Algebras and Update Strategies

For Derick Wood on his 70'th birthday

Michael Johnson
(Macquarie University, Australia
mike@ics.mq.edu.oz)

Robert Rosebrugh
(Mount Allison University, Canada
rrosebrugh@mta.ca)

RJ Wood
(Dalhousie University, Canada
rjwood@mathstat.dal.ca)

Abstract: The classical (Bancilhon-Spyratos) correspondence between view update translations and views with a constant complement reappears more generally as the correspondence between update strategies and meet complements in the order based setting of S. Hegner. We show that these two theories of database view updatability are linked by the notion of "lens" which is an algebra for a monad. We generalize lenses from the category of sets to consider them in categories with finite products, in particular the category of ordered sets.
Key Words: algebra, lens, update strategy
Category: E.1; H.1, H.2

1 Introduction

This article links two theories of database view updatability. The first is that of Bancilhon and Spyratos [1], and the second is that of S. Hegner [8]. The link is the notion called "lens" studied by B. Pierce and co-authors [4].

Given a database definition (for example by a set of DDL statements in SQL), the *database states* S are the valid ways of populating the database objects (for example the tables). A *view definition* specifies a way of assigning *view states* V to database states, so it is at least a mapping from S to V. An *update* u is often considered to be an endomorphism of states. In this generality, the *view update problem* is the following:

> given a view $S \xrightarrow{g} V$ and an update $V \xrightarrow{u} V$ of the view states, when is there a compatible update (known as a *translation*) $S \xrightarrow{t_u} S$ of the database states?

For t_u to be a translation means that $gt_u = ug$, that is, the following diagram commutes (as noted even in [1]):

$$\begin{array}{ccc} S & \xrightarrow{t_u} & S \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle g} \\ V & \xrightarrow[u]{} & V \end{array}$$

Notice that we have not said what sort of structure, if any, the database states S should have. There are several structures for database states and view states that have been considered in the literature on the view update problem. Moreover, the problem has usually been addressed for a specified set of view updates.

In the early 1980's, the influential article of Bancilhon and Spyratos modeled database states as an arbitrary unstructured set S and view states as an arbitrary unstructured set V. For them a view definition is simply a (surjective) function $S \xrightarrow{g} V$. Their criterion for the existence of a translator for a set U of view updates is the existence of a so-called "complement" view $S \xrightarrow{f} C$. In short, the idea is that the view and its complement form a lossless decomposition of database states, expressed by the injectivity of $S \xrightarrow{\langle g,f \rangle} V \times C$. Then updates to a view can be made leaving the database state unchanged (constant) on the complement. Hence the name "constant complement" updating strategy.

The more recent work of S. Hegner [8] studies the view update problem when the database states and the view states are arbitrary partially ordered sets and the view definition is a(n open) surjective monotone function. Hegner also considers complements, and shows that they correspond to mappings he calls update strategies which are related to the lenses we will soon consider.

We note that we have argued [10] (and so have several others [5], [12], [18], [19]) that the database states should be structured as a *category of models* for a *sketch*. The consequences of that approach for view updates have been considered elsewhere [11]. The model categories arising are often ordered sets, although not arbitrary, so the cited approach has something in common with that of Hegner.

In the context of studying their theory of "bi-directional programming", Pierce and co-authors were led to study the notion of *lens* [4], [6], defined equationally below. They showed that lenses in the category of sets correspond to database view updatability in the sense of Bancilhon and Spyratos, and more generally to the work of Gottlob et al. [7].

At about the same time as Pierce et al. noticed the relationship of lenses and constant complement update strategies, Hegner wrote about "update strategies" for a "closed family of updates". Hegner's definition of update strategy includes being a lens in the sense appropriate to the category of partially ordered sets.

The lens equations were first considered (as far as we know) in the early 1980's by F. Oles [14], [15] in a study of abstract models of storage. Oles (as reported in [13]) also characterized models of the equations in sets as projections. In the 1990's M. Hoffman and B. Pierce [9] considered the lens equations in their study of typing for programming languages.

Our contribution is to consider the lens equations in suitable generality. As we see below, the equations make sense in a category with products. We show that viewing lenses as algebras provides the claimed link from the work of Bancilhon and Spyratos to that of Hegner.

In Section 2 we review the needed category theory. In Section 3 we consider the monad $\Delta\Sigma$ on a slice of a category with products and characterize its algebras. In Section 4 we see the data for a (totally defined) lens in sets is the same as an algebra for $\Delta\Sigma$, and that Oles' characterization has a meaning for database updating strategies. Section 5 considers the model of Hegner, and shows that his update strategies are lenses in the category of ordered sets.

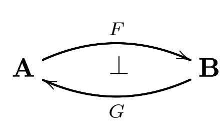
Acknowledgement. The second and third authors were undergraduate students when Derick Wood came to McMaster University in 1970. We both learned about formal language and automata theory from him. Derick was the M.Sc. supervisor of the second author, and he imparted both an abiding interest in theoretical Computer Science and the unforgettable lesson that research is fun.

2 Review of monads

We assume the reader is familiar with the most basic ideas from category theory, including functor, natural transformation and isomorphism, and (co)limits found in, for example Barr–Wells [2] or Pierce [16]. Categories will be denoted $\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$, functors $F, G, H, \ldots$ and natural transformations $\alpha, \beta, \gamma, \ldots$. We assume all of our categories are locally small, so that there is a set of arrows between any two objects, denoted $\mathbf{A}(C, C')$. We review some other definitions and results needed in the sequel.

Perhaps the most important concept from category theory is the following.

Definition 1. Let $\mathbf{A}$ and $\mathbf{B}$ be categories, and $F : \mathbf{A} \longrightarrow \mathbf{B}$ and $G : \mathbf{B} \longrightarrow \mathbf{A}$ be functors. Then F *is left adjoint to* G (also G is right adjoint to F, the pair F, G are adjoint) if for any objects A in $\mathbf{A}$ and B in $\mathbf{B}$ there is a bijection $\mathbf{B}(FA, B) \cong \mathbf{A}(A, GB)$ which is natural in A and B. We will depict this by:



and write $F \dashv G$ and call the relationship an *adjunction*.

Following identity arrows through the bijections mentioned determines natural transformations $\eta : 1_{\mathbf{A}} \longrightarrow GF$ and $\epsilon : FG \longrightarrow 1_{\mathbf{B}}$ called the *unit* and *counit*. They satisfy the so-called *triangular identities* and, furthermore, a pair of natural transformations satisfying these identities determines an adjunction. For details see [2].

Example 1. A standard example of adjunction is provided by diagonal functors and (co)products. Let $\mathbf{A}$ be a category. For the case of binary products, denote the functor whose value on A in $\mathbf{A}$ is the pair of objects (A, A) by $\Delta : \mathbf{A} \longrightarrow \mathbf{A} \times \mathbf{A}$. A right adjoint to Δ is a product functor, denoted $- \times - : \mathbf{A} \times \mathbf{A} \longrightarrow \mathbf{A}$. Indeed, $D \times E$ is the product of objects D and E of $\mathbf{A}$ exactly when, for any object A, there a bijection from pairs of arrows $A \longrightarrow D, A \longrightarrow E$ to arrows $A \longrightarrow D \times E$. Notice that the identity arrow on $D \times E$ then corresponds under the counit to a pair of arrows denoted generically as $\pi_0 : D \times E \longrightarrow D, \pi_1 : D \times E \longrightarrow E$ called the *projections*. The pair of identity arrows from A to A corresponds under the unit to an arrow denoted $A \xrightarrow{\delta_A} A \times A$. A coproduct functor is a left adjoint to Δ. ⋄

As we will review below, adjunctions generate examples of the next concept, and vice versa.

Definition 2. Let $\mathbf{A}$ be a category. A *monad* T on $\mathbf{A}$ is a triple $\mathsf{T} = (T, \eta, \mu)$ where $T : \mathbf{A} \longrightarrow \mathbf{A}$ is a functor, $\eta : 1_{\mathbf{A}} \longrightarrow T$ and $\mu : T^2 \longrightarrow T$ are natural transformations and they satisfy the unitary and associative equations:

$$\mu(\eta T) = 1_T = \mu(T\eta) \qquad \mu(\mu T) = \mu(T\mu)$$

Example 2. A familiar monad on the category **set** of sets is the *free monoid monad*. For a set X, let $TX = X^*$, the free monoid on X. T extends easily to a functor on **set**. The inclusion of generators (single letters) provides a function $\eta_X : X \longrightarrow TX$. Since $T^2X = (X^*)^*$ is "words of words", we can define $\mu_X : T^2X \longrightarrow TX$ to be the function which simply "multiplies out" and provides a word. The equations are easily seen to be satisfied. ⋄

Example 3. Let $F : \mathbf{A} \longrightarrow \mathbf{B}$ and $G : \mathbf{B} \longrightarrow \mathbf{A}$ be functors and $F \dashv G$. The functor GF underlies a monad $\mathsf{T}_{GF} = (GF, \eta_{GF}, \mu_{GF})$ where η_{GF} is the unit for the adjunction and $\mu_{GF} = G\epsilon F : GFGF \longrightarrow GF$. ⋄

Definition 3. Let $\mathsf{T} = (T, \eta, \mu)$ be a monad on a category $\mathbf{A}$. An *algebra for* T is a pair $X = (X, \xi)$ where $\xi : TX \longrightarrow X$ is an arrow of $\mathbf{A}$ satisfying the equations:

$$\text{unit law:} \quad \xi(\eta_X) = 1_{TX} \qquad \text{associative law:} \quad \xi(\mu_X) = \xi(T\xi)$$

A *morphism of algebras* from $X = (X, \xi)$ to $Y = (Y, \zeta)$ is an arrow $f : X \longrightarrow Y$ in $\mathbf{A}$ satisfying $f\xi = \zeta Tf$.

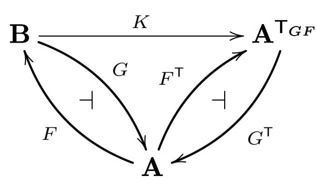
The algebras for a monad form a category denoted $\mathbf{A}^{\mathsf{T}}$ with composition inherited from $\mathbf{A}$. As is common practice, when the unit and multiplication for a monad are clear, as in Example 3 for example, we often just name a monad T by its functor part T, and refer to T algebras and so on.

Example 4. An algebra for the free monoid monad is a monoid in the category **set**. The category of algebras is the category of monoids and their homomorphisms. $\diamond$

Adjoints, monads and their algebras are related by the following well-known results:

Theorem 4. *Let T be a monad on $\mathbf{A}$. There are functors $F^{\mathsf{T}} : \mathbf{A} \longrightarrow \mathbf{A}^{\mathsf{T}}$ and $G^{\mathsf{T}} : \mathbf{A}^{\mathsf{T}} \longrightarrow \mathbf{A}$ defined by $G^{\mathsf{T}}(X, \xi) = X$, $F^{\mathsf{T}}X = (TX, \mu_X)$ and satisfying $F^{\mathsf{T}} \dashv G^{\mathsf{T}}$. Let $F : \mathbf{A} \longrightarrow \mathbf{B}$ and $G : \mathbf{B} \longrightarrow \mathbf{A}$ be functors with $F \dashv G$ and suppose $\mathsf{T} = \mathsf{T}_{GF}$. Then there is a* comparison functor *$K : \mathbf{B} \longrightarrow \mathbf{A}^{\mathsf{T}}$ defined by $KB = (GB, G\epsilon_{GB})$. Moreover, $KF = F^{\mathsf{T}}$ and $G^{\mathsf{T}}K = G$.*

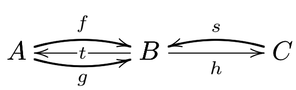
The next diagram sums up the situation



There are criteria which ensure that the comparison functor K is an isomorphism or equivalence of categories, namely the celebrated monadicity theorems of J. M. Beck. We review one of those below.

Definition 5. Let G be a functor $\mathbf{B} \xrightarrow{G} \mathbf{A}$ with a left adjoint F. Then G is called monadic if the comparison functor $\mathbf{B} \xrightarrow{K} \mathbf{A}^{\mathsf{T}_{GF}}$ is an equivalence of categories.

Before stating Beck's theorem we review some standard terminology. A functor G *reflects isomorphisms* if f is an isomorphism whenever Gf is so. A *contractible coequalizer* is a diagram of arrows:



satisfying $ft = 1_B, gt = sh, hs = 1_C$ and $hf = hg$. A pair of arrows $A \underset{g}{\overset{f}{\rightrightarrows}} B$ is a G-contractible coequalizer pair if it becomes part of a contractible coequalizer after application of G.

Theorem 6. *Let* $\mathbf{B} \xrightarrow{G} \mathbf{A}$ *be a functor. Then* G *is monadic (in the stronger sense that* K *is an isomorphism) if and only if* G *has a left adjoint;* G *reflects isomorphisms; and* $\mathbf{B}$ *has coequalizers of* G*-contractible coequalizer pairs which* G *preserves.*

When we use this theorem it will be the case that $\mathbf{B}$ has all coequalizers and the functor G preserves them. For more details we refer the reader to [3].

3 $\mathsf{T}_{\Delta\Sigma}$ algebras

In this section we consider algebras for a monad that is the basis of our description of lenses in the sequel. The monad uses a well-known construction: Let $\mathbf{C}$ be a category. For any object V of $\mathbf{C}$ the *slice category* $\mathbf{C}/V$ is constructed as follows. An object is an arrow $g : C \longrightarrow V$ to V. An arrow from g to $g' : C' \longrightarrow V$ is an arrow $f : C \longrightarrow C'$ satisfying $g'f = g$, so arrows are the same thing as commutative triangles ending at V. There is always a functor $\Sigma_V : \mathbf{C}/V \longrightarrow \mathbf{C}$ defined on objects by $\Sigma_V g = C$ and on arrows by $\Sigma_V f = f$.

Example 5. Now let $\mathbf{C}$ be a category with finite products. There is a functor $\Delta_V : \mathbf{C} \longrightarrow \mathbf{C}/V$ (*not* the Δ of Example 1) that is defined on objects by $\Delta_V C = V \times C \xrightarrow{\pi_0} V$ and on arrows by $\Delta_V f = 1_V \times f$. We will usually drop the V subscripts. Note that for an object $C \xrightarrow{g} V$, $\Delta\Sigma g = V \times C \xrightarrow{\pi_0} V$. There is an adjunction:

$$\mathbf{C}/V \underset{\Delta}{\overset{\Sigma}{\rightleftarrows}} \mathbf{C} \qquad (\Sigma \dashv \Delta)$$

The g'th component of the unit η for the adjunction is $g \xrightarrow{\eta_g} \Delta\Sigma g$ as in the commutative triangle:

$$\begin{array}{ccc} C & \xrightarrow{\langle g,1\rangle} & V \times C \\ & {\scriptstyle g}\searrow & \downarrow{\scriptstyle \pi_0} \\ & & V \end{array}$$

The adjunction determines the monad $\mathsf{T}_{\Delta\Sigma}$ on $\mathbf{C}/V$. The unit for the monad is η while the g'th component of its multiplication is

$$\Delta\Sigma\Delta\Sigma g \xrightarrow{\mu_g} \Delta\Sigma g$$

as in:

$$\begin{array}{ccc} V \times C \times C & \xrightarrow{\langle \pi_0,\pi_2\rangle} & V \times C \\ {\scriptstyle \pi_0}\searrow & & \swarrow{\scriptstyle \pi_0} \\ & V & \end{array}$$

◇

The following characterization of $\Delta\Sigma$ algebras is useful for the sequel. We will abbreviate $\langle\pi_0, \pi_2\rangle$ to $\pi_{0,2}$.

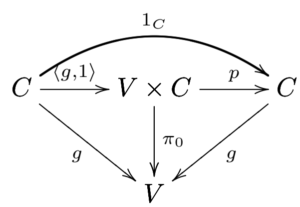
Proposition 7. *Let* $\mathbf{C}$ *be a category with finite products. An algebra structure on* $g : C \longrightarrow V$ *in* $\mathbf{C}/V$ *for the monad* $\Delta\Sigma$ *on* $\mathbf{C}/V$ *is an arrow* $p : V \times C \longrightarrow C$ *satisfying:*

i) $gp = \pi_0$

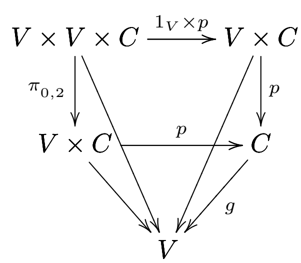
ii) $p\langle g, 1_C\rangle = 1_C$

iii) $p(1_V \times p) = p\pi_{0,2}$

Proof. As the commutativity of the diagram below illustrates, the equation i) shows that, viewed as a morphism from $\Delta\Sigma g$ to g, p is indeed a morphism of $\mathbf{C}/V$, while the equation ii) shows that p satisfies the unit law for the monad $\Delta\Sigma$.



In the next diagram, the unlabelled vertical arrows are projections, so the whole diagram makes a commutative square in $\mathbf{C}/V$. Since $\Delta\Sigma p = 1_V \times p$ and $\mu_g = \pi_{0,2}$ the equation iii) (the top square) shows that p is associative.



□

In the sequel it will be important to know when Δ is monadic.

4 Lenses and update translations

In this section we consider the notion of lens (in **set**) defined by Pierce and co-authors[4], [6], [17]. The context is their study of “bi-directional programming”.

As they point out, the lens equations also appear in the programming language literature, both in Oles category of "state shapes" [14] and in Hofmann and Pierce's work on "positive subtyping"[9].

As we will see shortly, the data for and the equations satisfied by a very well behaved lens in [4] determine an algebra for the monad $\Delta_V \Sigma_V$ on the category **set**. It is our basic observation that Δ_V is usually monadic. Thus, a $\Delta_V \Sigma_V$ algebra for $\mathbf{set}/V$, equivalently a very well behaved lens, is specified by an object of the domain of the monadic Δ_V, that is, a set. The set in question determines a view complement as studied by Bancilhon and Spyratos.

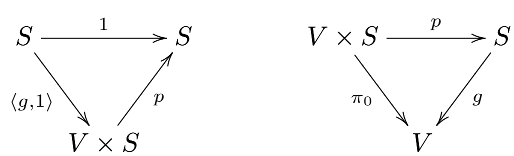
Briefly, a lens in **set** involves two mappings, "Get" and "Put", and equations. In the interpretation for databases, the Get mapping determines a view state from a database state. The Put (or "Putback") mapping determines a new database state s' from pair (v, s) of a database state and a view state. The idea is the following: If some update u of the Get of the database state s results in the view state v, then the Put of the pair (v, s) is the new database state s'. The Get of this new database state must be the view state v (equation PutGet below). Moreover, if the update u is trivial, the Put of (v, s) is just the projection on s (equation GetPut below).

Definition 8. A *lens in* **set** *[4]* is $L = (S, V, g, p)$ where S and V are sets (the states and the view states); g is a mapping $S \xrightarrow{g} V$ (the "Get" mapping); p is a mapping $V \times S \xrightarrow{p} S$ (the "Put" mapping). A lens is called *well behaved* if it satisfies:

(i) (PutGet) the Get of a Put is the projection: $g(p(v, s)) = v$

(ii) (GetPut) the Put for a trivially updated state is trivial: $p(g(s), s) = s$

Diagrammatically:



GetPut PutGet

A well behaved lens is called *very well behaved* if it satisfies:

(iii) (PutPut) composing Puts depends only on the second view state:
$p(v', p(v, s)) = p(v', s)$

Diagrammatically:

$$\begin{array}{ccc} V \times V \times S & \xrightarrow{1_V \times p} & V \times S \\ {\scriptstyle \pi_{0,2}}\downarrow & & \downarrow{\scriptstyle p} \\ V \times S & \xrightarrow[p]{} & V \end{array}$$

PutPut

Example 6. We illustrate that a lens as just defined may fail to be well behaved. The example is from [4].

Suppose that a relational database schema has two signatures, $R(A, B)$ and $S(B.C)$ (we ignore type information for A, B, C). The view database schema has just one signature $T(A, B, C)$. The set S of database states is the set of pairs **R**,**S** of tables with column headings from the signatures, and similarly the set V of view states is the set of tables **T**. The action of the Get mapping on a database state **R**,**S** is to determine the view state (table) **T** which is the natural join of **R**,**S**. For example, with **R** and **S** as follows, we get **T** as shown:

$$\left\{ \begin{array}{c|cc} \mathbf{R} & A & B \\ \hline & a_1 & b_1 \\ & a_1 & b_2 \end{array} \quad \begin{array}{c|cc} \mathbf{S} & B & C \\ \hline & b_1 & c_1 \\ & b_1 & c_2 \end{array} \right\} \longmapsto \left\{ \begin{array}{c|ccc} \mathbf{T} & A & B & C \\ \hline & a_1 & b_1 & c_1 \\ & a_1 & b_1 & c_2 \end{array} \right\}$$

The Put mapping on a pair consisting of a view state **T** and a database state **R,S** provides a new state $\mathbf{R}'$,$\mathbf{S}'$ by simply projecting **T** onto A, B and B, C respectively. For the example above:

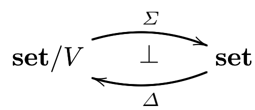
$$\left\{ \begin{array}{c|ccc} \mathbf{T} & A & B & C \\ \hline & a_1 & b_1 & c_1 \\ & a_1 & b_1 & c_2 \end{array} \right\} \longmapsto \left\{ \begin{array}{c|cc} \mathbf{R} & A & B \\ \hline & a_1 & b_1 \\ & & \end{array} \quad \begin{array}{c|cc} \mathbf{S} & B & C \\ \hline & b_1 & c_1 \\ & b_1 & c_2 \end{array} \right\}$$

Thus for this lens the GetPut equation is not satisfied. Of course, the reason is that the Put function we defined ignores the original database state. Failure of GetPut can be repaired simply by changing Put to take account of the original state. As we shall see, for any well behaved lens the Get and Put functions are surjective. The lens above thus necessarily fails to satisfy PutGet because the Get is not surjective. Indeed, for the Get defined, there is no Put defining a lens satisfying PutGet.

We can modify the example to a very well behaved lenses. First modify the view schema so that it has two signatures $T(A), V(C)$. Then the new Get mapping on a database state **R**,**S** determines the view state (tables) **T**,**V** by selecting components from rows of **R**,**S** with B component b_1. The Put on $\{\mathbf{T},\mathbf{V}\}$, $\{\mathbf{R},\mathbf{S}\}$ simply places rows with B component b_1 into **R**,**S** for each element of **T**,**V**. $\diamond$

Notice that there is a unique lens with in **set** with S empty. Since the identity is surjective, PutGet for a lens in **set** implies that if S is non-empty then g is surjective.

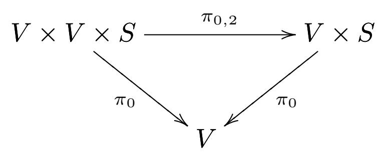
Recall the adjunction from Example 5 for the case $\mathbf{C} = \mathbf{set}$:



In this case, for $S \xrightarrow{g} V$ in $\mathbf{set}/V$ and X in **set**, we have $\Sigma(g) = S$ and $\Delta(X) = V \times X \xrightarrow{\pi_0} V$, so $\Delta\Sigma g = V \times S \xrightarrow{\pi_0} V$. The g'th component of the unit for the adjunction is $g \xrightarrow{\eta_g} \Delta\Sigma g$. The g'th component of the multiplication for the monad $\Delta\Sigma$ is

$$\Delta\Sigma\Delta\Sigma g \xrightarrow{\mu_g} \Delta\Sigma g$$

which as a commutative triangle in **set** is:



Let L be a lens. The PutGet law and the GetPut law say that i) and ii) of Proposition 7 are satisfied, and the PutPut law says that iii) of that Proposition is satisfied. Thus:

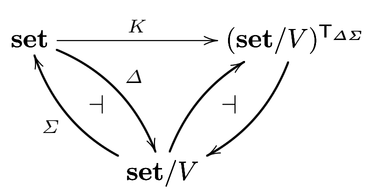
Proposition 9. *A very well behaved lens* $L = (S, V, g, p)$ *is exactly the data for an algebra* (g, p) *for the monad* $\mathsf{T}_{\Delta\Sigma}$ *on* $\mathbf{set}/V$.

Our primary interest is very well behaved lenses. We now assume that *unless otherwise noted all lenses are very well behaved.* With that assumption, and for later use, we make the following general definition.

Definition 10. Let $\mathbf{C}$ be a category with finite products and V an object of $\mathbf{C}$. A *lens in* $\mathbf{C}$ *with view states* V is an algebra for the monad $\Delta\Sigma$.

Equivalently, a lens in $\mathbf{C}$ with view states V is a pair of arrows $C \xrightarrow{g} V$, $V \times C \xrightarrow{p} C$ satisfying the equations in Proposition 7.

Next we consider monadicity of Δ. Consider the following diagram in which K is the comparison functor from **set** to $\Delta\Sigma$ algebras.



There is a trivial case: if $V = \emptyset$, then $\mathbf{set}/V \cong \mathbf{1}$, the terminal category, and then the category of $\Delta\Sigma$ algebras is also isomorphic to $\mathbf{1}$. Otherwise, as we show directly, K is an equivalence of categories. That is, we are going to show directly that Δ is monadic. Notice that K is defined on objects as follows:

$$K(C) = (\pi_0 : V \times C \longrightarrow V, \pi_{0,2} : V \times V \times C \longrightarrow V \times C)$$

We will need:

Lemma 11. *Let* $(S \xrightarrow{g} V, V \times S \xrightarrow{p} S)$ *be a* $\Delta\Sigma$ *algebra in* $\mathbf{set}$. *For all* v, v' *in* V, $g^{-1}(v) \cong g^{-1}(v')$

Proof. The statement is evidently true when S is empty. Otherwise g is surjective, so all $g^{-1}(v)$ are non-empty.

For v, v' in V, define $\varphi_{v,v'} : g^{-1}(v) \longrightarrow g^{-1}(v')$ by $\varphi_{v,v'}(s) = p(v', s)$, and note $\varphi_{v,v'}(s)$ is in $g^{-1}(v')$ since $g(p(v', s)) = v'$. Next,

$$\begin{aligned}
\varphi_{v',v}(\varphi_{v,v'}(s)) &= \varphi_{v',v}(p(v', s)) \\
&= p(v, p(v', s)) \\
&= p(v, s) \\
&= p(g(s), s) \\
&= s
\end{aligned}$$

where the last three equations follow from, respectively, PutPut, that s is in $g^{-1}(v)$, and GetPut. Interchanging the roles of v and v' in the equation just demonstrated shows that $\varphi_{v',v}$ is inverse to $\varphi_{v,v'}$ which completes the proof. □

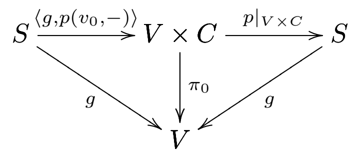
Theorem 12. *If* V *is non-empty,* K *is an equivalence and so* Δ *is monadic.*

Proof. Choose a v_0 in V. We define a functor $H : (\mathbf{set}/V)^{\mathsf{T}_{\Delta\Sigma}} \longrightarrow \mathbf{set}$ making K an equivalence. Let $(S \xrightarrow{g} V, p)$ be a $\Delta\Sigma$ algebra. Define $C = g^{-1}(v_0)$ and $H(g, p) = C$. Note that by Lemma 11, C is (up to isomorphism) independent of the choice of v_0. To define H on arrows recall that an arrow f in $(\mathbf{set}/V)^{\Delta\Sigma}$ from (g, p) to $(S' \xrightarrow{g'} V, p')$ is a mapping $S \xrightarrow{f} S'$ satisfying $g'f = g$ and $p'(V \times f) = fp$. Thus f restricts to $C = g^{-1}(v_0) \longrightarrow g'^{-1}(v_0) = C'$ and H is clearly functorial.

Next we show that KH is isomorphic to the identity on $(\mathbf{set}/V)^{\mathsf{T}_{\Delta\Sigma}}$. To do this we show that (g, p) is isomorphic to $KH(g, p) = (V \times C \xrightarrow[0]{\pi} V, \pi_{0,2})$. By the definition of C, $\langle g, p(v_0, -)\rangle$ maps S to $V \times C$ and $g = \pi_0\langle g, p(v_0, -)\rangle$ giving an arrow from g to π_0 in $\mathbf{set}/V$. It is an algebra homomorphism because:

$$\langle g(p(v, s)), p(v_0, p(v, s))\rangle = \langle v, p(v_0, s)\rangle = \pi_{0,2}\langle v, g(s), p(v_0, s)\rangle$$

The restriction of p to $V \times C$ provides an arrow in $\mathbf{set}/V$ from π_0 to g which is an algebra homomorphism by PutPut. To show these arrows are mutually inverse consider:



and note that $p(g(s), p(v_0, s)) = p(g(s), s) = s$ so the top composes to the identity on S. On the other hand, $\langle g(p(v, c)), p(v_0, p(v, c))\rangle = \langle v, p(v_0, c)\rangle = \langle v, p(g(c), c)\rangle = \langle v, c\rangle$ showing that the other composite is the identity.

Finally we need that $HK \cong 1_{\mathbf{set}}$, but this is easy to see. Indeed, since $K(C)$ is a structure on $\pi_0 : V \times C \longrightarrow V$, $HK(C) = \pi_0^{-1}(v) \cong C$ for any v. □

We note an important point from the proof:

Corollary 13. *Let $L = (S, V, g, p)$ be a (very well behaved) lens with V non-empty, v_0 in V, and C the set $g^{-1}(v_0)$. Denote the projection $V \times C \xrightarrow{\pi_0} V$. The arrow $\langle g, p(v_0, -)\rangle : S \longrightarrow V \times C$ is inverse to $p|_{V\times C}$ in* **set** *and defines an isomorphism $(g, p) \cong (\pi_0, \pi_{0,2})$ in $\Delta\Sigma$ algebras.*

Remark. The C in the corollary is the set Bancilhon and Spyratos call the "complement" of V. Here the complement view is simply the projection $V \times C \xrightarrow{\pi_1} C$, and of course we have a "constant complement" decomposition.

The theorem also follows easily by Beck's monadicity theorem, Theorem 6 above. We know Δ has a left adjoint. It is a "logical" functor, so it preserves all coequalizers (and this is also easy to see directly). There remains to show only that Δ reflects isomorphisms. However, $h : \Delta(X) \longrightarrow \Delta(Y)$ is iso exactly if the function $h : V \times X \longrightarrow V \times Y$ is. Since V is non-empty and h is an arrow of $\mathbf{set}/V$ (by the projections), for v_0 in V the restriction of h to $\{v_0\} \times X \longrightarrow \{v_0\} \times Y$ is a bijection. We conclude that $X \xrightarrow{\cong} Y$.

In some writings Pierce et al. allow the Get and Put to be partial functions and call the lenses of Definition 8 a "total lens" (for example [6]), but they remark that "In practice, we always want lenses to be total...". For most of our purposes, lenses with total Get and Put suffice, but we introduce the following terminology for use below.

Definition 14. A *partial lens in* **set** $L = (S, V, g, p)$ where S, V, g, p are as above, except that g and p may be partial mappings. A partial lens in **set** is *total for $P \subseteq \mathbf{set}(S, S)$ and $U \subseteq \mathbf{set}(V, V)$* if

(i) g is a total function

(ii) the domain of p is $\{(u(gs), s) | u \in U, s \in S\}$

(iii) $p(v, s) = s'$ implies $s' = r(s)$ for some update $r \in P$

In the database context, the set U is intended to be the set of (view) updates for which a translation is required, and P is a set of (database) updates which includes the translations of updates in U. By conditions ii) and iii), a partial lens which is total for any U such that $V \times S \subseteq \{(u(gs), s) | u \in U, s \in S\}$ and P such that the image of p is contained in the images of updates in P is the same thing as a lens in **set**. Now P is merely the set of potential translations, so as long as it contains the identity (so that iii) is satisfied), it does no harm to take $P = \mathbf{set}(S, S)$.

For database views, it certainly makes sense to require the Get g for a lens to be totally defined, but a Put might be partial.

Let $\mathbf{C}$ be a category with (chosen) finite limits. We also require that $\mathbf{C}$ have an epi-mono factorization system for its arrows, and that pullbacks of monic arrows are monic. Denote by $par(\mathbf{C})$ the partial map category. Its objects are those of $\mathbf{C}$ and an arrow $C \xrightarrow{f} C'$ is a span from C to C' denoted $C \xleftarrow{f_0} D_f \xrightarrow{f_1} C'$ with f_0 monic. Composition is by (chosen) pullback. With this hypothesis, it is easy to extend the Δ_V and Σ_V from above to the partial map category and to check that there is still an adjunction between them. In the **set** case:

$$par(\mathbf{set}/V) \underset{\Delta}{\overset{\Sigma}{\rightleftarrows}} par(\mathbf{set}) \qquad (\Sigma \dashv \Delta)$$

For the resulting monad, the comparison functor K has domain $par(\mathbf{set})$ and codomain the $\Delta\Sigma$ algebras $(par(\mathbf{set}))^{\mathsf{T}_{\Delta\Sigma}}$. However K is no longer an equivalence, nor is it even fully faithful. It is still the case, of course, that a $\Delta\Sigma$ algebra is a (partial) lens.

We end this section by recalling the relationship between lenses and the translators of Bancilhon and Spyratos [1]. As mentioned above, for Bancilhon and Spyratos a *view* $g : S \longrightarrow V$ is a surjective function. A *complete set of updates* is a set $U \subseteq \mathbf{set}(V, V)$ closed under composition and such that for u in U and s in S there is a v in U such that $vu(s) = s$. A *translator* T for U is a composition-preserving function $T : U \longrightarrow \mathbf{set}(S, S)$ such that for u in U, $gT(u) = ug$. The relationship noted by Pierce and Schmitt is:

Theorem 15. *There is a one-one correspondence between, one the one hand, triples (g, p, U) with $L = (S, V, g, p)$ a very well behaved lens that is total (Definition 14) for a complete set of updates $U \subseteq \mathbf{set}(V, V)$ (and $P \subseteq \mathbf{set}(S, S)$) and, on the other hand, triples (g, U, T) where T is a translator for the complete set of updates U of a view $S \xrightarrow{g} V$.*

This theorem appears in a manuscript [17] (referred to in [4]). By the theorem, lenses, or $\Delta\Sigma$ algebras, correspond to translators. Bancilhon and Spyratos showed directly that translators are essentially the same as product decompositions. The main point of this section is that Theorem 12 and Corollary 13 show that translators correspond to decompositions indirectly using Theorem 15. Moreover, our results determine the second factor in a decomposition of the domain of the view as the set determined by an algebra/lens (under H in the proof above).

For completeness, we note that [17] also shows that a merely well behaved lens corresponds to the notion of *dynamic view* in the sense of [7]. Furthermore it is shown by direct construction that the domain of a very well behaved lens decomposes as a product with V (not as a consequence of Theorem 12). Indeed, the product decomposition for lenses was also noted by Oles [14].

5 The Ordered Case: Update Strategies

Denote by **ord** the category of partially ordered sets and monotone mappings. We recall that **ord** is a category with finite limits. The finite limits are computed as in **set**. Indeed, the order on a product of ordered sets is essentially a product of their orders, and a terminal ordered set is a singleton set with its unique order. The equalizer of a pair of monotone mappings is their equalizer in **set** with the inherited order. It follows that monomorphisms in **ord** are regular (they are the equalizers), and the pullback of a mono is a mono.

Surjective mappings in **ord** are, of course, epimorphisms. However, like the category of categories, **ord** is not a regular category.

Since **ord** has products, for any ordered set V, we have an adjunction that we again denote $\Sigma \dashv \Delta$. A $\Delta\Sigma$ algebra is a lens in **ord** with view states V .

Below we will need to use a factorization system for arrows in **ord** which we now describe. Let $X \xrightarrow{f} Y$ be a monotone mapping of ordered sets. Denote by $\equiv_f$ the (kernel) equivalence relation on the set X defined by $x \equiv_f x'$ iff $fx = fx'$. As usual this means the function f factorizes as $X \xrightarrow{p_f} X/\equiv_f \xrightarrow{i_f} Y$ through the quotient set. We define a partial order $\leq_f$ on $X/\equiv_f$ as the transitive closure of the relation $\sqsubset$ on $X/\equiv_f$ defined by $[x_1] \sqsubset [x_2]$ iff $\exists x_1', x_2'$ such that $x_1 \equiv_f x_1'$, $x_1' \leq x_2'$ and $x_2' \equiv_f x_2$. The relation $\leq_f$ is reflexive and transitive by definition. That it is antisymmetric follows antisymmetry of the order on Y. The function p_f is clearly monotone by its definition. Transitivity of the order on Y makes i_f monotone.

In [8], Hegner defines a database schema to be a partially ordered set S. His intention is that S is the totality of database states and that database states may be comparable. For him a *view* is an *open surjection* $S \xrightarrow{g} V$ of ordered sets.

This means that g is required to be an onto monotone function, and whenever $v_1 \leq v_2$ in V there exist s_1, s_2 in S with $s_1 \leq s_2$ and $g(s_i) = v_i$. Open surjections are so named because they define open mappings for the order topologies on S and V.

Definition 16. [8] A *closed update family* T on a database schema S is an order compatible equivalence relation, i.e. $s_1 \leq s_2 \leq s_3$ and $s_1 \sim_T s_3$ implies $s_1 \sim_T s_2$.

The idea is that $s_1 \sim_T s_2$ means that s_1 is updatable to s_2. Transitivity and symmetry mean updates are composable and reversible (like a complete set of updates for Bancilhon and Spyratos). Note that $\leq_f$ as defined above is order-compatible.

Definition 17. [8] Let $S \xrightarrow{g} V$ be a view and T a closed update family on V. An *update strategy* for T is a partial function $V \times S \xrightarrow{p} S$ such that (equations valid when defined):

up1: $p(v, s)$ is defined iff $(v, g(s))$ in T

up2: $g(p(v, s)) = v$

up3: $p(g(s), s) = s$

up4: $p(g(s), p(v, s)) = s$

up5: $p(v', p(v, s)) = p(v', s)$

up6: $g(s) \leq v$ implies $s \leq p(v, s)$

up7: $s_1 \leq s_2 \leq p(v_1, s_1)$ implies $\exists v_2,\ p(v_2, s_1) = s_2$ & $p(g((p(v_1, s_1)), s_2) = p(v_1, s_1)$

up8: $s_1 \leq s_2$ & $v_1 \leq v_2$ implies $p(v_1, s_1) \leq p(v_2, s_2)$

The property *up8* states that an update strategy p is a monotone partial mapping. If V is empty there is, of course, exactly one view to V from the empty order, and otherwise as we show next an update strategy is exactly a lens in **ord**.

Theorem 18. *Let V be non-empty in* **ord**. *For a view $S \xrightarrow{g} V$, an update strategy p for the "all" closed update family, $V \times V$, is an algebra for the monad $\Delta\Sigma$ on* **ord**/V. *Conversely, a view $S \xrightarrow{g} V$ with a $\Delta\Sigma$ algebra structure in* **ord**, *$V \times S \xrightarrow{p} S$, determines an update strategy for the closed update family $V \times V$.*

Proof. For the first part, since g is surjective and T is symmetric, *up1* implies that p is total. By *up8*, p is monotone (as is g, being a view). Now *up2*, *up3* and *up5* state that an update strategy p satisfies the PutGet, GetPut and PutPut laws for a lens in **ord**, so by Proposition 7, p is a $\Delta\Sigma$ algebra structure on g.

For the converse, suppose that a view g is a $\Delta\Sigma$ algebra with structure p. Thus g is total and surjective since $gp = \pi_0$ is surjective. Moreover p is monotone so *up8* is satisfied. Because p is total, *up1* is trivially satisfied for $T = V \times V$ ($p(v, g(s))$ is defined iff $(v, g(s))$ is in T). By Proposition 7 again, the algebra equations imply *up2*, *up3* and *up5*. Thus, *up1*, *up2*, *up3*, *up5* and *up8* are satisfied.

Since $p(g(s), p(v, s))$ is always defined, *up5* (PutPut) implies *up4* as seen by $p(g(s), p(v, s)) = p(g(s), s) = s$. Furthermore, *up8* implies *up6* since $g(s) \leq v$ implies $(g(s), s) \leq (v, s)$ implies $s = p(g(s), s) \leq p(v, s)$.

That leaves *up7*. Using *up3*, we can restate *up7* as:
$s_1 \leq s_2 \leq p(v_1, s_1)$ implies $p(g(s_2), s_1) = s_2$ & $p(v_1, s_2) = p(v_1, s_1)$
(take $v_2 = g(s_2)$ and note $g(p(v_1, s_1)) = p(v_1, s_1)$)

Suppose that $s_1 \leq s_2 \leq p(v_1, s_1)$. Now $s_2 = p(v_2, s_2) \leq p(v_2, p(v_1, s_1)) = p(v_2, s_1)$ using $v_2 = g(s_2)$, the hypothesis and *up5*. On the other hand $p(v_2, s_1) \leq p(v_2, s_2) = s_2$ using that p is monotone. The inequalities give $p(g(s_2), s_1) = s_2$. Using the equality just proved and *up5* again gives $p(v_1, s_2) = p(v_1, p(v_2, s_1)) = p(v_1, s_1)$. □

For consistency with the definitions from [8], we are going to modify slightly the monad $\Delta\Sigma$ on **ord**. We denote the category of non-empty partially ordered sets by $\mathbf{ord}^+$. For a non-empty partially ordered set V, we denote the full subcategory of $\mathbf{ord}/V$ whose objects are open surjections by $\mathbf{ord}^+/_o V$.

Lemma 19. *The functors Σ and Δ restrict to $\mathbf{ord}^+/_o V$ and $\mathbf{ord}^+$, respectively, and for the restrictions we still have $\Sigma \dashv \Delta$.*

Proof. The only point we note is that a projection to a non-empty ordered set is clearly open. □

Once again, we denote the comparison functor from $\mathbf{ord}^+$ to $\Delta\Sigma$ algebras by K.

Theorem 20. *Let V be non-empty in* **ord**. *The comparison functor K is an equivalence and so Δ is monadic.*

Proof. The proof is the same as that for Theorem 12 once we note that the $\varphi_{v,v'}$ used there are monotone since they are defined using the monotone p. □

The analogue of Corollary 13 is:

Corollary 21. *Let V be non-empty in* **ord** *with v_0 in V. Let $S \xrightarrow{g} V$ be a view, p an update strategy for the closed update family $V \times V$, and C the ordered set $g^{-1}v_0$. The arrow $\langle g, p(v_0, -)\rangle : S \longrightarrow V \times C$ is inverse to $p|_{V \times C}$ in* **ord** *and defines an isomorphism $(g, p) \cong (\pi_0, \pi_{0,2})$ in $\Delta\Sigma$ algebras.*

Remark. The C in the corollary is the ordered set Hegner [8], Corollary 3.10, calls the meet complement of V.

Like the main result in the previous section, Theorem 20 also follows by Beck's monadicity theorem, Theorem 6. However in the **ord** case we need to consider contractible coequalizers, and the resulting argument is no simpler than the direct proof above.

Steve Lack reminded us that for **C** with finite limits and coequalizers, the functor Δ_V is monadic exactly when the unique arrow $V \xrightarrow{t_V} 1$ is an effective descent morphism. For this to be so, it is sufficient that V have an element (a right inverse to t_V), which explains the sufficiency of requiring the element v_0 in our results above.

Because **ord** has pullbacks, and the pullback of an injective monotone mapping is injective, we can define the category of partial monotone mappings *par*(**ord**). Since **ord** has finite limits, it is also the case that for an ordered set V, **ord**/V has finite limits. As in the previous Section, we have an adjunction:

$$\mathit{par}(\mathbf{ord}/V) \underset{\Delta}{\overset{\Sigma}{\rightleftarrows}} \mathit{par}(\mathbf{ord}) \qquad \Sigma \dashv \Delta$$

Once again the comparison functor K from *par*(**ord**) to $\Delta\Sigma$ algebras is not an equivalence, but a $\Delta\Sigma$ algebra is certainly a partial lens in **ord**. The partial lenses in the image of the comparison functor have a decomposition like that in Corollary 21.

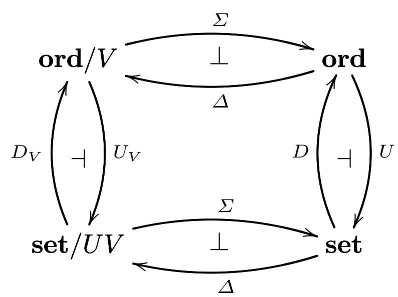
We conclude this section by pointing out that Corollary 21 again provides an indirect proof of the update strategy/meet complement correspondence.

6 Conclusion

The main results of this article show that the constant complement view updating strategies arise because they correspond to the concept of lens or $\Delta\Sigma$ algebra in appropriate categories. As pointed out in [8], the value of such strategies lies in their assurance that anomalous view updates are forbidden.

If we ignore the restriction to non-empty orders and open surjections in the previous section, we see there is more than analogy linking $\Delta\Sigma$ algebras in **set** and in **ord**. There is a forgetful functor $U : \mathbf{ord} \longrightarrow \mathbf{set}$ which has a left adjoint

D, whose value at a set X is the discrete ordered set on X. Now U can be extended to a functor $U_V : \mathbf{ord}/V \longrightarrow \mathbf{set}/UV$ which on objects simply applies U to a monotone mapping $X \longrightarrow V$. This functor has a left adjoint that we denote D_V. Its value at a function $Y \longrightarrow UV$ is the adjunct monotone mapping $DY \longrightarrow V$. The following diagram sums up the situation and we note that both squares commute. The functors Δ are monadic, and this is a sort of "adjoint change of base" for algebras. Moreover, the functors D_V and D express the **set** lenses as a special case of **ord** lenses.



We are currently considering lenses in the context of the categorical sketch data model [10]. In that model we showed how updatability is expressible via cartesian structure on the database states (category) [11]. In a forthcoming article we will address connections between the lens and complements approach to updating and that using (op)fibrations.

References

1. F. Bancilhon and N. Spyratos. Update semantics of relational views, *ACM Trans. Database Syst.* 6, 557–575, 1981.
2. M. Barr and C. Wells. *Category theory for computing science.* Prentice-Hall, second edition, 1995.
3. M. Barr and C. Wells. *Toposes, Triples and Theories.* Grundlehren Math. Wiss. 278, Springer Verlag, 1985.
   Available from `ftp://ftp.math.mcgill.ca/pub/barr/ttt/`
4. A. Bohannon, B. Pierce and J. Vaughan. Relational Lenses: A language for updatable views. *Proceedings of ACM PODS-2006*, 338–347, 2006.
5. Z. Diskin and B. Cadish. Algebraic graph-based approach to management of multidatabase systems. In *Proceedings of The Second International Workshop on Next Generation Information Technologies and Systems (NGITS '95)*, 1995.
6. J Foster, M. Greenwald, J. Moore, B. Pierce and A. Schmitt. Combinators for bidirectional tree transformations: A linguistic approach to the view update problem. *ACM Transactions on Programming Languages and Systems*, 29, No. 17, 2007.
7. G. Gottlob, P. Paolini and R. Zicari. Properties and update semantics of consistent views, *ACM Trans. Database Syst.* 13, 486–524, 1988.
8. S. J. Hegner. An order-based theory of updates for closed database views. Annals of Mathematics and Artificial Intelligence, 40, 63–125, (2004).
9. M. Hofmann and B. Pierce. Positive subtyping. *SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL)*, 186-197, 1995.

10. M. Johnson, R. Rosebrugh, and R. J. Wood. Entity-relationship-attribute designs and sketches. *Theory and Applications of Categories* 10, 94–112, 2002.
11. M. Johnson and R. Rosebrugh. Fibrations and universal view updatability. *Theoretical Computer Science*, 388, 109–129, 2007.
12. K. Lellahi and N. Spyratos. Towards a categorical data model supporting structured objects and inheritance. LNCS 504, 86–105,1991.
13. P. O'Hearn and R. Tennent. Parametricity and local variables. *Journal of the ACM* 42, 658–709, 1995.
14. F. J. Oles. A category-theoretic approach to the semantics of programming languages. PhD Thesis, Syracuse University, 1982.
15. F. J. Oles. Type algebras, functor categories and block structure. In *Algebraic methods in semantics*, 543–573. Cambridge Press, 1986.
16. B. Pierce. *Basic category theory for computer scientists*. MIT Press, 1991.
17. B. Pierce and A. Schmitt. Lenses and view update translation. Working Draft, April 2003.
18. F. Piessens and E. Steegmans. Categorical data specifications. *Theory and Applications of Categories*, 1, 156–173, 1995.
19. F. Piessens and E. Steegmans. Selective Attribute Elimination for Categorical Data Specifications. Proceedings of the 6th International AMAST. Ed. Michael Johnson. *Lecture Notes in Computer Science*, 1349:424-436, 1997.