want to build a category-based Bx. One advantage of the co-discrete representation of set-based bidirectional transformations is that set-based results can be derived from category-based results since co-discrete categories are merely a special case of categories.

Furthermore, if the application presents a natural "preferred" order of transition then the third approach might be used. Such information order and related situations are so common that this approach has been used implicitly for some time. The main goal of this paper is to develop the machinery that mathematically underlies this approach, and to show how it incorporates the other two approaches so that ordinary category-based, or indeed, via co-discrete categories, ordinary set-based bidirectional transformations, can be recovered as special cases.

1.5 Lenses as algebras for monads

In earlier work [13, 14, 9] the authors, sometimes with their colleague Wood, have shown that asymmetric lenses of various kinds are simply algebras for certain well-understood monads. This gives a strong, unified, and algebraic treatment of lenses, and brings to bear a wide-range of highly-developed mathematical tools.

The monads involved are all in some sense state space dependent.

For asymmetric lenses we will call the state spaces of the two systems $\mathbf{S}$ and $\mathbf{V}$, and suppose given a Get function or functor $G : \mathbf{S} \longrightarrow \mathbf{V}$.

In the set-based case the monad, $\Delta\Sigma$, captures completely the notion that all that is needed for a Put operation is a given state $S \in \mathbf{S}$ and a new state $V' \in \mathbf{V}$ with no necessary relationship between GS and V'.

In the category-based case the monad $(-, 1_V)$ (which we will rename here as R because it will be used to model the rightwards transitions), captures completely the notion that a Put operation depends on a given state $S \in \mathbf{S}$ and an arrow of the state space $\mathbf{V}$ of the form $GS \longrightarrow\!\!\!\!\!\rightarrow V'$.

In this paper we will show how to construct analogously a monad L which models the leftwards transitions, and which has as algebras lenses for the backwards (in database terms, delete) updates. A Put operation for leftwards transitions depends on a given state $S \in \mathbf{S}$ and an arrow of the state space $\mathbf{V}$ of the form $V' \longrightarrow\!\!\!\!\!\rightarrow GS$. (Notice that the update from GS to V' transitions *leftwards* along the arrow to reach V'.)

Most importantly in this paper we exhibit a distributive law, in the sense of Beck [3], which relates R and L and provides a new monad RL which captures completely the notion that for the general third case a Put operation should depend on a given state $S \in \mathbf{S}$ and an arbitrary composition of leftwards and rightwards arrows starting from GS and ending, after zig-zagging, at some $V' \in \mathbf{V}$. This is our main technical result.

Algebras for the new monad RL are lenses that synchronise with arbitrary strings of leftwards and rightwards transitions in $\mathbf{V}$.

1.6 Plan of this paper's technical content

The paper shows that the category theoretic structure of arrows rightwards from GS can be captured as a monad R. Similarly, the category theoretic structure of arrows leftwards from GS can be captured as a monad L. There are some important new technical advantages in this, and some important new implications for the lens community, but the basic ideas of these two monads are not new (first appearing in the paper of Street [17]).

Database theorists often work with the information order. But an arbitrary database transition involves inserts *and* deletes — a mixing of L and R transitions in the information order. Until now this has been done with handwaving — the R monad tells us all about inserts, the L monad tells us all about deletes, so we mix transitions together doing say a delete followed by an insert as a general transition which could be drawn as a "span" $S \longleftarrow S' \longrightarrow\!\!\!\!\!\rightarrow S''$ (get from S to S'' by deleting some rows from S to get to S', and then inserting some rows into S' to get to S''). And of course we want to distinguish this from other ways of getting from S to S'', perhaps $S \longleftarrow T \longrightarrow\!\!\!\!\!\rightarrow S''$ where T may be very different from S'. But we've moved from the monads that tell us everything we need to know about rightwards transitions and their interactions, and leftwards transitions and their interactions, to a vague idea of mixing such transitions together.

The main point of the paper is the discovery of a previously unobserved distributive law between L and R which, like all distributive laws, gives a new monad RL, and this composite monad precisely captures the calculus of these spans ("calculus" meaning we can calculate with it using routine procedures (various versions of μ and η below) determining algorithmically how all possible mixings, zigzags, etc, interact, which ones are equivalent to one another, and so on).

It should perhaps be noted here that the spans in this paper are mathematically the same as, but largely semantically otherwise unrelated to, the authors' use of equivalence classes of spans to describe *symmetric* lenses of various kinds in [10, 11].

The plan of this paper is fairly straightforward. In Section 2 we introduce in detail the two monads R and L. In Section 3 we develop the distributive law, slightly generalising the work of Beck as it is in fact a pseudo-distributive-law. (Recall that in category theory, when an axiom is weakened by requiring only a coherent isomorphism rather than an equality, the resulting notion is given the prefix "pseudo-". The coherency of the isomorphisms involved can be daunting, but in this paper the isomorphisms arise from universal properties and thus coherency is automatic and we will say no more about it.) In addition in Section 3 we display explicitly the basic operations of the new monad RL. In Section 4 we study the algebras for such monads, noting that these algebras are (generalised) lenses. And how useful might all this be? Well, Proposition 3 in Section 4 is an example of a strong new result that couldn't even be stated without the new monad RL.

We hope that having these things in mind might be some help in seeing through the technical details in the mathematics that follows.

1.7 Summarising the introduction

State spaces with reversible transitions have been the source of a number of confusions. The fact that every state is accessible from every other state in the same connected component, has sometimes led to researchers ignoring transitions (the set-based state spaces referred to above). Conversely, if instead of ignoring transitions *all* the transitions are explicitly included in the state space, then in many applications we are failing to distinguish two

different types of transition, the "rightwards" and the "leftwards". The resulting plethora of transitions of mixed types can seriously complicate any analysis.

With the two monads L and R, state spaces with reversible transitions can be managed effectively. Rather than constructing state spaces in which transitions come in pairs $S \longrightarrow S'$ and $S' \longrightarrow S$, we include only one of each pair (there is usually a "preferred" direction which can be the one included). Then the monad R is used for analyses and constructions using the categorical structure of the preferred transitions. Similarly the monad L can be used for analyses and constructions using the reverse transitions. And arbitrary composites of transitions can be broken down into "zig-zags" of L and R transitions, and in many cases into *spans*, L transitions followed by R transitions.

However, at this point we have come to the "hand-waving". What does it really mean mathematically to deal with zig-zags of L and R transitions? What does it mean to deal with an L transition followed by an R transition? And what is the calculus of mixed L and R transitions?

The main point of this paper is to show how under a modest hypothesis (the existence of pullbacks in the state space category $\mathbf{V}$) there is, up to isomorphism, a distributive law [3] between the two comma category monads R and L. In the presence of such a distributive law, the composite RL becomes a monad — the monad of anchored spans. The monad of anchored spans, including its relationship to L and R and the calculus it generates, answers all the questions in the preceding paragraph in a precise way. It eliminates the "hand-waving" and replaces it with a proper mathematical treatment of the interactions of R and L.

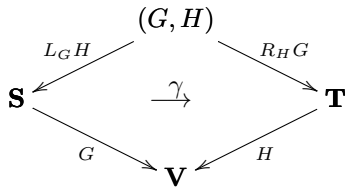
2 Two comma category monads

For basic category theoretic notions readers are referred to any of the standard texts, including for example [2] and [16]. At present the mathematical parts of this paper have been written succinctly, frequently assuming that readers have well-developed category theoretic backgrounds.

Given two functors with common codomain $\mathbf{V}$, say $\mathbf{S} \xrightarrow{G} \mathbf{V} \xleftarrow{H} \mathbf{T}$, the comma category (G, H) was introduced in the thesis of F.W. Lawvere. It has as objects triples (s, t, a) where s is an object of $\mathbf{S}$, t is an object of $\mathbf{T}$, and $a : Gs \longrightarrow Ht$ is an arrow of $\mathbf{V}$. The arrows of the comma category are, as one would expect, given by an arrow $p : s \longrightarrow s'$ of $\mathbf{S}$ and an arrow $q : t \longrightarrow t'$ of $\mathbf{T}$ which make the corresponding square in $\mathbf{V}$ commute (so for an arrow (p, q) from (s, t, a) to (s', t', a') we have $(Hq)a = a'(Gp)$ in $\mathbf{V}$) .

The comma category has evident projections to $\mathbf{S}$ and $\mathbf{T}$ shown in the figure below.

We denote the comma category and its projections as follows:



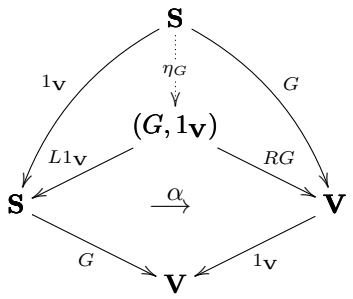
Where possible we will suppress the subscripts on projections. This is especially desirable if the subscript is itself an identity functor, and this situation arises frequently since the comma categories we will consider below will usually have the identity functor on $\mathbf{V}$ for one of either G or H.

The central arrow, γ in the figure, is included because the comma category has not just projections LH and RG, but also a natural transformation $\gamma : G(LH) \longrightarrow H(RG)$ (because each object (s, t, a) of (G, H) is in natural correspondence with an appropriate arrow, a itself, of $\mathbf{V}$). Indeed, the comma category is a kind of 2-categorical limit — it is universal among spans from $\mathbf{S}$ to $\mathbf{T}$ with an inscribed natural transformation.

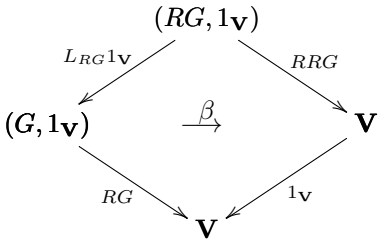
Explicitly, the universality of the comma category is given by the following. Each functor $F : \mathbf{X} \longrightarrow (G, H)$ corresponds bijectively with a triple (K, L, φ) where $K : \mathbf{X} \longrightarrow \mathbf{S}$, $L : \mathbf{X} \longrightarrow \mathbf{T}$ and $\varphi : GK \longrightarrow HL$ (with the correspondence given by composing each of LH, RG and γ with F).

Using this universal propertly we establish some further notation. Write η_G for the functor corresponding to

the triple $(1_{\mathbf{V}}, G, 1_G) : \mathbf{S} \longrightarrow (G, 1_{\mathbf{V}})$ as in



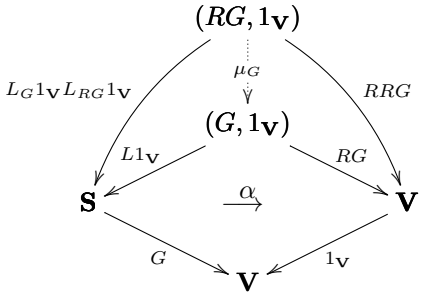
We denote the iterated comma category and projections



Write $\mu_G : (RG, 1_{\mathbf{V}}) \longrightarrow (G, 1_{\mathbf{V}})$, for the functor corresponding to the triple $(L_G 1_{\mathbf{V}} \cdot L_{RG} 1_{\mathbf{V}}, RRG, \beta(\alpha L_{RG} 1_{\mathbf{V}}))$, noting that the transformation is

$$G \cdot L_G 1_{\mathbf{V}} \cdot L_{RG} 1_{\mathbf{V}} \xrightarrow{\alpha L_{RG} 1_{\mathbf{V}}} RGL_{RG} 1_{\mathbf{V}} \xrightarrow{\beta} RRG$$

in



The assignment $G \mapsto RG$ (remembering that RG is more completely $R_{1_{\mathbf{V}}} G$, but as noted above we frequently suppress such subscripts) defines (on objects) the functor part of a monad on the slice category $\mathbf{cat}/\mathbf{V}$. The η_G and μ_G defined above are the unit and multiplication for this monad at an object G. The action of the functor on arrows, and the associativity and identity axioms for the monad, all follow from the facts that η and μ are defined by universal properties.

Similarly, $H \mapsto LH = L_{1_{\mathbf{V}}} H$ defines a monad L on $\mathbf{cat}/\mathbf{V}$.

3 The distributive law

For the remainder of this paper we assume that $\mathbf{V}$ has pullbacks.

Consider the monads R and L defined at the end of the previous section. We will denote the units and multiplications for R and L by η^R, η^L, μ^R and μ^L.

As we will show, there is a distributive law

$$LR \xrightarrow{\lambda} RL$$

between these two monads and consequently, the composite RL is a monad.

We begin by describing the composites LR and RL.

If $G : \mathbf{S} \longrightarrow \mathbf{V}$ then the domain of the functor RG is the comma category $(G, 1_{\mathbf{V}})$ and so has as objects arrows from $\mathbf{V}$ indexed by objects of $\mathbf{S}$ of the form $Gs \xrightarrow{a} v$. The important projection RG (important because it is the one which shows us the effect on G of the monad functor R) gives $RG(Gs \xrightarrow{a} v) = v$. The projection RG is also important because when we apply L to RG, L will build on the projected value v. Thus applying the functor L to RG gives a functor $LRG : (1_{\mathbf{V}}, RG) \longrightarrow \mathbf{V}$ whose domain has objects of the form $Gs \xrightarrow{a} v \xleftarrow{b} v'$, that is cospans (a, b) from Gs to v'. Moreover $LR(G)(a, b) = v'$.

On the other hand, the domain of the functor $LG : (1_{\mathbf{V}}, G) \longrightarrow \mathbf{V}$ has objects of the form $w \xrightarrow{c} Gs$ in $\mathbf{V}$ and $LG(w \xrightarrow{c} Gs) = w$. Applying the functor R to LG gives a functor $RL(G) : (LG, 1_{\mathbf{V}}) \longrightarrow \mathbf{V}$ whose domain has objects of the form $Gs \xleftarrow{c} w \xrightarrow{d} w'$, that is spans (c, d) from Gs to w', and $RL(G)(c, d) = w'$.

Now we can define the G'th component of λ. It is the (pseudo-)functor (over $\mathbf{V}$) $\lambda_G : (1_{\mathbf{V}}, RG) \longrightarrow (LG, 1_{\mathbf{V}})$ defined at an object $Gs \xrightarrow{a} v \xleftarrow{b} v'$ of the domain of LRG by taking the pullback in $\mathbf{V}$ (and on arrows using the induced maps). Thus $\lambda_G(a, b)$ is a span in $\mathbf{V}$ from Gs to v'. The value of the functor $LR(G)$ at (a, b) is v' and the value of $RL(G)$ at the pullback of the cospan (a, b) is also v'. Thus, $RL(G)(\lambda_G(a, b)) = LR(G)(a, b)$ on the nose.

To see that λ is natural, suppose that $G' : \mathbf{S}' \longrightarrow \mathbf{V}$ and the functor $F : \mathbf{S} \longrightarrow \mathbf{S}'$ defines an arrow from G to G' in $\mathbf{cat}/\mathbf{V}$, that is $G'F = G$. We need to show that $\lambda_{G'} LR(F) = RL(F)\lambda_G$. Thus the following square of functors must commute (over $\mathbf{V}$):

$$\begin{array}{ccc} (1_{\mathbf{V}}, RG) & \xrightarrow{\lambda_G} & (LG, 1_{\mathbf{V}}) \\ {\scriptstyle LR(F)}\downarrow & & \downarrow{\scriptstyle RL(F)} \\ (1_{\mathbf{V}}, RG') & \xrightarrow[\lambda_{G'}]{} & (LG', 1_{\mathbf{V}}) \end{array}$$

where $LR(F)$ is the functor $(1_{\mathbf{V}}, R(F))$ which defines a morphism $LRG \longrightarrow LRG'$. The square does commute, up to isomorphism, since the effect of $LR(F)$ on a cospan $Gs \xrightarrow{a} v \xleftarrow{b} v'$ (an object of $(1_{\mathbf{V}}, RG)$) is actually the same cospan $G'Fs = Gs \xrightarrow{a} v \xleftarrow{b} v'$ and $\lambda_{G'}$ computes a pullback span, $G'Fs \xleftarrow{c} w \xrightarrow{d} v'$. On the other hand $RL(F)\lambda_G$ applied to the (same) cospan (a, b) computes a pullback span, $Gs \xleftarrow{c'} w' \xrightarrow{d'} v'$, and then applies $RL(F)$ which likewise leaves the span unchanged, and so isomorphic to (c, d).

Next we consider the distributive law equations [3]. One equation involving the units is:

$$\begin{array}{ccc} & L & \\ {\scriptstyle L\eta^R}\swarrow & & \searrow{\scriptstyle \eta^R L} \\ LR & \xrightarrow[\lambda]{} & RL \end{array}$$

At a functor G, the left hand side of the triangle $L\eta^R$ applies to an object $v \xrightarrow{a} Gs$ of the domain of LG and the result is the cospan $v \xrightarrow{a} Gs \xleftarrow{1} Gs$. Application of λ_G gives a pullback span which we can choose to be $v \xleftarrow{1} v \xrightarrow{a} Gs$. On the other hand $\eta^R L$ applied to $v \xrightarrow{a} Gs$ is exactly the same span $v \xleftarrow{1} v \xrightarrow{a} Gs$.

The other unit equation is:

$$\begin{array}{ccc} & R & \\ {\scriptstyle \eta^L R}\swarrow & & \searrow{\scriptstyle R\eta^L} \\ LR & \xrightarrow[\lambda]{} & RL \end{array}$$

At a functor G, the left hand side $\eta^L R$ applies to an object $Gs \xrightarrow{b} v$ of the domain of RG and the result is the cospan $Gs \xrightarrow{b} v \xleftarrow{1} v$. Application of λ_G gives a pullback span which we choose to be $Gs \xleftarrow{1} Gs \xrightarrow{b} v$. Again, this is the same as the effect of $R\eta^L$ on $Gs \xrightarrow{b} v$.

We consider only one of the equations involving multiplications; the other is similar. The equation we consider is:

$$\begin{array}{ccccc} LRR & \xrightarrow{\lambda R} & RLR & \xrightarrow{R\lambda} & RRL \\ {\scriptstyle L\mu^R}\downarrow & & & & \downarrow{\scriptstyle \mu^R L} \\ LR & & \xrightarrow[\lambda]{} & & RL \end{array}$$

Again, we look at the equation in the domain and at an object G. A typical object of the domain of $LRR(G)$ is an extended cospan of the form $Gs \xrightarrow{a} v \xrightarrow{a'} v' \xleftarrow{b} w$. Since $\mu^R(a, a')$ is simply the composite $Gs \xrightarrow{a'a} v'$, we see that $\lambda_G(L\mu^R(G))(a, a', b)$ is a pullback span $Gs \xleftarrow{c} u \xrightarrow{d} w$ of b along $a'a$. On the other hand, $\lambda_G R(G)$ applied to (a, a', b) computes a pullback span of b along a' with result $Gs \xrightarrow{a} v \xleftarrow{c'} u' \xrightarrow{d'} w$. Applying $RG\lambda_G$ computes a pullback of c' along a with result $Gs \xleftarrow{c''} u'' \xrightarrow{d''} u' \xrightarrow{d'} w$. Applying $\mu^R_G LG$ composes d'' and d' giving the span $Gs \xleftarrow{c''} u'' \xrightarrow{d''d'} w$ which is isomorphic to $Gs \xleftarrow{c} u \xrightarrow{d} w$.

The preceding considerations give:

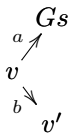
Proposition 1 *The transformation $LR \xrightarrow{\lambda} RL$ is a (pseudo-)distributive law.* ∎

Using this proposition, and with minor modifications of the work of Beck [3] to take account of the isomorphisms in the pseudo-naturality squares and in the equations involving multiplications, we obtain:

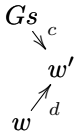
Proposition 2 *The composite functor RL on $\mathbf{cat}/\mathbf{V}$ is a monad, the monad of spans, with $\mu^{RL} = \mu^R L \cdot RR\mu^L \cdot R\lambda L$ and $\eta^{RL} = \eta^R L \cdot \eta^L$.* ∎

It is convenient for later work to introduce some notation and use it to describe the unit and multiplication for the composed monad RL.

As noted above, for $G : \mathbf{S} \longrightarrow \mathbf{V}$, the domain of RLG is a comma category whose objects are certain spans in $\mathbf{V}$. Let us denote them

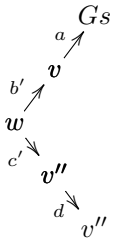


Objects of the domain of LRG are depicted:



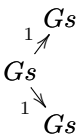
An object of the domain of $RLRLG$ is a "zig-zag":

The effect of the multiplication for RL on the zig-zag is to first form the span:



where w is a pullback of b and c, and then to compose each of the legs using the multiplication μ^L for the top leg (which is an instance of LLG), and the multiplication μ^R for the bottom leg (which is by then an instance of $RRLG$).

The unit for RL simply forms spans of identity arrows, so each object s of $\mathbf{S}$ yields a span



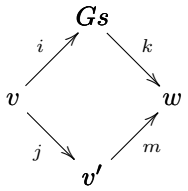
4 Algebras

To illustrate the benefits of the distributive law and the resultant monad RL we present here a significantly stronger version, made possible by the use of RL, of a result from [8]. That paper was a study of two kinds of generalised lenses, one for preferred transitions and one for their reverses. It showed that the two lenses are respectively an R-algebra and an L-algebra and conversely. The main interest was how they should interact via a "mixed put-put law" called condition (*) below. The new result shows that having two such lenses satisfying condition (*) is equivalent to having an RL-algebra. In other words, if we can update inserts (R transitions) and deletes (L transitions) and those updates satisfy condition (*), then we can update spans — arbitrary compositions of R and L transitions.

We first describe a condition that is essentially the Beck-Chevalley condition for our purposes. (It does not matter if the reader is not familiar with other instances of the Beck-Chevalley condition.)

Suppose that $\mathbf{S} \xrightarrow{G} \mathbf{V}$ and that $RG \xrightarrow{r} G$ and $LG \xrightarrow{l} G$ are R- and L-algebra structures. If $Gs \xrightarrow{k} w$ is an object of $(G, 1_{\mathbf{V}})$, we denote its image under r by $r(s,k)$. Similarly if $v \xrightarrow{i} Gs$ is an object of $(1_{\mathbf{V}}, G)$, we denote its image under l by $l(i,s)$. Since r and l are arrows in $\mathbf{cat}/\mathbf{V}$, $G(r(s,k)) = w$ and $G(l(i,s)) = v$. We say that *condition (*) is satisfied* if

for any object s in $\mathbf{S}$ and any pullback (in $\mathbf{V}$)



it is the case that $r(l(i,s),j) \cong l(m, r(s,k))$.

Proposition 3 *If $RLG \xrightarrow{\xi} G$ is an RL-algebra, then $r = \xi R\eta^L G$ and $l = \xi\eta^R LG$ define R- and L-algebras, respectively, that satisfy condition (*). Conversely, suppose $RG \xrightarrow{r} G$ and $LG \xrightarrow{l} G$ are R- and L-algebra structures satisfying (*). Define $\xi : RLG \longrightarrow G$ (on objects) by $\xi(Gs \xleftarrow{i} v \xrightarrow{j} v') = r(l(i,s),j)$. Then ξ determines an RL-algebra structure on G.*

Proof. Suppose $RLG \xrightarrow{\xi} G$ is an RL-algebra, r and l are defined as in the statement, and the square in condition (*) is a pullback in $\mathbf{V}$. The definitions of r and l amount, on objects, to $r(Gs \xrightarrow{a} v) = \xi(Gs \xleftarrow{1} Gs \xrightarrow{a} v)$ and $l(v \xrightarrow{a} Gs) = \xi(Gs \xleftarrow{a} v \xrightarrow{1} v)$. For clarity we will sometimes suppress the objects of $\mathbf{V}$, (v, Gs, etc), in what follows. Then, using the notation from the pullback square and noting that $G(\xi(\xleftarrow{1} \xrightarrow{k})) = w$,

$$\begin{aligned}
l(m, r(s,k)) &= l(m, \xi(\xleftarrow{1} \xrightarrow{k})) \\
&= \xi(G(\xi(\xleftarrow{1} \xrightarrow{k})) \xleftarrow{m} \xrightarrow{1}) \\
&= \xi \cdot RL(\xi)(\xleftarrow{1} \xrightarrow{k} \xleftarrow{m} \xrightarrow{1}) \\
&= \xi\mu_G(\xleftarrow{1} \xrightarrow{k} \xleftarrow{m} \xrightarrow{1}) \\
&= \xi(\xleftarrow{1} \xleftarrow{i} \xrightarrow{j} \xrightarrow{1}) \\
&= \xi(\xleftarrow{i} \xrightarrow{j}) \\
&= r(\xi(\xleftarrow{i} \xrightarrow{1}), j) \\
&= r(l(i,s), j)
\end{aligned}$$

(where the fourth equality uses the associative law for the RL-algebra ξ) which proves condition (*).

To check that r and l satisfy the algebra axioms for R and L respectively is routine.

Conversely, suppose that r and l are R- and L-algebras respectively, and that they satisfy condition (*). Suppose that ξ is defined on objects as in the statement. Notice that it is straightforward to extend the definition of ξ to arrows of (the domain of) RLG.

For the RL-algebra associative law we need to verify that for a "zig-zag" $W = Gs \xleftarrow{x} \xrightarrow{y} \xleftarrow{z} \xrightarrow{w}$ starting from Gs, say, that $\xi RL\xi(W) = \xi\mu_G(W)$. Now $RL(\xi)$ applies ξ to $\xleftarrow{x} \xrightarrow{y}$ (while carrying along z and w) giving an object G-over the codomain of z, and ξ applies to this result along with z and w. So using the definition of ξ above, and writing $\xleftarrow{a} \xrightarrow{b}$ for the pullback of $\xrightarrow{y} \xleftarrow{z}$,

$$\begin{aligned}
\xi RL(\xi)(W) &= \xi(G(r(l(x,s),y)) \xleftarrow{z} \xrightarrow{w}) \\
&= r(l(z, r(l(x,s),y)), w) \\
&= r(r(l(a, l(x,s)), b), w) \\
&= r(r(l(xa, s), b), w) \\
&= r(l(xa, s), wb) \\
&= \xi(\xleftarrow{xa} \xrightarrow{wb}) \\
&= \xi(\mu_G(W))
\end{aligned}$$

in which the third equation is an application of property (*) and the fourth and fifth equations use the associative laws of the L-algebra and the R-algebra respectively.

The RL-algebra identity law for ξ is straightforward noting the definition of η^{RL} in terms of η^R and η^L. ∎

5 Related work

In the development of a 2-categorical treatment of the Yoneda Lemma, Ross Street [17] studied monads equivalent to L and R, and also a 'composite' monad M, *not* equivalent to the composite studied here. The present authors, and their colleague Wood, introduced L and R as part of an on-going study of the use of monads in the study of generalised lenses [14]. That work, like much of the earlier work of Johnson and Rosebrugh starting from [6], studied inserts and deletes in isolation, and depended upon the presumption that these could then be reintegrated as spans. A theoretical 2-categorical analysis of spans of models was carried out in [12], but it has had little practical application to date. More recently, a search for a 'mixed put-put law' [8] led the authors to the discovery

of the distributive law reported here, and hence to the new composite monad RL and the corresponding calculus of mixed transitions.

In the realm of database state spaces it seems that most authors have taken the set-based state space approach. We have argued elsewhere [7] that view updating has been limited unnecessarily to constant complement updating [1] because of the failure to treat transitions as first class citizens. One notable exception is the insightful work of Hegner [5] which introduced an information order on the set of states. This order corresponds precisely to choosing to include insert transitions in the state space, but to leave delete transitions out (to be recovered as reversed insert transitions as described in Section 1 above).

Spans have a wide variety of applications, and so have long been studied category theoretically. Given a category $\mathbf{C}$ with pullbacks there is a bicategory $\mathsf{span}\mathbf{C}$ with the same objects as $\mathbf{C}$, with spans of arrows from $\mathbf{C}$ as arrows, and with composition given by pullback, and most treatments take this point of view. Such spans are oriented, despite their symmetry, by definition, since they are arrows of a bicategory. To the authors' knowledge, the monad of anchored spans presented here, including its relationship to the comma category monads L and R, is entirely new. In addition it has noteworthy and desirable differences from earlier treatments because the orientation of spans comes from the fibering over $\mathbf{V}$, and because, in the manner of comma categories, spans appear as objects rather than as arrows.

Meanwhile spans have also had a wide range of applications among researches into bidirectional transformations. In particular the graph transformation community has used spans for many years and the recent paper of Orejas et al [15] makes use of spans of updates in a manner closely related to the current paper. We leave to future work the exploration of the possible applications of our developments in those areas.

6 Conclusions

The main findings from this paper are

1. By explicitly working on $\mathbf{cat}/\mathbf{V}$ the two comma categories $(G, 1_{\mathbf{V}})$ and $(1_{\mathbf{V}}, G)$ can be fibred over $\mathbf{V}$ and so sources and targets can be tracked, resulting in monads R and L respectively. Those monads capture the category theoretic structure of arrows out of images of G and into images of G respectively, and, being built from those comma categories, they lift arrows of $\mathbf{V}$ to objects of RG and LG (fibred over $\mathbf{V}$).

2. There is a pseudo-distributive law between R and L, and so RL is itself a monad, the monad of anchored spans. Furthermore the tracking of sources and targets, provided by the fibring, orients the spans as spans from images of G (the "anchoring"). The monad RL captures the category theoretic structure of spans from images of G, and, being built from iterated comma categories, lifts spans in $\mathbf{V}$ from images of G to objects of RLG (fibred over $\mathbf{V}$).

3. Having an RL-algebra is equivalent to having a pair of algebras (one R and one L) satisfying condition (*).

The first finding shows that we have found the right context in which to work with comma categories for a variety of state space based applications. The second finding solves the long standing problem of making mathematically precise the composition of preferred and reversed transitions (eliminating the "hand-waving"). The third finding neatly illustrates the extra power available when spans of transitions can be dealt with as single objects.

7 Acknowledgements

The authors are grateful for the support of the Australian Research Council and NSERC Canada, and for insightful suggestions from anonymous referees.

References

[1] Bancilhon, F. and Spyratos, N. (1981) Update Semantics of Relational Views, *ACM Trans. Database Syst.* **6**, 557–575.

[2] Barr, M. and Wells, C. (1995) *Category Theory for Computing Science.* Prentice-Hall.

[3] Beck, J. M. (1969) Distributive Laws. *Seminar on Triples and Categorical Homology Theory* Lecture Notes in Math. **80**, 95–112. Available in Reprints in *Theory Appl. Categ.* **18** (2008).

[4] Zinovy Diskin, Yingfei Xiong, Krzysztof Czarnecki (2011), From State- to Delta-Based Bidirectional Model Transformations: the Asymmetric Case, *Journal of Object Technology* **10**, 6:1–25, doi:10.5381/jot.2011.10.1.a6

[5] Hegner, S. J. (2004) An Order-Based Theory of Updates for Closed Database Views. *Ann. Math. Artif. Intell.* **40**, 63–125.

[6] Johnson, M. and Rosebrugh, R. (2001) View Updatability Based on the Models of a Formal Specification. *Proceedings of Formal Methods Europe 2001*, Lecture Notes in Comp. Sci. **2021**, 534–549.

[7] Johnson, M. and Rosebrugh, R. (2007) Fibrations and Universal View Updatability. *Theoret. Comput. Sci.* **388**, 109–129.

[8] Johnson, M. and Rosebrugh, R. (2012) Lens Put-Put Laws: Monotonic and Mixed. *Electronic Communications of the EASST*, **49**, 13pp.

[9] Johnson, M. and Rosebrugh, R. (2013) Delta Lenses and Fibrations. *Electronic Communications of the EASST*, **57**, 18pp.

[10] Johnson, M. and Rosebrugh, R. (2014) Spans of Lenses. *CEUR Proceedings*, **1133**, 112–118.

[11] Johnson, M. and Rosebrugh, R. (2015) Spans of Delta Lenses. To appear *CEUR Proceedings*.

[12] Johnson, M., Rosebrugh, R. and Wood, R. J. (2002) Entity-Relationship-Attribute Designs and Sketches. *Theory Appl. Categ.* **10**, 94–112.

[13] Johnson, M., Rosebrugh, R. and Wood, R. J. (2010) Algebras and Update Strategies. *J.UCS* **16**, 729–748.

[14] Johnson, M., Rosebrugh, R. and Wood, R. J. (2012) Lenses, Fibrations, and Universal Translations. *Math. Structures in Comp. Sci.* **22**, 25–42

[15] Orejas, F., Boronat, A., Ehrig, H., Hermann, F., and Schölzel, H. (2013) On Propagation-Based Concurrent Model Synchronization. *Electronic Communications of the EASST* **57**, 19pp.

[16] Pierce, B. (1991) *Basic Category Theory for Computer Scientists*. MIT Press.

[17] Street, R. (1974) Fibrations and Yoneda's Lemma in a 2-category. *Lecture Notes in Math.* **420**, 104–133