RJ 3700 (42776) 1/14/83
Computer Science

ON THE SEMANTICS OF UPDATES IN DATABASES: Preliminary Report*

Ronald Fagin
IBM Research Laboratory
San Jose, California 95193

Jeffrey D. Ullman**
Moshe Y. Vardi***
Department of Computer Science
Stanford University
Stanford, California 94205

ABSTRACT: We suggest here a methodology for updating databases with integrity constraints and rules for deriving inexplicit information. First we consider the problem of updating arbitrary theories by inserting into them or deleting from them arbitrary sentences. The solution involves two key ideas: when replacing an old theory by a new one we wish to minimize the change in the theory, and when there are several theories that involve minimal changes, we look for a new theory that reflects that ambiguity. The methodology is also adapted to updating databases, where different facts can carry different priorities, and to updating user views.

*To appear, Proc. Second ACM SIGACT-SIGMOD Symp. on Principles of Database Systems, Atlanta, March 1983.
**Research partially supported by AFOSR grant 80-0212.
***Research supported by a Weizmann Post-doctoral Fellowship, Fulbright Award, and NSF grant DCS-81-12907.

ON THE SEMANTICS OF UPDATES IN DATABASES

Preliminary Report

Ronald Fagin
IBM Research Laboratory
San Jose, California 95193

Jeffrey D. Ullman[†]
Moshe Y. Vardi[‡]
Department of Computer Science
Stanford University
Stanford, California 94305

Abstract

We suggest here a methodology for updating databases with integrity constraints and rules for deriving inexplicit information. First we consider the problem of updating arbitrary theories by inserting into them or deleting from them arbitrary sentences. The solution involves two key ideas: when replacing an old theory by a new one we wish to minimize the change in the theory, and when there are several theories that involve minimal changes, we look for a new theory that reflects that ambiguity. The methodology is also adapted to updating databases, where different facts can carry different priorities, and to updating user views.

1. Introduction

The ability of the database user to modify the content of the database, the so-called *update* operation, is fundamental to all database management systems. Since many users do not deal with the entire conceptual database but only with a view of it, the problem of view updating, i.e., translating updates on a user view into updates of the actual database, is of paramount importance, and has been addressed by several works, e.g., [BS, CA, Cl, Da, DB1, DB2, FS, Ja2, KD, Ke, Kl, Os]. An assumption that underlies all of these works is that only the view update issue is problematic (because of the ambiguity in translating view updates into database updates), and that the issue of updating the database directly is quite clear.

Some works ([NY, Sc, To]) have realized that there are difficulties in updating the database when it has to satisfy integrity constraints. Consider for example a database of propositional facts with the integrity constraint $A\&B \rightarrow C$. If the database is initially $\{A\}$ and we are asked to insert B, then we have a problem because the state $\{A,B\}$ is illegal. This seems to be easy to fix, because we can insert C automatically and get the legal state $\{A,B,C\}$. But what if we are now asked to delete C? It is not clear at all what is the operation to be taken.

While in the above example it seems that the integrity constraint is the source of the problem, we believe that even in the absence of such constraints the semantics of updates on the database itself is not completely clear. Consider for example a relational database with a ternary relation $SUPPLIES$, where a tuple

† Research partially supported by AFOSR grant 80-0212.

‡ Research supported by a Weizmann Post-doctoral Fellowship, Fulbright Award, and NSF grant DCS-81-12907.

$\langle a,b,c \rangle$ means that supplier a supplies part b to project c. Suppose now that the relation contains the tuple $\langle$*Hughes*, *tiles*, *Space Shuttle*$\rangle$, and that the user asks to delete this tuple. A simple-minded approach would be to just go ahead and delete the tuple from the relation. However, while it is true that Hughes does not supply tiles to the Space Shuttle project anymore, it is not clear what to do about three other facts that were implied by the above tuple, i.e, that Hughes supplies tiles, that Hughes supplies parts to the Space Shuttle project, and that the Space Shuttle project uses tiles. In some circumstances it might not be a bad idea to replace the deleted tuple by three tuples with null values:

$\langle$*Hughes*, *tiles*, *NULL*$\rangle$,

$\langle$*Hughes*, *NULL*, *Space Shuttle*$\rangle$,

and

$\langle$*NULL*, *tiles*, *Space Shuttle*$\rangle$.

The common denominator to both examples is that the database is not viewed merely as a collection of atomic facts, but rather as a collection of facts from which other facts can be derived. It is the interaction between the updated facts and the derived facts that is the source of the problem. This is exactly the source of the problem in the case of view updates, since the view is really a collection of facts that are derived from the database. Thus, understanding the semantics of updates in general will hopefully also lead to a solution of the view update problem.

We believe that the appropriate framework for studying the semantics of updates is one in which we treat the database as a consistent set (not necessarily finite) of statements in first-order logic, i.e., a *theory*. A theory is a description of the world, but it is not necessarily a complete description. Every state that is a model of the theory is a possible state of the world. Thus, the database can be viewed as an exact description of our knowledge of the world. This approach is propounded in [Ko, MUV, NG, Rei] for several reasons: It has the advantage of uniformity in treating atomic facts, integrity constraints, and derivation rules, which are all expressed as sentences in first-order logic. It offers ease in modeling partial information [Mi, Rei]. It also facilitates defining a universal relation interface [MUV].

When one tries to update a theory by inserting, deleting, or replacing some first-order statement, several new theories can accomplish the update. It seems reasonable that we would like to change the existing theory as little as possible while still accomplishing the update. That is, some partial order should be defined on the possible new theories, a partial order that reflects the divergence of the new theory from the old one, and only theories that are minimal with respect to this partial order should be considered. This idea is originally due to Todd [To], who considered only theories of atomic facts.

If there is no unique minimal new theory, then the update does not give us enough information about the possible new states of the world. While Todd considers such a case as an illegal update, we see it as a case of incomplete information, and we believe that the new theory should reflect this state of knowledge. In fact, since an update reflects the user's most recent piece of knowledge, we believe that no update should be illegal. If the user insists that his update is correct, then the theory should be modified to reflect this new piece of knowledge. (However, it could be the case that the new state of knowledge is not expressible in first-order logic).

There is, however, a marked difference between databases and arbitrary theories. In a theory all sentences carry the same weight, in the sense that there is no way to prefer one over the other. On the other hand, in a database some parts are easier to update than others. For example, while a salary change in the appropriate relation is a daily routine, changing the integrity constraints governing this relation occurs usually only in the course of a database reorganization. Thus, we have to modify our framework to accommodate for that difference. Giving different parts of the theory greater weight than others is similar in spirit to the idea of "modal categories" in [Res].

We see the view update problem as a specific example of the more general problem of updating databases under constraints. Works on view updates often encounter difficulties that, we believe, are attributable to an underlying dubious assumption. The assumption is that we know exactly what change to the view is desired by the user. These works see the central problem as that of reflecting that change "correctly" in the database. Rather, we propose to look at a request to update a view as supplying an information unit in terms of the view. To implement the update, we interpret that information unit in terms of the database and then treat it as an ordinary update.

While the application that we have in mind here is updating databases, we believe that the framework developed here is also relevant to any kind of knowledge base management system. From the point of view of Artificial Intelligence, what we have here is a logic for *belief revision*, that is, a logic for revising a system of beliefs to reflect perceived changes in the environment or acquisition of new information. The reader who is interested in that aspect is referred to [DL].

What we propose in this preliminary report is mainly the logical framework, without getting down to implementation details. We do, however, give one concrete example. We shall discuss more of the implementation aspect in the full paper.

2. Updating theories

Our basic units of information are first-order well-formed formulas with no free variables. Two extreme cases should, however, be excluded. Inconsistent formulas represent contradictory information, and valid formulas represent always-true information. Since we want to prevent insertions of contradictory information or deletions of always-true information, we define a *sentence* to be a first-order well-formed formula with no free variables that is neither valid nor inconsistent. A *theory* is a consistent set of sentences. The set of logical consequences of a theory T is denoted T^*, i.e.,

$$T^* = \{\sigma : T \text{ logically implies } \sigma\}.$$

If $T = T^*$ then T is closed with respect to logical implication, and we say that it is a *closed* theory. We will sometimes restrict ourselves to closed theories. The class of all models of a theory T is denoted by $Mod(T)$. In our context, models of T are databases that obey T. Our results hold regardless whether we consider arbitrary models or just finite models.

When the user asks for an update, he intends to replace the existing theory by a new one[1]. Let T be a theory, and let σ be a sentence. A theory S *accomplishes the insertion of* σ *into* T if $\sigma \in S$. S *accomplishes the deletion of* σ *from* T if $\sigma \notin S^*$.

Some observations should be made with regard to this definition. We consider the sentences in T as basic facts from which all the sentences in T^* follow. Thus, there is a difference between the sentences in T and the sentences in $T^* - T$, in the sense that those in T are represented explicitly, while those in $T^* - T$ are not, even though they logically follow from the sentences that are in T. When we insert σ into T we want it to be represented explicitly, so we require that $\sigma \in S$ in order for S to accomplish the insertion. On the other hand, it is not sufficient to have $\sigma \notin S$ for S to accomplish the deletion of σ from T, and we require that $\sigma \notin S^*$. (If only closed theories are considered, then we do not have this distinction between insertion and deletion. We deal with updates of closed theories in the end of this section.) Also, we distinguish between the deletion of σ and the insertion of $\neg\sigma$. We insert $\neg\sigma$ when we know that σ is not true anymore, and we delete σ when we do not know anymore that σ is true.

If $T \cup \{\sigma\}$ is consistent, then obviously this theory accomplishes the insertion and should be taken as the

[1] We consider here only insertions and deletion. Replacements will be dealt with in the full version of the paper.

result of the update. Similarly, if $\sigma \notin T^{*}$, then T itself can be taken as the result of deleting σ from T. The interesting cases are when we try to insert σ into T, where $T \cup \{\sigma\}$ is inconsistent, or when we try to delete σ from T, where $\sigma \in T^{*}$.

Suppose now that we are trying to update a theory T, and that T_1 and T_2 are two theories that accomplish the update. Each of T_1 and T_2 is different from T. We would clearly prefer the theory that involves a "smaller change" from T. In order to formalize this intuitive notion of "smaller change" when going from a theory T to a theory S, we have to consider the set $S-T$ of facts that are added to the theory, and the set $T-S$ of facts that are deleted from the theory.

We say that T_1 has *fewer insertions than* T_2, *with respect to* T, if $T_1 - T \subset T_2 - T$; T_1 *has no more insertions than* T_2, *with respect to* T, if $T_1 - T \subseteq T_2 - T$; and T_1 *has the same insertions as* T_2, *with respect to* T, if $T_1 - T = T_2 - T$.[2] Similarly, we say that T_1 *has fewer deletions than* T_2, *with respect to* T, if $T - T_1 \subset T - T_2$; T_1 *has no more deletions than* T_2, *with respect to* T, if $T - T_1 \subseteq T - T_2$; and T_1 *has the same deletions as* T_2, *with respect to* T, if $T - T_1 = T - T_2$. We shall omit reference to T when it is clear from the context.

Clearly, we would like to minimize both the set of inserted facts and the set of deleted facts, and this is why Todd [To] says that T_1 does not involve a greater change than T_2 with respect to T if T_1 has no more insertions and no more deletions than T_2.

We, however, contend that the notion of "smaller change" can not be defined precisely without considering the nature of the update. The next lemma claims that when dealing with deletions it suffices to consider the set of deleted facts.

[2] We use $\subseteq$ to denote containment and $\subset$ to denote proper containment.

Lemma 1. Let T be a theory and let σ be a sentence. Then for each theory S that accomplishes the deletion of σ from T, there is a theory S' such that

(1) S' accomplishes the deletion of σ from T,

(2) $S' \subseteq T$, and

(3) S' has the same deletions as S. □

It follows that when dealing with deletions it suffices to consider the set of deleted facts. Let us now consider insertions. It would have been nice if, analogously to the case of deletions, it would have sufficed to consider the set of added facts. Unfortunately, this is not the case. Consider a propositional theory with the propositions A and $\neg(A \& B)$. Suppose that we want to insert B. It is clear that either A or $\neg(A \& B)$ must be deleted. Thus, consideration of the set of the deleted facts is unavoidable. The reason for the discrepancy between deletions and insertions is that consistency is preserved when going to subsets but not when going to supersets.

The next lemma suggests that in the case of an insertion if we were to minimize the set of inserted facts first, or if we were to minimize both sets of inserted and deleted facts simultaneously, then the result would not be very useful.

Lemma 2. Let T be a closed theory, let σ be a sentence not in T, and let T' be the closed theory $\{\sigma\}^{*}$. Then there is no closed theory S that accomplishes the insertion of σ into T, such that either S has fewer insertions than T', or S has the same insertions as T' but S has fewer deletions than T'. □

The idea is that if we do not wish to add anything that does not follow from σ, then we must delete everything not following from σ.

As a consequence of Lemmas 1 and 2 we believe that with both deletions and insertions we should try first to minimize the set of deleted facts. This is justified on the intuitive grounds that we would like to stick with as many as possible of the facts that were known to be true.

Thus, we say that T_1 accomplishes an update u of T *with a smaller change than* T_2 if both T_1 and T_2 accomplish u, and either T_1 has fewer deletions than T_2 or T_1 has the same deletions as T_2 but T_1 has fewer insertions than T_2. Observe that when u is a deletion we can assume, by Lemma 1, that there are no inserted facts.

Now that we have defined formally the notion of "smaller change" we can define the notion of "minimal change". We say that S accomplishes an update u of T *minimally* if there is no theory S' that accomplishes u with a smaller change than S.

The above definition is non-constructive in the sense that it does not give us any clue as to how to find those theories that accomplish an update minimally. The following theorem gives a constructive equivalent condition.

Theorem 1. Let S and T be theories, and let σ be a sentence. Then,

(1) S accomplishes the deletion of σ from T minimally if and only if S is a maximal subset of T that is consistent with $\neg\sigma$, and

(2) $S \cup \sigma$ accomplishes the insertion of σ into T minimally if and only if S is a maximal subset of T that is consistent with σ. □

Observe that there is an interesting duality in the theorem:

Corollary. S accomplishes the deletion of σ from T minimally if and only if $S \cup \neg\sigma$ accomplishes the insertion of $\neg\sigma$ into T minimally. □

If when trying to update a theory T there is a unique theory S that accomplishes the update minimally, then clearly we would take S as our new theory. What, however, should be done if several theories accomplish the update minimally? Consider for example the propositional theory $T=\{A\&B\rightarrow C, A, B, C\}$. The reader can verify that there are three theories that accomplish the deletion of C from T minimally: $T_1=\{A\&B\rightarrow C, A\}$, $T_2=\{A\&B\rightarrow C, B\}$, and $T_3=\{A, B\}$.

Our contention is that a theory T is a description of the class $Mod(T)$ of possible worlds. If $T_1, \ldots, T_n, \ldots$ are the theories that accomplish an update of T minimally, then the only thing we do know for sure after the update is that the world must be a model of some T_i; that is, our class of possible worlds is the class $\bigcup_{i\geq 1} Mod(T_i)$. It is not a priori clear that this class is elementary, i.e., can be axiomatized by a first-order theory (it can be shown that this class can be axiomatized in an infinitary language). Nevertheless, if it is elementary then its theory is the new state of knowledge and should be taken as the new theory. That is to say, if $\bigcup_{i\geq 1} Mod(T_i)=Mod(T')$, then T' is a *result* of the update of T. Observe that there can be more than one result of an update, but all results are logically equivalent.

One case for which we know that updates are well-defined is when there are only finitely many theories that accomplish the update minimally. Let $T_1, \ldots, T_n$ be theories. The *disjunction* of these theories is

$$\bigvee_{i=1}^{n} T_i=\{\tau_1\vee\cdots\vee\tau_n : \tau_i\in T_i, 1\leq i\leq n\}$$

Lemma 3. Let $T_1, \ldots, T_n$ be theories, let T' be the theory $\bigvee_{i=1}^{n} T_i$, and let T'' be $\bigcap_{i=1}^{n} T_i^{*}$. Then $Mod(T')=Mod(T'')=\bigcup_{i=1}^{n} Mod(T_i)$. □

Thus, if $T_1, \ldots, T_n$ are the theories that accomplish an update u minimally, then T' and T'' given in the lemma are results of u.

Suppose now that we try to update a finite theory T. In that case there are only finitely many theories that accomplish the update minimally, and by Lemma 3 we are guaranteed that there exists a result of the update. Furthermore, in that case all the theories that accomplish the update minimally are finite, so the theory T' given in the lemma is also a finite theory (note that T'' is infinite).

That means that we can again update it and get a well-defined result.

Another case for which updates are well defined is the case of closed theories. By that we mean that we consider only updates of closed theories, and among the theories that accomplish the update minimally we consider only the closed theories. Formally, let T be a closed theory, let u be an update, and let $T_1, \ldots, T_n, \ldots$ be the closed theories that accomplish u minimally. T' is the *closed result* of the update u if T' is closed and $Mod(T') = \bigcup_{i \geq 1} Mod(T_i)$. Observe that T', if it exists, is unique. The following theorem guarantees the existence of the result.

Theorem 2. Let T be a closed theory, let u be an update, and let $T_1, \ldots, T_n, \ldots$ be the closed theories that accomplish u minimally. Then $T' = \bigcap_{i \geq 1} T_i$ is the closed result of the update u. □

While the above theorem guarantees the existence of the result, the description of the result by a possibly infinite intersection is highly non-constructive. The next theorem describes explicitly the closed result of an update of a closed theory.

Theorem 3. Let T be a closed theory, and let σ be a sentence. If $\sigma \in T$, then the closed result of deleting σ from T is $(\{\neg\sigma\} \vee T)^*$. If $\sigma \notin T$ then the closed result of inserting σ into T is $\{\sigma\}^*$.[3] □

It follows that when we try to augment a closed theory by a statement that is inconsistent with it we have to abandon completely the old theory. We demonstrate the insertion case with an example of an insertion into a propositional theory.

[3] Clearly, if $\sigma \notin T$, the closed result of deleting σ from T is T, and if $\sigma \in T$, the closed result of inserting σ into T is T.

Example 1. Let T be the propositional theory $\{A, \neg B\}^*$, and suppose that we want to insert B into T. It helps to think of T as $\{A, \neg B, A \equiv \neg B\}^*$, which is another way of describing the same closed theory. There are two closed theories that accomplish the insertion minimally: $T_1 = \{A, B\}^*$ and $T_2 = \{\neg A, B\}^*$. Thus, the closed result of the insertion is the theory T', where $T' = \{(A \& B) \vee (\neg A \& B)\}^* = \{B\}^*$. Consider now non-closed theories. The reader can verify that T' is also a result of inserting B into the theory $\{A, \neg B, A \equiv \neg B\}$. If, however, we insert B into the theory $\{A, \neg B\}$, then the theory $\{A, B\}$ accomplishes the insertion minimally, and hence is a result of the insertion. □

This example demonstrates the difference in updating closed and non-closed theories. This difference explains why we have chosen to consider non-closed theories. If the database has the facts A and $\neg B$ and then B is inserted, then we expect the fact $\neg B$ to be replaced by the fact B. It is true that if both A and $\neg B$ hold then $A \equiv \neg B$ also holds, but the truth of $A \equiv \neg B$ does not seem to be as basic as the truth of A and $\neg B$. If, however, we consider the closure of the theory, then $A \equiv \neg B$ has the same status as A and $\neg B$, and that forces us to delete another fact in addition to $\neg B$, either A or $A \equiv \neg B$.

This example also shows that our logic is *non-monotonic*. Standard logics are monotonic in the sense that adding more axioms or more inference rules enables us to prove new theorems, while the old theorems are still true. In a non-monotonic logic some of the old theorems may be invalidated in such a case. Indeed, the insertion of B into the theory $\{A, \neg B, A \equiv \neg B\}$, forces us to delete all the facts in the theory.

3. Updating Databases

When we try to apply the framework developed in the previous section to updates of databases we encounter difficulties, because in that framework all theories that accomplish an update minimally are equally viable candidates to be the new theory. However, consider a propositional database $\{A\}$ that has the integrity constraint $A \& B \rightarrow C$. The theory of this database is $\{A \& B \rightarrow C, A\}$. The theory $\{A, B\}$ accomplishes the insertion of B into the database minimally. But it does not make sense to throw away an integrity constraint because of an update that violates it. The solution is to give sentences in the databases priorities. Now, instead of arbitrarily constructing theories that accomplish an update minimally, we will construct them by selecting sentences according to their priorities.

A *tagged sentence* is a pair $\langle i, \sigma \rangle$, where i is a natural number and σ is a sentence. A *logical database* is a finite set of tagged sentences. The intention is that the lower the tag, the higher the priority. We shall occasionally ignore the fact that the sentences are tagged and just regard them as sentences, while also talking about their tags. The sentences that are tagged by 0 have the highest priority. We expect the the integrity constraints to be tagged by 0.

Let D be a logical database. Then $D^i = \{\langle j, \tau \rangle : \langle j, \tau \rangle \in D \text{ and } j \leq i\}$ is the set of sentences in D whose tag is smaller or equal to i. (In particular, $D^{-1} = \emptyset$.) The *theory* of D is obtained by stripping the tags, i.e., $Th(D) = \{\tau : \langle i, \tau \rangle \in D\}$. A logical database E *accomplishes the insertion of* σ *into* a logical database D if $\langle i, \sigma \rangle \in E$, for some $i \geq 0$. E *accomplishes the deletion of* σ *from* D if $\sigma \notin Th(E)^*$.

Now, when we compare two logical databases to see which of them accomplishes an update with a smaller change, we compare them according to the priorities given to the sentences. Let D be a logical database with n as the highest tag in it, and let E and F be two logical databases that accomplish an update u. We say that E *accomplishes u with a smaller change than* F if either for some i, $0 \leq i \leq n$, we have either

$$D^{i-1} - E^{i-1} = D^{i-1} - F^{i-1},$$

but

$$D^i - E^i \subset D^i - F^i,$$

or else

$$D^n - E^n = D^n - F^n$$

but

$$E - D \subset F - D.$$

Remark. The above definition has also the advantage that it facilitates attaching an authorization mechanism to the update mechanism by specifying for every user a tag i such that E^i must be equal to D^i. For example, even though the definition allows a change in D^0, i.e., the integrity constraints, we anticipate that most users will not be authorized to update that part of the database. □

We say that a logical database E accomplishes u minimally if there is no logical database F that accomplishes u with a smaller change than E.

The next theorem is the analog of Theorem 1 for logical databases.

Theorem 4. Let D and E be logical databases, with n the highest tag in D, and let σ be a sentence.

(1) E accomplishes the deletion of σ from D minimally if and only if E^i is a maximal subset of D^i that is consistent with $\neg\sigma$ for $i = 1, \ldots, n$.

(2) $E \cup \langle j, \sigma \rangle$ accomplishes the insertion of σ into D minimally if and only if E^i is a maximal subset of D^i that is consistent with σ for $i = 1, \ldots, n$. □

As is the case with theories, a result of an update should be a logical database whose class of models is exactly the union of all classes of models of logical databases that accomplish the update minimally. If $D_1, \ldots, D_n$ are the logical databases that accomplish an update u minimally, and D' is a logical database such that

$$Mod(Th(D')) = \bigcup_{i=1}^{n} Mod(Th(D_i)),$$

then D' is a result of u. Lemma 3 gives us a way to construct such a result, because if

$$Th(D')^* = (\bigvee_{i=1}^{n} Th(D_i))^*,$$

then D' is a result of u.

What we have not specified is how to convert a theory into a logical database; that is, how to tag the sentences in a meaningful way. This is left to the database administrator to specify, since this is the means through which he can control the actual implementation of updates. We shall illustrate this point with an example.

Example 2. Let the database consist of a relation $R(Employee, Child, Department)$ with the functional dependency $Employee \rightarrow Department$. Let the current relation be

Employee	*Child*	*Department*
Gauss	*Yoni*	*Math*
Turing	*Yoram*	*Math*
Turing	*Gabi*	*Math*
Babbage	*Andrei*	*CS*

The language we use has a relation name R for the relation R and constants for the elements of the domain. The integrity constraints consist of the given functional dependency plus distinctness axioms saying that all elements are distinct [Rei]. The logical database D consists of the integrity constraints tagged by 0, existential sentences, tagged by 1, describing the tuples in the projection of the relation on columns *Employee* and *Child*, and atomic sentences, tagged by 2, describing the tuples in the relation. That is, D^0 is the set

$$\{\langle 0, \forall y_1 \cdots y_5(\mathrm{R}(y_1,y_2,y_3) \& \mathrm{R}(y_1,y_4,y_5) \rightarrow y_3 = y_5)\rangle,$$
$$\langle 0, Gauss \neq Turing\rangle, \ldots, \langle 0, Math \neq CS\rangle\},$$

D^1 is the set

$$D^0 \cup$$
$$\{\langle 1, \exists x(\mathrm{R}(Gauss, Yoni, x))\rangle,$$
$$\langle 1, \exists x(\mathrm{R}(Turing, Yoram, x))\rangle,$$
$$\langle 1, \exists x(\mathrm{R}(Turing, Gabi, x))\rangle,$$
$$\langle 1, \exists x(\mathrm{R}(Babbage, Andrei, x\rangle\}.$$

and D^2 is the set

$$D^1 \cup$$
$$\{\langle 2, \mathrm{R}(Gauss, Yoni, Math)\rangle,$$
$$\langle 2, \mathrm{R}(Turing, Yoram, Math)\rangle,$$
$$\langle 2, \mathrm{R}(Turing, Gabi, Math)\rangle,$$
$$\langle 2, \mathrm{R}(Babbage, Andrei, CS)\rangle\},$$

Suppose now that we insert into D a sentence σ, where σ is $\exists x(\mathrm{R}(Turing, x, CS))$. Let us now construct a logical database that accomplishes the insertion minimally. In fact, there is a unique such logical database D'. D^0 is consistent with σ so we put it in D'. Actually, D^1 is also consistent with σ so we can put it in D'. We can also put $\langle 2, \mathrm{R}(Gauss, Yoni, Math)\rangle$ and $\langle 2, \mathrm{R}(Babbage, Andrei, CS)\rangle$ in D'. However, we can put neither $\langle 2, \mathrm{R}(Turing, Yoram, Math)\rangle$ nor $\langle 2, \mathrm{R}(Turing, Gabi, Math)\rangle$ in D', because then the functional dependency together with σ entail $Math = CS$, in contradiction to the axiom $Math \neq CS$, which was already put in D'. It follows that

$$D' = \{\langle i, \sigma\rangle\} \cup D^1 \cup$$
$$\{\langle 2, \mathrm{R}(Gauss, \cdots)\rangle, \langle 2, \mathrm{R}(Babbage, \cdots\rangle\}$$

is the unique (up to the choice of i) logical database that accomplish the insertion minimally. The reader can verify that $Th(D')$ logically implies both R($Turing, Yoram, CS$) and R($Turing, Gabi, CS$). It follows that the logical database describing the extension

Employee	*Child*	*Department*
Gauss	*Yoni*	*Math*
Turing	*Yoram*	*CS*
Turing	*Gabi*	*CS*
Babbage	*Andrei*	*CS*

is a result of the insertion. Thus, the net effect of the insertion is to change Turing's department from Math to CS. Note how our decision to have the *Employee-Child* facts in the logical database was essential for us to obtain this result. This is an example how the database administrator can control the implementation of updates. □

4. Updating Views

A problem that has attracted a lot of interest is that of updating views. The problem is usually posed in the following way. A *user view* (or view, for short) is defined by some mapping α on the collection of possible databases. The intention is that when the database is B, the user sees $V = \alpha(B)$. Suppose now that the user wants to apply an update operation u that will effect a certain change in V. This should be accomplished by applying some update operation v to B so that the change in B "reflects" the change in V "correctly". There are two major difficulties here. First, it is not clear what is the right way to define formally the above intuitive notion of "correct reflection". Secondly, it is not clear how to construct v given u.

In our opinion the difficulties arise from the premise that the user knows exactly how the update is going to effect his view. The user view, however, has to satisfy certain integrity constraints, either because the database has to satisfy some integrity constraints or because it has to be an image of α. For example, it is easy to show that if the view is defined by a join operation then it has to satisfy a certain join dependency. As observed earlier, the effect of updates in the presence of integrity constraints is not transparent at all.

We believe that a better approach is to consider an update operation as an addition or deletion of an information unit. If the user expresses that information in terms of his view, then we should first translate that into information in terms of the database and then apply the methodology of the preceding sections. The crucial point is shifting the focus from the change that the update is supposed to effect to the information that it carries. Let us now formalize our approach.

We assume for simplicity that the database consists of one n-ary relation R, the view consists of one m-ary relation P, and both the database and the view have the same underlying domain A. The relation names are R and P, respectively. For any relation name Q, we use the notation $\tau(\mathrm{Q}, x_1, \ldots, x_k)$ to denote a formula τ that has Q as its only relation name and has $x_1, \ldots, x_k$ as free variables. The *view definition* is a formula $\varphi(\mathrm{R}, x_1, \ldots, x_m)$. Thus, given a database $B = (A, R)$, the view V is (A, P), where

$$P = \{\langle a_1, \ldots, a_m \rangle : (A, R) \models \varphi(\mathrm{R}, a_1, \ldots, a_m)\}.$$

Suppose now that we have a sentence $\sigma(\mathrm{P})$, which conveys some information about the view. What information does it convey about the database? In order to answer that, it helps to lump both the database and the view into an extended structure (A, R, P). By the way P was defined, we know that this extended structure satisfies the sentence $\psi(\mathrm{P}, \varphi(\mathrm{R}))$:

$$\forall x_1 \cdots x_m (\mathrm{P}(x_1, \ldots, x_m) \equiv \varphi(\mathrm{R}, x_1, \ldots, x_m)).$$

The desired translation is given by the following lemma.

Lemma 4. Let $\varphi(\mathrm{R}, x_1, \ldots, x_m)$ be the view definition, let $\sigma(\mathrm{P})$ be a sentence, and let $\sigma(\varphi(\mathrm{R}))$ be the result of replacing each atomic formula $\mathrm{P}(v_1, \ldots, v_m)$ by

$\varphi(\mathrm{R}, v_1, \ldots, v_m)$. Then

$$\psi(\mathrm{P}, \varphi(\mathrm{R})) \models \sigma \equiv \sigma(\varphi(\mathrm{R})). \square$$

Intuitively, Lemma 4 says that the information that $\sigma(\mathrm{P})$ conveys about P is equivalent to the information that $\sigma(\varphi(\mathrm{R}))$ conveys about R. In the terms of [Jal] $\sigma(\varphi(\mathrm{R}))$ is the *interpretation* of $\sigma(\mathrm{P})$ under $\varphi(\mathrm{R}, x_1, \ldots, x_m)$. Thus, if the user asks to insert $\sigma(\mathrm{P})$ into his view or to delete $\sigma(\mathrm{P})$ from his view, the update should be implemented by inserting $\sigma(\varphi(\mathrm{R}))$ into the database or deleting $\sigma(\varphi(\mathrm{R})$ from the database by the methodology of the preceding sections.

Remark. Our approach also gets around another difficulty. While the database can be described by a finite logical database, it is not clear what is the logical description of the view. Let $T(\mathrm{R})$ be the theory of the logical database. Then the theory of the view is

$$\{\sigma(\mathrm{P}) : T(\mathrm{R}) \cup \{\psi(\mathrm{P}, \varphi(\mathrm{R}))\} \models \sigma(\mathrm{P})\}.$$

This theory is not only infinite but may not even be finitely axiomatizable. In practice the user is going to see only a subset of this theory, say all quantifier-free sentences. Nevertheless, the update can be implemented without any reference to what the user actually sees. □

Another methodology for updating views, based on the notion of *complementary view*, was suggested in [BS]. We say that two views, defined by the mappings α and β, respectively, are complementary if the mapping (α,β), defined by $(\alpha,\beta)(B) = (\alpha(B), \beta(B))$, is an injective mapping. The basic idea is to accompany each view-defining mapping α by a complementary view defining mapping β. Suppose now that the database is B, $V = \alpha(B)$, $W = \beta(B)$, and the user wants to change V to V'. If there is a database B' such that $(\alpha,\beta)(B') = (V', W)$ then there is a unique one, since (α,β) is injective. Thus, changing B to B' is the correct translation of the update.

While the idea is quite attractive it is not very constructive. Not only it is unclear how to verify that the two views are complementary and how to find B' given V' and W, but, as already argued, the basic premise, that the user knows what change his update is going to effect, is dubious. Nevertheless, the spirit of the methodology can be captured by our approach of assigning priorities to sentences. We now present the logical framework.

Recall that the database consists of a relation R with relation name R, and that the view consists of a relation P with relation name P, defined by $\varphi(\mathrm{R}, x_1, \ldots, x_m)$. In addition we now have another view, consisting of an l-ary relation Q with relation name Q, defined by $\chi(\mathrm{R}, x_1, \ldots, x_l)$. We use x to denote a sequence $x_1, x_2, \ldots$ of variables, where the length of the sequence is to be determined by the context. Let R' be a new n-ary relation name, let $\varphi(\mathrm{R}',\mathrm{x})$ be the result of replacing R by R' in $\varphi(\mathrm{R},\mathrm{x})$, and similarly for $\chi(\mathrm{R}',\mathrm{x})$.

The following theorem gives a necessary and sufficient logical condition for complementariness.

Theorem 5. Let $\varphi(\mathrm{R},\mathrm{x})$ and $\chi(\mathrm{R},\mathrm{x})$ be view definitions. Then φ and χ define complementary views if and only if

$$\forall \mathrm{x}(\varphi(\mathrm{R},\mathrm{x}) \equiv \varphi(\mathrm{R}',\mathrm{x})) \& \forall \mathrm{x}(\chi(\mathrm{R},\mathrm{x}) \equiv \chi(\mathrm{R}',\mathrm{x}))$$

$$\models \forall \mathrm{x}(\mathrm{R}(\mathrm{x}) \equiv \mathrm{R}'(\mathrm{x})). \square$$

Remark. Theorem 5 has a very interesting corollary, based on Beth's Definability Theorem [Be]. If $\varphi(\mathrm{R},\mathrm{x})$ and $\chi(\mathrm{R},\mathrm{x})$ define complementary views, then there is a formula $\rho(\mathrm{P}, \mathrm{Q}, x_1, \ldots, x_n)$, such that if

$$P = \{\langle a_1, \ldots, a_m \rangle : (A, R) \models \varphi(\mathrm{R}, a_1, \ldots, a_m)\},$$

and

$$Q = \{\langle a_1, \ldots, a_l \rangle : (A, R) \models \chi(\mathrm{R}, a_1, \ldots, a_l)\},$$

then

$$R = \{\langle a_1, \ldots, a_m \rangle : (A, P, Q) \models \rho(\mathrm{P}, \mathrm{Q}, a_1, \ldots, a_n)\}.$$

That is, if the two complementary mappings α and β are given as first-order formulas, then the inverse of the mapping (α,β) is also given by a first-order formula. We note that the proof of this corollary assumes that infinite databases are also considered. □

The basic idea in the complementary views approach is that some information has to be kept invariant while the update is performed. Let D be the logical database, and let $T(Q)$ be a theory about the complementary view. In analogy with $\psi(P, \varphi(R))$, let $\psi(Q, \chi(R))$ be the sentence

$$\forall x(Q(x) \equiv \chi(R,x)).$$

By Lemma 4 we know that, assuming that $\psi(Q,\chi(R))$ holds, $T(Q)$ is logically equivalent to

$$T(\chi(R)) = \{\sigma(\chi(R)) : \sigma(Q) \in T(Q)\}.$$

Thus, keeping $T(Q)$ invariant when an update involving $\sigma(P)$ is performed is equivalent to keeping $T(\chi(R))$ invariant when that update is performed with $\sigma(\varphi(R))$. In order to achieve that, we perform the update on a new logical database D':

$$\{\langle 0,\tau\rangle : \tau \in T(\chi(R))\} \cup \{\langle i+1,\tau\rangle : \langle i,\tau\rangle \in D\}.$$

That is, we add $T(\chi(R))$ to the logical database, while giving it a higher priority.

Example 3. Consider again the relation $R(Employee,Child,Department)$ with the functional dependency $Employee \rightarrow Department$ of Example 2. Let the logical database D be:

$$\{\langle 0, \forall y_1 \cdots y_5(R(y_1,y_2,y_3) \& R(y_1,y_4,y_5) \rightarrow y_3 = y_5)\rangle,$$

$$\langle 0, Gauss \neq Turing\rangle, \ldots, \langle 0, Math \neq CS\rangle,$$

$$\langle 1, R(Gauss, Yoni, Math)\rangle, \ldots,$$

$$\langle 1, R(Babbage, Andrei, CS)\rangle\}.$$

The view that we are interested in is the relation $P(Employee,Department)$ obtained by projecting R on the columns *Employee* and *Department*. Thus, its defining formula $\varphi(R, x_1,x_2)$ is:

$$\exists x(R(x_1,x,x_2)).$$

The complementary view is the relation $Q(Employee,Child)$ obtained by projecting R on the columns *Employee* and *Child*. Thus, its defining formula $\chi(R, x_1,x_2)$ is:

$$\exists x(R(x_1,x_2,x)).$$

The relation names of P and Q are P and Q, respectively. Suppose that the user asks to insert to his view the tuple

Employee	*Department*
Turing	*CS*

i.e., the formula $\sigma(P)$ is:

$$P(Turing,CS).$$

and $\sigma(\varphi(R))$ is:

$$\exists x(R(Turing,x,CS)).$$

The information that we want to keep invariant is the relation $Q(Employee,Child)$:

Employee	*Child*
Gauss	*Yoni*
Turing	*Yoram*
Turing	*Gabi*
Babbage	*Andrei*

Thus, $T(Q)$ is:

$$\{Q(Gauss,Yoni), \ldots, Q(Babbage,Andrei)\},$$

and $T(\chi(R))$ is

$$\{\exists x(R(Gauss, Yoni, x)), \ldots,$$
$$\exists x(R(Babbage, Andrei, x)\}.$$

Now, D', the new logical database is:

$$\{\langle 0, \exists x(R(Gauss, Yoni, x))\rangle, \ldots,$$
$$\langle 0, \exists x(R(Babbage, Andrei, x))\rangle,$$
$$\langle 1, \forall y_1 \cdots y_5(R(y_1,y_2,y_3) \& R(y_1,y_4,y_5) \rightarrow y_3 = y_5)\rangle,$$
$$\langle 1, Gauss \neq Turing\rangle, \ldots, \langle 1, Math \neq CS\rangle,$$
$$\langle 2, R(Gauss, Yoni, Math)\rangle, \ldots,$$
$$\langle 2, R(Babbage, Andrei, CS)\rangle\},$$

We leave it to the reader to verify that the effect of the update is change Turing's department from Math to CS, as in Example 2. □

5. Concluding Remarks

In the previous sections we described a framework for updates of theories and logical databases, which is also applicable to updating views. As was demonstrated in Example 2, the actual result of an update is dependent upon the way the logical database is constructed. More research should done on that aspect to make our methodology practical.

As an example, we shall propose, in the full version of the paper, a specific construction for relational databases. We shall show that if we restrict ourselves to databases where the initial state has complete information and the integrity constraints are data dependencies, then as a result of insertions and deletions we get a theory that is very similar to the generalized relational theory with nulls of Reiter [Rei]. This is meant to be a demonstration of the methodology and not a practical approach, since it is computationally quite intractable: insertions are NP-hard and deletions cause combinatorial explosion of disjuncts.

Until now we have assumed that our constraints are *state laws*, i.e., they deal with the legality of database states. In contrast, *transition laws* are constraints that deal with the legality of transitions between states. For example, the constraint "ages can never decrease" is a transition law. The way we deal with transition laws is again to lump both the present and the next database states into an extended database, in which state laws express the transition laws. Obviously, we require that the update has no effect on the part of the extended database that reflects the present state.

Acknowledgement. We would like to thank Jim Davidson, Joan Feigenbaum, Anna Karlin, Arthur Keller, Gabi Kuper, John McCarthy, and Allen VanGelder, for helpful discussions and suggestions.

REFERENCES

[Be] Beth, E.W.: On Padoa's method in the theory of definitions. Indag. Math. 15 (1953), pp. 330-339.

[BS] Bancilhon, F., Spyratos, N.: Update semantics of relational views. ACM Trans. on Database Systems 6 (1981), pp. 557-575.

[CA] Carlson, C.R., Arora, A.K.: The updatability of relational views based on functional dependencies. Proc. IEEE COMPSAC, 1979, pp. 415-420.

[Cl] Clemons, E.K.: An external schema facility to support database update. In *Database*, Academic Press, 1978.

[Da] Dayal, U.: Schema mapping problems in database systems. Technical Report TR-11-79, Center for Research in Computing Technology, Harvard University, 1979.

[DB1] Dayal, U., Bernstein, P.A.: On the updatability of relational views. Proc. 4th Int'l Conf. on VLDB, Berlin, 1978, pp. 368-377.

[DB2] Dayal, U., Bernstein, P.A.: Translation of update operations on relational views. ACM Trans. on Database Systems 8 (1982), pp. 381-416.

[DL] Doyle, J., London, P.: A selected descriptor-indexed bibliography to the literature on belief revision. SIGART Newsletter 71(1980), pp. 7-23.

[FS] Furtado, A.L., Sevcik, K.C.: Permitting updates through views of databases. Inf. Syst. 4 (1979), pp. 269-283.

[Ja1] Jacobs, B.E.: On interpretations in database logic and their applications. Technical Report TR 815, Dept. of Computer Science, University of Maryland at College Park, 1979.

[Ja2] Jacobs, B.E.: Application of database logic to the view update problem. Technical Report TR 960, Dept. of Computer Science, University of Maryland at College Park, 1980.

[KD] Kaplan, S.J., Davidson, J.: Interpreting natural language database updates, Proc. 19th Ann. Meeting of the Assoc. for Computational Linguistics, Stanford, 1981, pp. 139-142.

[Ke] Keller, A.M.: Updates to relational databases through views involving joins. In *Improving database usability and responsiveness* (P. Scheuermann, ed.), Academic Press, 1982, pp. 363-384.

[Kl] Klug, A.C.: Theory of database mapping. Technical Report CSRG-98, Dept. of Computer Science, University of Toronto, 1978.

[Ko] Kowalski, R.: Logic as database language. Unpublished Manuscript, Dept. of Computing, Imperial College, London, 1981.

[Mi] Minker, J.: On indefinite databases and the closed world assumption. Technical Report TR-1076, Dept. of Computer Science, University of Maryland at College Park, July 1981.

[MUV] Maier, D., Ullman, J.D., Vardi, M.Y.: The revenge of the JD. Proc. 2nd ACM Symp. on Principles of Database Systems, Atlanta, 1983.

[NG] Nicolas, J.M., Gallaire, H.: Database - theory vs. interpretation. In *Logic and Databases* (H. Gallaire and J. Minker, eds.), Plenum Press, 1978, pp. 33-54.

[NY] Nicolas, J.M., Yazdanian, K.: Integrity checking in deductive databases. In *Logic and Databases* (H. Gallaire and J. Minker, eds.), Plenum Press, 1978, pp. 325-344.

[Os] Osman, I.M.: Updating defined relations. Proc. Nat'l Computer Conf., Vol. 48,, AFIP Press, 1979, pp. 733-740.

[Rei] Reiter, R.: Towards a logical reconstruction of relational database theory. Unpublished Manuscript, University of British Columbia, 1981.

[Res] Resher, N.: *Hypothetical Reasoning*, North-Holland, Amsterdam, 1964.

[Sc] Sciore, E.: The universal instance and database design. Technical Report TR-271, Dept. of EECS, Princeton University, 1980.

[To] Todd, S.: Automatic constraint maintenance and updating defined relations. Proc. IFIP 77 (B. Gilchrist, ed.), North-Holland, 1977, pp. 145-148.