1

Enterprise Software with Half-Duplex Interoperations

Michael Johnson

Macquarie Unviversity, Sydney, Australia, mike@ics.mq.edu.au

Summary. In [3] the author and his colleague Dampney showed how a new solution to view update problems based on a category theory can be used to support enterprise information system interoperation. We now have considerable experience in applying that technique in industry. One of the major applications has been to developing interoperations between large databases. The view update approach to interoperations provides reasonable functionality while limiting the modifications needed to implement it. However, for mathematical reasons the view update approach to interoperations requires a significant amount of attribute harmonisation which means that companies must share data that they sometimes view as confidential. In testing this approach it has become apparent that, like full-duplex communications, it guarantees interoperations in both directions between the cooperating databases. However, in the systems tested interoperations were frequently necessary in one direction only for particular actions (inserts or deletes) on particular entities. This paper introduces a "half-duplex" approach to designing interoperations between extant systems in which data flows both ways between the systems, but on any action on any particular entity it will only flow one way. This significantly extends the domain of application of the technique by avoiding the former need to share information which some organisations insisted on keeping confidential.

1 Introduction

In 2001 Johnson and Rosebrugh introduced a new approach to view update problems for formally specified databases [8]. Since then they and their colleague Dampney have been studying the application of that approach in enterprise data models. Although new solutions to view update problems have a range of applications, the main application has been to developing and supporting interoperations for enterprise software, particularly for information systems [9], [3].

The techniques that we use make major use of precise specification methods, specifically those based on category theory. We were originally motivated

* Research partially supported by the Australian Research Council

to develop such techniques for data models by a very large consultancy which compelled us to use formal methods to manage the complexity. Our methodology has come to be called the *sketch data model* because it is based upon the category theoretic notion of mixed sketch [1]. It has been developed considerably and tested in other consultancies as well as being used to solve view update problems and to support enterprise interoperations.

Typically, our approach to system interoperations requires the solution of *two* view update problems — one for each system. The view update solutions allow data derived from changes in one system to be reflected in a (category theoretically) universal way in the other system. Two solutions provide for two-way interoperability which we refer to as *full-duplex*.

In our experience in analysing actual systems we have found that the full power of full-duplex interoperations is rarely necessary and that the modifications required for full-duplex interoperation can be challenging for business reasons, in particular, because of the need or desire to maintain confidentiality. In this paper we describe a new *half-duplex* methodology. In the new methodology information system interoperation happens in both directions, but data flow and update propagation occur in one direction at a time, with the direction depending upon which entities are being updated and on the nature of the update.

2 Sketch Data Models

The author's approach to the view update problem is based on the Sketch Data Model (SkDM), so to establish notation we briefly review the SkDM here (we do not review standard category theoretic notions which can be found in any introductory textbook on category theory).

In outline, an information system is specified in the SkDM by giving a *schema*, $\mathbb{E}$. A schema is a graph, roughly corresponding to an ER graph, and a set of (categorical) constraints. The constraints take three forms:

1. Commuting diagrams are pairs of paths in the graph with common origin and destination.
2. Limit constraints specify that a certain node in the graph is to act as the "limit" of a specified diagram in the graph.
3. Coproduct constraints specify that a certain node in the graph is to act as the "coproduct" of specified nodes in the graph.

An example of a fragment of a schema graph is shown in Figure 1. In that figure the diamond is required to be a pullback (a kind of limit constraint) and that justifies the name of the node Invoices at Supplier S. The node labeled 1 is also a limit constraint and stands conventionally for the limit of the empty diagram.

An information system, sometimes called a database state or instance, is an assignment of, for every node in the schema a finite set, (the set of

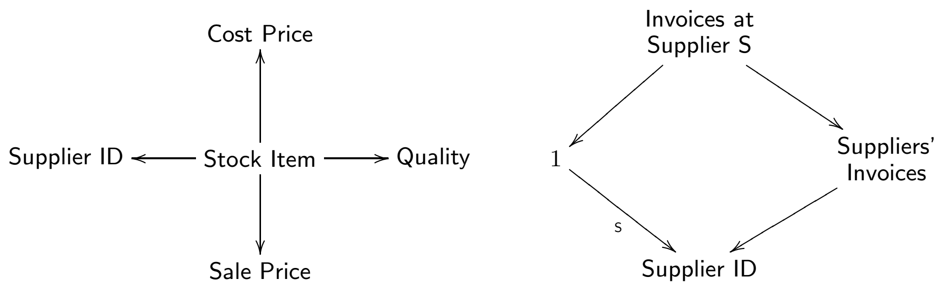

Fig. 1. A fragment of a Customer's database schema

instances or values of that entity or attribute), and for every arrow in the schema a function between the corresponding sets, (the relationships among the entity instances, or the attribute values corresponding to the instances), such that

1. The commuting diagrams do indeed commute as diagrams of corresponding functions
2. The sets assigned to limit nodes are indeed the limits of the corresponding specified diagrams of functions
3. The sets assigned to coproduct nodes are indeed coproducts (disjoint unions) of the sets assigned to the corresponding specified nodes.

The Sketch Data Model has been used extensively in industrial work, and the choice of constraints has been shown to be adequate to model a very wide range of real-world constraints.

Each schema $\mathbb{E}$ generates a *classifying category* $Q(\mathbb{E})$. Roughly speaking it is the smallest category containing the schema, satisfying the constraints, and closed under finite limits and finite coproducts. The classifying category has important technical uses, and the objects of the classifying category $Q(\mathbb{E})$ correspond to the structural queries that can be applied to an information system with schema $\mathbb{E}$ (and this is why the letter Q is used to denote the passage to the classifying category).

Every category $\mathcal{C}$, and so a fortiori every classifying category, has an underlying schema $U\mathcal{C}$. Its graph is the underlying graph of the category, and its constraints are all of the constraints that happen to be true in the category: all of the commuting diagrams, all of the limits, and all of the coproducts. Frequently U is not explicitly mentioned, so if a category appears where a schema is expected an application of U is understood.

A *schema map* between two schema is a graph morphism between the corresponding graphs which maps each of the constraints on the first schema graph to a constraint (already) specified in the second schema graph.

Finally, a view of $\mathbb{E}$ is a schema $\mathbf{V}$ and a schema map $V : \mathbf{V} \longrightarrow UQ(\mathbb{E})$. The schema $\mathbf{V}$ exhibits the view as an information system in the sketch data model, while the map V specifies for each entity how to obtain its data as a query on an information system based on $\mathbb{E}$. In the sequel we will as usual suppress references to U.

For further details we refer the reader to [8].

3 Half-duplex interoperations

The need for half-duplex interoperation is most easily seen by considering interoperating databases with a common entity v which has different sets of attributes A, B, in the two databases. To use the full-duplex interoperation techniques of [9] the databases need to be changed so that both of them have as attributes of v the union of the attribute sets $A \cup B$. This *attribute harmonisation* is usually not difficult to achieve, but it has two negative results

1. The changes may be logically inappropriate for one of the databases since they introduce attributes that do not logically belong in that particular database
2. Especially in interoperations between businesses, it may happen that one (or more) of the attributes to be added is in fact considered confidential by the organisation which uses it. In extreme cases this can lead to limitations on interoperation to preserve confidentiality, and even in those cases where interoperation is obtained the negotiations to release confidential information for attribute harmonisation can be very difficult.

Clearly if a notion of half-duplex interoperation can reduce the need for attribute harmonisation it will facilitate the establishment of interoperation among extant systems.

Attribute harmonisation is necessary for full-duplex interoperation because in the sketch data model methodology interoperation depends upon finding a common view and solving the view update problem for both systems. Furthermore, by [8] Proposition 14, the view update problem will not have a solution at v unless the view has access to all of the attributes $(A \cup B)$ in which case of course both systems must also have access to all of those attributes.

The author has analysed a range of circumstances in real data models where attribute harmonisation was required. In a large majority of the cases I discovered that attribute harmonisation could be avoided or reduced since, at the particular entity in question, updates only ever took place in one direction, say from a database with schema $\mathbb{E}$ to a database with schema $\mathbb{E}'$.

$$\begin{tikzcd} & Q\mathbb{E} \\ \mathbf{V} \arrow[ur] \arrow[dr] & \\ & Q\mathbb{E}' \end{tikzcd}$$

This means that changes of an entity v in the $\mathbb{E}$ database will result in changes of the view which will in turn require changes of the database with schema $\mathbb{E}'$. Thus the view update problem needs to be solved for the view $\mathbf{V} \longrightarrow Q\mathbb{E}'$. However, there is *no* need to solve the view update problem at v for the view $\mathbf{V} \longrightarrow Q\mathbb{E}$, and attribute harmonisation can be significantly reduced. This is the motivation for half-duplex interoperation. We will now provide a formal definition of half-duplex interoperation.

Definition 1 A *half-duplex interoperation* between two information systems with schemata $\mathbb{E}_0$ and $\mathbb{E}_1$ is a schema $\mathbf{V}$ together with schema maps V_0 and V_1

$$\begin{tikzcd} & Q\mathbb{E}_0 \\ \mathbf{V} \arrow[ur, "V_0"] \arrow[dr, "V_1"'] & \\ & Q\mathbb{E}_1 \end{tikzcd}$$

such that for each object v in $\mathbf{V}$, for each action (insert or delete), and for each V_i, $(i = 0, 1)$ either

1. the view update problem has been solved for that action along V_i at v, or
2. the administrator of the database with schema $\mathbb{E}_{1-i}$ has agreed to prohibit that action on instances of $V_{1-i}v$.

Notice that since $V_{1-i}v$ is an object of $Q\mathbb{E}_{1-i}$ the prohibition may effect several entities in the database with schema $\mathbb{E}_{1-i}$.

4 Examples

A simple example which illustrates the methodology arises in standard co-operation between a supplier S and a customer C. The supplier can see the customer's quantity of stock, but cannot change it. The customer on the other hand adjusts the recorded stock levels as products are consumed or sold, and as deliveries arrive. Similarly the customer can see his unpaid invoices as recorded on the supplier's system, but he cannot modify those invoices. Instead the supplier inserts or deletes invoices as orders are dispatched to the customer, or as payments are received. Thus the interoperations are half-duplex with both insert and delete updates flowing from the customer to the supplier for the stock entity, and from the supplier to the customer for the invoice entity.

It is worth examining this example in more detail as it can illustrate a number of features of the methodology. To keep the example simple the schemata will have few constraints and we will explore only very small fragments of the full schemata.

Let $\mathbb{E}_0$ be the schema for the customer information system. The fragment we need to consider is shown in Figure 1. (More typically the schema would not include the node Invoices at Supplier S which would occur automatically in $Q(\mathbb{E}_0)$, but including it, along with the constraint that the diamond is a pullback, will make no difference and saves us from remembering to apply Q.)

We assume that the Customer records in Stock Item each item of stock along with its attributes Cost Price, Sale Price, Supplier ID and Quality. For the purposes of this example we assume that the Customer has agreed to share the Stock Item entity with Supplier S to aid in their cooperation by having Supplier S anticipate Customer C's needs and then offer appropriate stock for possible purchase. However, Customer C does not intend to reveal the suppliers of the stock items held (Supplier S and Supplier S's competitors) nor will the Customer reveal pricing and quality evaluation information.

A fragment of the Supplier's classifying category is shown in Figure 2. Once again the diamonds shown there are required to be pullbacks. This time we will suppose that the schema $\mathbb{E}_1$ does not include the nodes Stock at Customer C and Invoices for Customer C. Rather we suppose that Figure 2 is a fragment of $Q(\mathbb{E}_1)$ which necessarily includes both pullbacks. We also assume that the Supplier has agreed to make Customer C's invoices available to him via the information system interoperations, but of course other customers' invoices will not be visible. Nor will the Supplier reveal information about credit status. Since we seek to encourage interoperations we have provided the Supplier with the facility to store information about stock levels of any of his customers, but we will only ensure interoperation with Customer C.

Now to the interoperations code. To establish interoperations between the Customer and the Supplier we need to find a common view of the two information systems and then solve various view update problems. Finding the common view is now straightforward, since we have in fact constructed the pullbacks that are needed to form that view. We simply let $\mathbf{V}$ consist of two nodes called Stock and Invoice. No edges are needed for $\mathbf{V}$, nor are there any constraints in $\mathbf{V}$. The two schema morphisms V_0 and V_1 carry the two nodes to Stock Item and Invoices at Supplier S and to Stock at Customer C and Invoices for Customer C respectively. We will consider full-duplex and half-duplex interoperations in turn.

$$\begin{tikzcd}
 & \text{Stock at Customer C} \arrow[dl] \arrow[dr] & \\
1 \arrow[dr, "c"'] & & \text{Customers' Stock} \arrow[dl] \\
 & \text{Customer ID} &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \text{Invoices for Customer C} \arrow[dl] \arrow[dr] & \\
1 \arrow[dr, "c"'] & & \text{Customers' Invoices} \arrow[dl] \arrow[d] \\
 & \text{Customer ID} & \text{Credit Status}
\end{tikzcd}$$

Fig. 2. A fragment of a Supplier's classifying category

For full-duplex interoperations we need to solve both view update problems (V_0 and V_1) at both entities of $\mathbf{V}$. This is not possible unless $\mathbf{V}$ can "see" all of the attributes of the entities that are to interoperate — this is where the need for attribute harmonisation arises. In other words the Customer will have to reveal his pricing, quality and stock item supplier information, and the Supplier will have to reveal his credit status information. Fortunately there is no need for the Customer to reveal his invoices from other suppliers, nor for the Supplier to reveal his invoices or stock levels for other customers, because the view update problem can be solved for pullbacks like these by [8] Proposition 15. Nevertheless, the need to reveal confidential information is daunting.

Now for half-duplex interoperations. Since the Customer is the only actor who can adjust stock levels we only need to solve the view update problem for V_1 at the entity Stock. But once again, [8] Proposition 15 applies, and this time we don't have to reveal any confidential information. Similarly inserts and deletes of invoices can only be carried out by the Supplier, so at the Invoice entity we need only solve the view update problem for V_0, and as before it can be solved without revealing confidential information. There is no need for attribute harmonisation in our example.

It should perhaps be noted that this is not a contrived example. When, as often happens, interoperations are one-way for particular entities there is usually relatively little need for attribute harmonisation.

We noted above that the customer-supplier interoperations are half-duplex with both insert and delete updates flowing from the customer to the supplier for the stock entity, and from the supplier to the customer for the invoice entity. There are also examples in which the half-duplex interoperations occur in different directions for the same entity depending on whether the operation is an insert or a delete. By way of example consider a courier operation in which Dispatch and Delivery are carried out by two different companies — the Dispatch company interacts with the customer, collects from them the goods, and charges them for the service, and then pays the Delivery company to carry out the delivery. If the Dispatch and Delivery companies wish to have interoperating information systems they will each presumably have an entity corresponding to Package in Transit, and only the Dispatch company will carry out inserts on that entity, while only the Delivery company will carry out deletes on that entity. (This is a fairly typical situation for an operation where instances of entities are processed in sequence by the different organisations, for example in a production process, or where there is a notion of Check-in and Check-out carried out by the two different organisations.)

Because it is generally easy to solve delete view update problems ([8]) the Dispatch company can keep many attributes of Package in Transit without needing to reveal them for half-duplex interoperability (although they would need to be revealed for full-duplex interoperability). On the other hand, for interoperation to work, even for half-duplex interoperation, the Delivery company's attributes of Package in Transit will need to be available to the view **V**. Fortunately in the range of activities involving production or check-in and check-out it is usually the case that confidential information (credit card numbers, payment details, customer relationship details etc) is recorded at check-in, so again there is rarely need for serious attribute harmonisation.

5 Current work

Recent work [10] has suggested a new and very promising approach for eliminating altogether the need for attribute harmonisation in many cases.

The paper [10] explores three approaches to the use of the sketch data model to formally specify databases where partial information, for example `null` values on attributes, is permitted. What is referred to in that paper as "the second approach" was intended to support attribute harmonisation, but turned out to contain significant technical complications. The other two approaches, the first, and the third, were found to be in most respects equivalent.

Recently the author has been exploring the interaction between those two approaches and the techniques for full-duplex and half-duplex interoperations. To my surprise the formal theory of view updates [8] applies seamlessly to the third approach, but not at all to the first approach. The essence of the difference is as follows.

In the first approach, an attribute which is optional (in the sense that entities may be associated with a `null` value for that attribute) is extended at specification time by the addition of a special value in the sense of Date [4]. Unfortunately, modifications of the special value, perhaps replacing it by a normal value, are not qualitatively different from other attribute value changes. In particular, the universal condition which must be satisfied by view updates is *not* tested against states in which a `null` value is replaced by an actual value.

In the third approach, an attribute which is optional is only defined for an explicitly specified subobject. (In [10] we assumed that the subobject of defined values is complemented but here we will not make that assumption.) In this case recording a normal value for a currently null attribute value for a particular entity amounts to adding a new entity to the subobject along with the actual attribute value associated with that entity. Thus the instantiation of `null` values by normal attribute values *is* tested as part of the universal condition of view updates.

As a result of this difference, if the third approach is used to specify possibly partial attributes, then states which have null values have state morphisms to states in which one or more of the null values are instantiated with normal values. Thus the universal property of insert updates will be satisfied by states which are null at all permissible attributes that do not have values defined in the original model, nor in the update. This in turn means that in many cases the need to share attributes to obtain solutions to view update problems, and hence to obtain interoperation, can be relaxed.

Of course, in a database which does not support partial information we cannot use these new ideas. Our current study of such systems suggests that the half-duplex interoperation methodology does

6 Related work

The use of sketches in data modeling is becoming more widespread. For example Piessens [13], [14] developed a notion of data specification including sketches. He has since obtained results on the algorithmic determination of equivalences of model categories [15]. This Morita theory was intended to support plans for view integration. Diskin and Kadish have used sketches to support category theoretic conceptual modeling in several papers including [5]. Others, including Lippe and ter Hofstede [11], Islam and Phoa [7], and Baclawski et al [2], have used category theory to support data modeling. Rosebrugh et al [6] have developed databases to store finitely presented categories.

Meanwhile, there has been a renewed interest in view updates. Menon et al [12] have used view updating to analyse system inconsistency, and in recent lectures at the Newton Institute Benjamin Pierce has revisited the theory of views.

To the author's knowledge, the notion of half-duplex interoperation and its corresponding partial solutions to view update problems, together with their use in supporting interoperation, is new. He first proposed it in a lecture at Advances in Concurrent Engineering, in 2001, and now that he has accumulated experience in searching for instances of half-duplex interoperation in enterprise applications he is convinced that it is worth developing further.

The proposal to avoid attribute harmonisation altogether, in cases where partial information is supported, is new and relatively unexplored to date.

References

1. M. Barr and C. Wells. *Category theory for computing science*. Prentice-Hall, second edition, 1995.
2. K. Baclawski, D. Dimovici and W. White. A categorical approach to database semantics. *Mathematical Structures in Computer Science*, 4:147–183, 1994.
3. C.N.G Dampney and Michael Johnson. Enterprise Information Systems: Specifying the links among project data models using category theory. In *Enterprise Information Systems* III, Eds J. Filipe, B. Sharp, and P. Miranda, 191–197, Kluwer, 2002.
4. C. J. Date. *Introduction to Database Systems*. Addison-Wesley, eighth edition, 2004.
5. Zinovy Diskin and Boris Kadish. Variable set semantics for keyed generalized sketches: formal semantics for object identity and abstract syntax for conceptual modeling. *Data and Knowledge Engineering*, 47:1–59, 2003.
6. M. Fleming, R. Gunther and Robert Rosebrugh. A database of categories. *Journal of Symbolic Computation*, 35:127-135, 2003.
7. A. Islam and W. Phoa. Categorical models of relational databases I: Fibrational formulation, schema integration. Proceedings of the TACS94. Eds M. Hagiya and J. C. Mitchell. *Lecture Notes in Computer Science*, 789:618–641, 1994.
8. Michael Johnson and Robert Rosebrugh. View updatability based on the models of a formal specification. Formal Methods Europe (FME01). Eds J. Fiadeiro and P. Zave. *Springer Lecture Notes in Computer Science*, 2021:534–549, 2001.
9. Michael Johnson and Robert Rosebrugh. Database interoperability through state based logical data independence. *International Journal of Computer Applications in Technology*, 16:97–102, 2003.
10. Michael Johnson and Robert Rosebrugh. Three approaches to partiality in the sketch data model. *Electronic Notes in Theoretical Computer Science*, 78:1–18, 2003.
11. E. Lippe and A ter Hofstede. A category theoretical approach to conceptual data modelling. *RAIRO Theoretical Informatics and Applications*, 30:31–79, 1996.
12. Catherine Menon, Michael Johnson and Charles Lakos. Inconsistency management and view updates. ENTCS, 141:27–51, 2005.
13. F. Piessens. *Semantic data specifications: an analysis based on a categorical formulation*. PhD thesis, Katholieke Universiteit Leuven, 1996.
14. F. Piessens and Eric Steegmans. Categorical data specifications. *Theory and Applications of Categories*, 1:156–173, 1995.
15. F. Piessens and Eric 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.