Determining Correct View Update Translations via Query Containment*

Anthony Tomasic†
Stanford University

Abstract

Given an intensional database (IDB) and an extension database (EDB), the view update problem translates updates on the IDB into updates on the EDB. One approach to the view update problem uses a translation langauge to specify the meaning of a view update. In this paper we prove properties of a translation language. This approach to the view update problem studies the expressive power of the translation language and the computational cost of demonstrating properties of a translation. We use an active rule based database language for specifying translations of view updates. This paper uses the containment of one datalog program (or conjunctive query) by another to demonstrate that a translation is semantically correct. We show that the complexity of correctness is lower for insertion than deletion. Finally, we discuss extensions to the translation language.

1 Introduction

A deductive database consists of an *intensional* (IDB) and *extensional* database (EDB). In the case of datalog, the meaning of a database is clear [24]. Updates to the EDB are understood as classical relational set oriented updates. However, the meaning of updates to the IDB is ambiguous. To remove this ambiguity, we use an active rule base database language which specifies how to translate a view update on an IDB predicate to updates on EDB predicates.

Example 1 *Consider three research papers a, b, c which contain citations. We denote citations by the predicate e. Let the EDB contain two citations.*

$$e(a,b)$$

$$e(b,c)$$

*Appeared in *Proceedings of the Workshop on Deductive Databases and Logic Programming of the International Conference on Logic Programming (ICLP)*, Santa Margherita Ligure, Italy. Ulrich Geske and Dietmar Seipel (editors), Gesellschaft für Mathematik und Datenverarbeitung MBH, GMD-Studient Nr. 231, pages 75-83, June, 1994.

†Department of Computer Science, Margaret Jacks Hall, Stanford, CA USA 94305-2140. e-mail address: tomasic@cs.stanford.edu

The transitive closure over the predicate e denotes the research papers implicit cited. Let the IDB view tc denote this relationship.

$$\begin{array}{rcl} tc(X,Y) & \leftarrow & e(X,Y) \\ tc(X,Y) & \leftarrow & tc(X,Z)\ \&\ tc(Z,Y) \end{array}$$

Finally, suppose we wish to delete research papers from our database via a view update on the tc relation. Research on the view update problem has shown that the meaning of such an update is ambiguous (see the related work section below for a discussion). To deal with this ambiguity, we permit active rules to define the meaning of view updates explicitly. In our example, we define the deletion of a paper from the view as the same as deleting the citations of the paper.

$$on\ del\ tc(X,Y)\ do\ del\ e(X,Z).$$

This rule translates, for instance, the (view) update del $tc(a,b)$ *on the IDB to the update del* $e(a,Z)$ for all Z *on the EDB and subsequently would delete the fact* $e(a,b)$.

The rule in the example above specifies that the view update "delete a transitive closure arc $tc(X,Y)$ for some X and Y" means the update "delete all outgoing arcs from node X."[1] Thus, given the view update, the database will perform in its place the update.

The EDB update can always be coded directly into the application requesting the view update. However, the principal advantage of view updates is then lost – namely, the independence of updates from schema modification. If the schema is changed, view update translations can automatically be checked for various properties only if they are separate from the application.

Example 2 *Consider an IDB view for transitive closure,*

$$\begin{array}{rcl} tc(X,Y) & \leftarrow & e(X,Y) \\ tc(X,Y) & \leftarrow & tc(X,Z)\ \&\ tc(Z,Y) \end{array}$$

an EDB with two facts,

$$e(a,b)$$

$$e(b,c)$$

and the active rule,

$$on\ ins\ tc(X,Y)\ do\ ins\ e(X,Z)\&\ ins\ e(Z,Y).$$

This rule translates, for instance, the (view) update ins $tc(c,e)$ *on the IDB to the updates ins* $e(c,Z)$ *and ins* $e(Z,e)$ for some Z *on the EDB and would insert, say,* $e(c,d)$ *and* $e(d,e)$ *if* Z *were* d.

This example demonstrates insertion view updates in the same framework. The rule in the example above specifies that the view update "insert a transitive closure arc $tc(X,Y)$ for some X and Y" means the update "insert two arcs $e(X,Z)$ and $e(Z,Y)$ for some Z supplied by the user."

[1] This translation deletes edges from node X which are not on paths to Y and thus is not minimal in some sense. We address this issue by extending the translation language in Section 3.

In the case of deletion, free variables in the translation (Z in Example 1) are *universally* quantified, and all tuple instances of the corresponding predicate are deleted. In the case of insertion, the free variables in the translation (Z in Example 2) are *existentially* quantified, and the tuple instances of the corresponding predicate are inserted. We assume here that the additional constants needed for an insertion are provided by the user or some other source [18].

The particular translation of a deletion of a transitive closure arc in Example 1 insures the *correctness* property of the translation. That is, whatever IDB fact f appears in the deletion view update, f will not be modeled (derived) in the database that results from the view update translation. Translations with this property are *semantically correct* [17] or simply *correct*. Similarly, in the case of a view update insertion of an IDB fact f, a correct translation insures that f will be modeled in the resulting database. We believe that testing a translation for this property is a useful function of a deductive database system. In this paper we show how to determine if a translation is correct.

Of course, many other correct translations for $del\ tc(X, Y)$ are possible (e.g., delete the incoming arcs to node Y). The variety of correct translations for a view is the source of the ambiguity of the view update problem. Since our approach is based on a translation language, all correct translations which can be expressed in the translation language are equally valid. Since correctness is a property which our method demonstrates about translations, we would like to cover as large a class of translations as possible.

In general, active rules permit arbitrary changes to the database. In this paper we consider only *translations*: a subset of active rules which specify the meaning of view updates. Our method is unique in that correctness for translations onto recursive views is demonstrated. A larger class of translations is desirable – however the computational complexity of proving properties on a more expressive translation language quickly becomes intractable. We relate complexity results on query containment [4] to the complexity of proving correctness for view update translations in Section 2.

1.1 Related Work

In addition to the relational framework [1, 3, 5, 6, 7, 8, 12, 18], the view update problem has been attacked from various logical vantage points. One method extends the semantics of the database to express some or all the possible correct translations of a view update [10, 21, 25] or to directly store the view updates and provide new semantics for the database [16]. This approach increases the complexity of query processing. The approach presented here does not increase the complexity. The opposite approach, to restrict the class of translations in an attempt to compute a unique result [11], has also been studied. We believe this work is limited in scope. Another approach classifies the types of ambiguity in relational view updates [14, 18] and deals with each type of ambiguity in an *ad hoc* manner. Related work on translation editors [15, 19] computes the implications of a view update translation and presents decisions to resolve ambiguity to the database administrator. We believe that correctness is a property of interest to database administrators and would be reported by these editors. The addition of integrity constraints clearly impacts translations and [23] considers extracting information from functional dependencies. For datalog, [20] considers view updates for deletion but defines insertion as the insertion of IDB facts. Another methods [2, 9, 13], closely related to conjunctive query containment, generate all possible translations of a view update. The number of possible translations of a view update is very large, and we believe that usually a database adminstrator knows the correct translation of a view update and simply needs a language to express the translation.

The approach taken here is most closely related to DLP [17] which introduced a language and several criteria for translations. DLP also extends the semantics of the database to include updates whereas we retain the standard semantics. Our work can be viewed as a method for showing correctness for a subclass of DLP translations. This paper enlarges the class of translations which can be decidably shown as correct for datalog. Finally, a simple translation language is discussed in [22].

In the next section we formally present a framework and prove the correctness of translations can be tested by using containment. In Section 3 we discuss extensions and limitations of this approach.

2 Datalog

In this section we define a class of translations and prove a method for testing correctness of a translation with respect to a datalog program.

Definition 1 *A* program *P is a pair (E, I) consisting of an IDB of datalog rules I and an EDB of facts E. Throughout this paper the IDB of a program is fixed over a view update translation.*

Following Example 1, $E = \{e(a,b), e(b,c)\}$ and $I = \{tc(X,Y) \leftarrow e(X,Y), tc(X,Y) \leftarrow tc(X,Z)\& tc(Z,Y)\}$.

In active rule databases, a rule can specify that some update trigger an arbitrary collection of updates. Here, we consider a subset of rules which translate insertions (deletions) on a single IDB predicate into insertions (deletions) on EDB predicates. For convenience in the following proofs, we write these rules as datalog.

Definition 2 *A* translation *is an insertion translation of the form ins R or a deletion translation of the form del D. R is a datalog rule of the form $H \leftarrow G_1 \& \cdots \& G_n$ where H is an IDB predicate and each G_i is an EDB predicate. D is a set of rules, each of which has the same form as R.*

The active rule in Example 1 is written *del* $tc(X,Y) \leftarrow e(X,Z)$. The syntactic transformation from datalog to the corresponding active rule is straightforward; elide the references to *ins* and *del* on the right hand side of an active rule and replace *on* with $\leftarrow$.

Note that the rule appearing in an insertion translation is a conjunctive query and the rule appearing in a deletion translation is a nonrecursive datalog program.

Definition 3 *A* view update *u is of the form ins F or del F. F is an IDB predicate. F is a fact.*

The view update of Example 1 is *del* $tc(a,b)$.

Definition 4 *A* view update translation *of a view update u, a translation t and a program P where $P = (E, I)$ is a program $P' = (E', I)$ where E' is of the form*

1. *if u = del F and $\sigma_i(H_i) = F$ and t_i = del $H_i \leftarrow G_{i1} \dots G_{ij} \dots G_{in_i}$, then $E' = E - \bigcup_{ij} \forall \theta_k(\sigma_i(G_{ij}))$ such that $\theta_k(\sigma_i(G_{i,j})) \in E$,*
   where σ_i is the MGU of H_i and F and the deleted tuples are ground by virtue of θ (which is universally quantified), or

2. *if* $u = ins\ F$ *and* $\sigma(H) = F$ *and* $t = ins\ H \leftarrow G_1 \& \cdots \& G_n$*, then* $E' = E \cup \bigcup_i \theta(\sigma(G_i))$*, where* σ *is the MGU of* H *and* F *and the added tuples are ground by virtue of* θ *(supplied by the user).*

For Example 1, $u = del\ tc(a, b)$, $\sigma_1 = [X/a, Y/b]$, $H_1 = tc(X, Y)$, $t_1 = del\ tc(X, Y) \leftarrow e(X, Z)$, $\theta_1 = [Z/b]$, $E = \{e(a, b), e(b, c)\}$, and $E' = \{e(b, c)\}$.

For Example 2, $u = ins\ tc(c, e)$, $\sigma = [X/c, Y/e]$, $H = tc(X, Y)$, $t = ins\ tc(X, Y) \leftarrow e(X, Z) \& e(Z, Y)$, $\theta = [Z/d]$, $E = \{e(a, b), e(b, c)\}$, and $E' = E \cup \{e(c, d), e(d, e)\}$.

Note that θ is for variables which appear only in the body of a translation. For insertion, θ provides (possibly new) constants for these variables. For deletion, the various instances of θ provide constants which match all the EDB facts which satisfy a rule.

Definition 5 *A view update translation* P' *of* (u, t, P) *is* correct *if*

1. $u = del\ F$ *and* $P' \not\models F$*, or*

2. $u = ins\ F$ *and* $P' \models F$*.*

Definition 6 *A translation* t contains *a program* P *where* $P = (E, I)$ *if* t *is* $ins\ R$ *and* $R \subseteq I$ *or if* t *is* $del\ D$ *and* $I \subseteq D$*.*

By $A \subseteq B$ we mean that if $A \models F$ then $B \models F$ (for all possible EDB). An algorithm to test if a conjunctive query is contained by a program ($R \subseteq I$) is given in [24, Algorithm 14.2]. An algorithm to test if a program is contained by a nonrecursive datalog program ($P \subseteq D$), is given in [4]. Both these algorithms are independent of the EDB.

Theorem 1 *If* t *contains* P*, then a view update translation* P' *of* (u, t, P) *is correct.*

Proof. If $u = ins\ F$ then we must show $P' \models F$ or equivalently $E' \cup I \models F$. Suppose $P \models F$, then we are done, since the view update translation only adds facts to generate P', and datalog is monotonic. Suppose $P \not\models F$. Let $t = ins\ R$. The view update translation adds a set of facts of the form $g_i = \theta(\sigma(G_i))$ to the database. Since $\sigma(H) = F$, the g_i obtained by applying $\theta(\sigma(\cdot))$ to the body of R must derive F i.e. $E' \cup R \models F$. Thus, $E' \cup I \models F$ since $R \subseteq P$. (Note that the additional constants added by θ cannot appear in H since F is a fact.)

If $u = del\ F$ then we must show $P' \not\models F$ or equivalently $E' \cup I \not\models F$. Suppose $P \not\models F$, then we are done, since the view update translation only deletes facts to generate P', and datalog is monotonic. Suppose $P \models F$. Let $t = del\ D$. Then $E \cup D \models F$ since $I \subseteq D$. By inspection of the definition of view update translation, $E' \cup D \not\models F$ (the definition deletes every EDB fact in every proof of F). Thus, $E' \cup I \not\models F$ since $I \subseteq D$. QED.

Note that the above proof rests on containment of programs in a way that is unnecessarily strong. For instance, consider the program I

$$\begin{aligned} p(X) &\leftarrow q(X) \\ q(X) &\leftarrow r(X) \end{aligned}$$

and the correct deletion translation $D = del\ p(X) \leftarrow r(X)$. Technically, $I \not\subseteq D$ because of the predicate q. However, we are interested in containment only with respect to the predicate p. We believe that the extension of containment to containment "relative" to a predicate (p in this case) is straightforward and we assume containment is relative for the rest of the paper.

The use of containment to demonstrate correct translations permits an expressive form of insertion. For example, to insert into transitive closure, we can write the correct translation $ins\ tc(X,Y) \leftarrow e(X,Y)$, or $ins\ tc(X,Y) \leftarrow e(X,Z)\ \&\ e(Z,Y)$, etc. Thus, we can correctly define a finite path of any length as a translation for the view update.

For the insertion case, testing containment is computationally cheap. However, for deletion this flexibility has an associated price. The computational cost of determining containment is triplely exponential in the deletion case. However, if the view for which the translation is define is not recursive, the computation cost drops to exponential. There is work on polynomial time containment testing for a subclass of datalog for the insertion case [24], but for the deletion case this issue remains open.

3 Extensions

In this section we informally discuss some extensions and limitations to the method presented in the previous section. One extension along the lines of DLP [17] involves querying the database as part of the translation. For example, consider insertion into the view

$$p(X) \quad \leftarrow \quad q(X)\ \&\ r(X)$$

One possible correct translation is $ins\ p(X) \leftarrow q(X)\ \&\ r(X)$. Suppose, however, that we wish to translate the view update only if $p(X) \leftarrow q(X)$ is already true. We extend the notation of translations to include parenthesis to mean a query on the database. Thus, the translation $ins\ p(X) \leftarrow (q(X))\ \&\ r(X)$ would for a view update $ins\ p(a)$ query the database for $q(a)$. If $q(a)$ is in the database, the view update translation would proceed and insert $r(a)$. However, if $q(a)$ is not in the database, the user (or application) would be signaled with an exception. The proofs in the previous section can be easily extended to determine correctness given that any queries in the translation process are satisfied.

For stratified datalog, the syntax of translations can be extended to have insert and delete in the body such as the translation $ins\ p(X) \leftarrow ins\ q(X)\ del\ r(X)$ for the view $p(X) \leftarrow q(X)\ \&\ \neg r(X)$. This extension permits much more expressive power in handling translations by modification of the rules themselves. For instance, the translation of updates to the EDB can be "bypassed" by adding predicates to the EDB to store updates. In the above view, the translation $del\ p(X) \leftarrow ins\ r(X)$ can be viewed as simply recording the deletion view update in the relation r. (This flexibility is also available in DLP.)

There are some limitations to our translation language however. Consider again transitive closure. We can the write the correct translation $del\ tc(X,Y) \leftarrow e(X,Z)$ but this is a crude way of removing a path. For instance, a reasonable and correct way to accomplish the translation is to "delete all the edges on any path from X to Y". This translation cannot be expressed in our language. The problem stems from the fact that the view update translation inflexibly falsifies the body of a rule – it simply deletes all EDB facts which match. For example, consider the view $p \leftarrow q(X)\ \&\ r(X)$ and the EDB consisting of four facts $\{q(a), r(a), q(b), r(b)\}$. The translation $del\ p \leftarrow q(X)$ removes the q facts and the translation $del\ p \leftarrow r(X)$ remove the r facts, but there is no correct translation which removes $q(a)$ and $r(b)$ although the resulting EDB $\{r(a), q(B)\}$ is correct.

4 Conclusion

In this paper we have shown a method which determines if a translation encodes a natural semantics for view updates. Namely, that after an insertion of a fact, the fact is modeled in the resulting database and after a deletion of a fact, the fact is not modeled in the resulting database. The proof of the method relies directly on the containment of conjunctive queries by datalog programs and the containment of datalog programs by nonrecursive datalog programs. Thus, various results on the complexity of containment problems also apply to checking translations. In addition, we discuss extensions to the translation language for stratified datalog and discuss limitations of the described approach.

Acknowledgements: Thanks to Surajit Chaudhuri, Ashish Gupta, Jeff Ullman, Tak Yan, and the anonymous referees for comments and discussions on the subject of this paper.

References

[1] F. Bancilhon and N. Spyratos. Update semantics of relational views. *ACM Transactions on Database Systems*, 6(4):557–575, 1981.

[2] Francois Bry. Intensional updates: Abduction via deduction. In *Proceedings of the Seventh International Conference On Logic Programming*, 1990.

[3] C. R. Carlson and A. K. Arora. The updatability of relational views based on functional dependencies. In *Proceedings COMPSAC 79*, pages 415–420, 1979. Reprinted in *Tutorial: Database Management in the 1980's*, J. A. Larson and H. A. Freeman (editors), IEEE Computer Society Press, 1981.

[4] Surajit Chaudhuri and Moshe Y. Vardi. On the equivalence of recursive and nonrecursive datalog programs. In *Proceedings of the Eleventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, San Diego, 1992.

[5] S. S. Cosmadakis and C. H. Papadimitriou. Updates of relational views. *Journal of the ACM*, 31(4):742–760, 1984.

[6] C. J. Date. *Relational Database: Selected Writings.* Addison-Wesley, Reading, Massachusetts, 1986.

[7] J. E. Davidson. *Interpreting Natural Language Database Updates.* PhD thesis, Stanford University, 1984.

[8] U. Dayal and P. A. Bernstein. On the correct translation of update operations on relational views. *ACM TODS*, 8(3):381–416, 1982.

[9] Hendrik Decker. Drawing updates from derivations. In *Proceedings of the Third International Conference on Database Theory (ICDT '90)*, 1990.

[10] R. Fagin, J. D. Ullman, and M. Y. Vardi. On the semantics of updates in databases. In *Proceedings 2nd ACM Symposium on Principles of Database Systems*, pages 352–365, Atlanta, 1983.

[11] Stephen J. Hegner. Foundations of canonical update support for closed database views. In *Proceedings of the Third International Conference on Database Theory (ICDT '90)*, 1990.

[12] B. E. Jacobs. Application of database logic to the view update problem. Technical Report TR 960, University of Maryland, College Park, 1980.

[13] A. C. Kakas and P. Mancarella. Database updates through abduction. In *Proceedings of the 16th VLDB Conference*, Brisbane, Australia, 1990.

[14] A. M. Keller. Algorithms for translating view updates to database updates for views involving selections, projections, and joins. In *PODS '85*, pages 154–163, 1985.

[15] A. M. Keller. Choosing a view update translator by dialog at view definition time. In *Proceedings of VLDB*, pages 467–474, Kyoto, 1986.

[16] D. Laurent, V. Phan Luong, and N. Spyratos. Updating intensional predicates in deductive databases. In *Proceedings of the Ninth IEEE International Conference on Data Engineering (ICDE)*, 1993. (To Appear).

[17] Sanjay Manchanda and David Scott Warren. A logic-based language for database updates. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann, 1987.

[18] Y. Masunaga. A relational database view update translation mechanism. In *Proceedings of VLDB*, pages 309–320, Singapore, 1984.

[19] C. B. Medeiros and F. Wm. Tompa. Understanding the implications of view update policies. *Algorithmica*, 1:337–360, 1986.

[20] J.-M. Nicolas and K. Yazdanian. An outline of BDGEN: a deductive DBMS. In R. E. A. Mason, editor, *Proceedings of IFIP 83*, pages 711–717, 1983.

[21] Francesca Rossi and Shamim A. Naqvi. Contributions to the view update problem. In *Proceedings of the Sixth International Conference on Logic Programming*, Lisbon, 1989.

[22] Anthony Tomasic. View update translation via deduction and annotation. In *ICDT '88 2nd International Conference on Database Theory*, pages 338–352, 1988. Springer-Verlag Lecture Notes in Computer Science 326.

[23] Riccardo Torlone and Paolo Atzeni. Updating deductive databases with functional dependencies. In *Proceedings of the Second International Conference on Deductive and Object-Oriented Databases (DOOD '91)*, 1991.

[24] Jeffrey D. Ullman. *Principals of Database and Knowledge-Base Systems*, volume 2: The New Technologies. Computer Science Press, 1989.

[25] M. W. Wilkins. A model-theoretic approach to updating logical databases. Technical Report STAN-CS-86-1096, Stanford University, 1986.